Efficient Computations of the Tate Pairing for the Large MOV Degrees

Tetsuya Izu[1] and Tsuyoshi Takagi[2]

[1] FUJITSU LABORATORIES Ltd.,
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
izu@labs.fujitsu.com
[2] Technische Universität Darmstadt, Fachbereich Informatik,
Alexanderstr.10, D-64283 Darmstadt, Germany
ttakagi@cdc.informatik.tu-darmstadt.de

Abstract. The Tate pairing has plenty of attractive applications, e.g., ID-based cryptosystems, short signatures, etc. Recently several fast implementations of the Tate pairing has been reported, which make it appear that the Tate pairing is capable to be used in practical applications. The computation time of the Tate pairing strongly depends on underlying elliptic curves and definition fields. However these fast implementation are restricted to supersingular curves with small MOV degrees. In this paper we propose several improvements of computing the Tate pairing over general elliptic curves over finite fields $\mathbb{F}_q$ $(q = p^m, p > 3)$ — some of them can be efficiently applied to supersingular curves. The proposed methods can be combined with previous techniques. The proposed algorithm is specially effective upon the curves that has a large MOV degree k. We develop several formulas that compute the Tate pairing using the small number of multiplications over $\mathbb{F}_{q^k}$. For $k = 6$, the proposed algorithm is about 20% faster than previously fastest algorithm.

Keywords: Elliptic curve cryptosystem, Tate pairing, Jacobian coordinate, MOV degree, efficient computation.

1 Introduction

After the proposal of the cryptosystems by Koblitz and Miller [Kob87,Mil86], elliptic curves have attracted a lot of cryptographic interests. Menezes-Okamoto-Vanstone found some weak curves, the supersingular curves, on which the Weil pairing transforms the cyclic group on the elliptic curve into a finite filed with small extension degree (MOV degree) [MOV93]. Then Frey-Müller-Rück extended their attack and found more weak curves by using the Tate pairing [FMR99]. These curves are avoided for cryptographic use. Now, elliptic curves for cryptography have been standardized by many organizations [IEEE,NIST,SEC], on which the transformation just produces a finite filed with huge extension.

Recently, Okamoto-Pointcheval found a new class of problems in which the Decision Diffie-Hellman (DDH) problems are easy but the Diffie-Hellman (DH) problems are still hard [OP01]. The Weil and the Tate pairings exactly provide this problem and they are used as a constructive tool for cryptography. Indeed, an enormous number of primitives have been proposed [Jou02], such as the tripartite Diffie-Hellman scheme [Jou00], the key agreement scheme [Sma01], the encryption scheme [BF01], the signature scheme [BLS01,Hes02,Pat02,SOK00], and the self-blindable credential certificates [Ver01]. These primitives require the specific elliptic curves, rather than standardized curves in [IEEE,NIST,SEC], on which the transformation by the pairings produces a small extension degree. Supersingular elliptic curves are suitable for this purpose [Gal01]. Recent results [MNT01,DEM02], which construct ordinary elliptic curves with given (small) extension degree, enable us to use ordinary curves in these primitives and provide more freedom for cryptographic applications. On the other hand, fast implementation of the pairings have been studied. Elliptic curves over characteristics 2 and 3 fields are attractive, because point doubling in characteristic 2 (tripling in characteristic 3) can be performed very efficiently [BKLS02,GHS02,SW02]. However, when we implement the pairing based cryptosystems in software or hardware, it is natural to use existing familiar technologies, namely characteristic p fields. They are easy to be implemented because there are massive results of researches since the RSA. Recently, it is reported that characteristic 3 fields are less efficient for hardware implementation [PS02]. Moreover, the discrete logarithm problem over $\mathbb{F}_{2^m}$ can be solved faster than that over other finite fields, and thus the size of finite field $\mathbb{F}_{2^m}$ must be chosen larger [Cop83].

Contribution of This Paper

In this paper, we pursuit the fast computation of the Tate pairing over characteristic p fields. Previous studies, such as [BKLS02,GHS02], were dedicated to the other characteristics or supersingular curves. Our main target is a general (not necessarily supersingular) elliptic curve with arbitrary extension degree. We propose several efficient computations of the Tate pairing. They are specially effective upon large extension degrees e.g., $k > 2$ — some of them can be efficiently applicable to $k = 2$ as well. Our algorithms are independent from the previously proposed methods and we can combine our proposed method with them.

The computation of the Tate pairing consists of three stages: (1)To compute the elliptic curve addition or doubling, (2)To compute the coefficients of two lines l_1 and l_2, (3)To evaluate the values $l_1(Q), l_2(Q)$ and update the value of the Tate pairing. In this paper, we improve/optimize the computation time of each step. For the first step, we develop a new coordinate, called *the simplified Chudonovsky-Jacobian coordinate* $\mathcal{J}^s$. We also proposed an addition formula that directly computes $(2^w)Q$ instead of repeatedly applying an elliptic curve doubling. These modifications can save several multiplications. For the second step, we optimize the generation of the coefficients of lines l_1, l_2. For the third step, we encapsulate the lines l_1, l_2 into one quadratic equation, and thus we can avoid the computation over extension fields $\mathbb{F}_{q^k}$. This modification can reduce the number of multiplication of $\mathbb{F}_{q^k}$. The efficiency of all the proposed methods are estimated. We also demonstrate how the proposed methods improve the whole computation of the Tate pairing. For $k = 6$, the proposed methods are about 20% faster than the previously known algorithms.

This paper is organized as follows: In Section 2, we review the arithmetic of elliptic curves and the definition of the Tate pairing. In Section 3, the computation of the Tate pairing using the Miller's algorithm is described. In Section 4, we present our proposed algorithms. In Section 4.5 the application of our algorithm to supersingular curves is discussed. In Section 5 we compare our proposed algorithm with the previously known algorithms. In Section 6 we state our occlusion of this paper.

2 Preliminaries

In this section we explain the basic arithmetic of elliptic curves and Tate pairing. We assume that base field $K = \mathbb{F}_q$ $(q = p^m, p > 3)$ is a finite field with q elements in this paper, where p is called the characteristic of K.

2.1 Elliptic Curves

Elliptic curves over K can be represented by the Weierstrass-form equation

$$E(K) := \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b \ (a, b \in K,\ 4a^3 + 27b^2 \neq 0)\} \cup \mathcal{O}, \quad (1)$$

where $\mathcal{O}$ is the point of infinity. An elliptic curve $E(K)$ has an additive group structure. Let $P_1 = (x_1, y_1),\ P_2 = (x_2, y_2)$ be two elements of $E(K)$ that are different from $\mathcal{O}$ and satisfy $P_2 \neq \pm P_1$. Then the addition $P_3 = P_1 + P_2 = (x_3, y_3)$ is defined by $x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = (y_2 - y_1)/(x_2 - x_1)$ for $P_1 \neq P_2$, and $\lambda = (3x_1^2 + a)/(2y_1)$ for $P_1 = P_2$. For two points $P_1,\ P_2$ of $E(K)$, we call $P_1 + P_2$ $(P_1 \neq P_2)$ the elliptic curve addition (ECADD) and $P_1 + P_2$ $(P_1 = P_2)$, that is $2 * P_1$, the elliptic curve doubling (ECDBL). For a given integer d and a point P on the elliptic curve $E(K)$, compute the point $d * P$ is called the scalar multiplication.

2.2 Coordinate System

There are several ways to represent a point on an elliptic curve. The costs of computing an ECADD and an ECDBL depend on the representation of the coordinate system. The detailed description of the coordinate systems is given in [CMO98]. The major coordinate systems are as follows: the affine coordinate ($\mathcal{A}$), the projective coordinate ($\mathcal{P}$), the Jacobian coordinate ($\mathcal{J}$), the Chudonovsky-Jacobian coordinate ($\mathcal{J}^C$), and the modified Jacobian coordinate ($\mathcal{J}^m$). We summarize the costs in Table 1, where $M,\ S,\ I$ denotes the computation time of a multiplication, a squaring, and an inverse in the definition field K, respectively. The speed of ECADD or ECDBL can be enhanced when the third coordinate is $Z = 1$ or the coefficient of the definition equation is $a = -3$.

The Jacobian coordinate offers a faster ECDBL (but a slower ECADD). The equation of the curve is given by $E_J : Y^2 = X^3 + aXZ^4 + bZ^6$ by setting

Coordinate	ECADD		ECDBL	
System	$Z \neq 1$	$Z = 1$	$a \neq -3$	$a = -3$
$\mathcal{A}$	$2M + 1S + 1I$	—	$2M + 2S + 1I$	
$\mathcal{P}$	$12M + 2S$	$9M + 2S$	$7M + 5S$	$7M + 3S$
$\mathcal{J}$	$12M + 4S$	$8M + 3S$	$4M + 6S$	$4M + 4S$
$\mathcal{J}^C$	$11M + 3S$	$8M + 3S$	$5M + 6S$	$5M + 4S$
$\mathcal{J}^m$	$13M + 6S$	$9M + 5S$	$4M + 4S$	

Table 1. Computing times of an addition (ECADD) and a doubling (ECDBL)

$x = X/Z^2,\ y = Y/Z^3$ in (1), and a point on the curve is represented by (X, Y, Z) where two points (X, Y, Z) and $(\lambda^2 X, \lambda^3 Y, \lambda Z)$ $(\lambda \neq 0)$ are identified as same. The addition formulas for Jacobian coordinate are given in Table 2.

`ECADD` $(8M + 3S)$	`ECDBL` $(4M + 6S)$
Input: $P_1 = (X_1, Y_1, Z_1), P_2 = (X_2, Y_2, 1)$	Input: $P_1 = (X_1, Y_1, Z_1),\ a$
Output: $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$	Output: $P_4 = 2 * P_1 = (X_4, Y_4, Z_4)$
$U_1 \leftarrow X_1,\ U_2 \leftarrow X_2 Z_1^2$	$M \leftarrow 3X_1^2 + aZ_1^4$
$S_1 \leftarrow Y_1,\ S_2 \leftarrow Y_2 Z_1^3$	$S \leftarrow 4X_1 Y_1^2$
$H \leftarrow U_2 - U_1,\ R \leftarrow S_2 - S_1$	$X_4 \leftarrow M^2 - 2S$
$X_3 \leftarrow -H^3 - 2U_1 H^2 + R^2$	$Y_4 \leftarrow M(S - X_4) - 8Y_1^4$
$Y_3 \leftarrow -S_1 H^3 + R(U_1 H^2 - X_3)$	$Z_4 \leftarrow 2Y_1 Z_1$
$Z_3 \leftarrow Z_1 H$	

Table 2. Addition formulas in Jacobian coordinate

Chudonovsky-Chudonovsky proposed the *Chudonovsky-Jacobian coordinate* $\mathcal{J}^C$, in which a point is represented by a quintuplet (X, Y, Z, Z^2, Z^3) where (X, Y, Z) represents a point in the Jacobian coordinate. In the Chudonovsky-Jacobian coordinate, there is no need to compute a squaring and a tripling of Z-coordinates of inputs because they are in the coordinate, but need to compute a squaring and a tripling of Z-coordinates of an output.

2.3 Addition Chain

Let d be an n-bit integer and P be a point of the elliptic curve E. A standard way for computing a scalar multiplication $d * P$ is to use the binary expression $d = d[n-1]2^{n-1} + d[n-2]2^{n-2} + \ldots + d[1]2 + d[0]$, where $d[n-1] = 1$ and $d[i] = 0, 1$ $(i = 0, 1, ..., n-2)$. The binary method computes an ECDBL for every $d[i]$ and an ECADD if $d[i] \neq 0$. In average it requires $(n-1)$ ECDBLs and $(n-1)/2$ ECADDs. Because computing the inverse $-P$ of P is essentially free, we can relax the condition "binary" to "signed binary" $d = \sum_{i=0}^{n-1} d[i]2^i$, where $d[i] = -1, 0, 1$. It is called the signed binary method (or the addition-subtraction method). NAF offers a way to construct the addition-subtraction chain, which requires $(n-1)$ ECDBLs and $(n-1)/3$ ECADDs in average [IEEE] for an n-bit scalar multiplication. We denote the signed binary expression obtained by NAF as $d = \sum NAF(d)[i]2^i$. In the binary methods, points P and $-P$ are constant that we can set Z-coordinates of them to 1 for an efficiency reason.

2.4 Tate Pairing

Let ℓ be a positive integer coprime to q (In most cryptographic primitives, ℓ is set to a prime such that $\ell | \#E(\mathbb{F}_q)$). Let k be the smallest positive integer such that the ℓ-th root of unity is in $\mathbb{F}_{q^k}$, namely $\ell | (q^k - 1)$. k is called the *MOV degree* [MOV93]. Let $E(\mathbb{F}_{q^k})[\ell]$ be the subgroup of points in $E(\mathbb{F}_{q^k})$ of order ℓ. Then the Tate pairing $\langle \cdot, \cdot \rangle$ is defined by

$$\langle \cdot, \cdot \rangle : E(\mathbb{F}_{q^k})[\ell] \times E(\mathbb{F}_{q^k})/\ell E(\mathbb{F}_{q^k}) \mapsto \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^\ell$$

where the right hand value is modulo ℓ-th powers.

The Tate pairing is computed via the following function f_P. Here P is a point of order ℓ. There is a function f_P whose divisor $div(f)$ is equal to $\ell(P) - \ell(\mathcal{O})$. Then we have $\langle P, Q \rangle = f(\overline{Q})$ where $\overline{Q}$ denotes a divisor in the same class as Q such that the support of $\overline{Q}$ is disjoint with the support of (f). This is done by setting $\overline{Q} = (Q + S) - (S)$ where $(Q) - (\mathcal{O})$ is the divisor and $S \in E(\mathbb{F}_{q^k})$. For cryptographic applications, values of the Tate pairing are expected to be unique. Thus the Tate pairing is computed by

$$\langle P, Q \rangle = (f_P(Q + S)/f_P(S))^{(q^k-1)/\ell}. \tag{2}$$

The properties of the Tate pairing are as follows [GHS02]:

1. (Well-defined) $\langle \mathcal{O}, Q \rangle \in (\mathbb{F}^*_{q^k})^\ell$ for all $Q \in E(\mathbb{F}_{q^k})[\ell]$ and $\langle P, Q \rangle \in (\mathbb{F}^*_{q^k})^\ell$ for all $P \in E(\mathbb{F}_{q^k})[\ell]$, $Q \in \ell E(\mathbb{F}_{q^k})$.
2. (Non-degeneracy) For each point $P \in E(\mathbb{F}_{q^k}) - \{\mathcal{O}\}$, there exist a point $Q \in E(\mathbb{F}_{q^k})$ such that $\langle P, Q \rangle \notin (\mathbb{F}^*_{q^k})^\ell$.
3. (bilinearity) For any integer n, $\langle nP, Q \rangle \equiv \langle P, nQ \rangle \equiv \langle P, Q \rangle^n$ modulo ℓ-th power.

We describe the standard key sizes of q, k, l in the following. q^k is at least larger than 1024 bits in order to make the discrete logarithm problem over $\mathbb{F}_{q^k}$ intractable. l is at least larger than 160 bits in order to resist the baby-step-giant-step algorithm or Pollard's λ algorithm for solving the short discrete logarithm problem of $(f_P(Q + S)/f_P(S))^{(q^k-1)/\ell} \in \mathbb{F}_{q^k}$.

3 Computing the Tate Pairing

In this section we estimate the computing time of the Tate pairing via the Miller's algorithm.

3.1 Miller's Algorithm

A standard algorithm for computing the $f_P(Q+S)/f_P(S)$ (in the Tate pairing) is the Miller's algorithm [Men93]. Let $\ell[i]$ be the bit representation of an n-bit prime ℓ where $\ell[n-1] = 1$. We review the Miller's algorithm in Table 3.

Input: ℓ, $P \in E(\mathbb{F}_q)$, $Q, S \in E(\mathbb{F}_{q^k})$
Output: $f_P(Q+S)/f_P(S) \in \mathbb{F}_{q^k}$

```
1: $T = P$, $f = 1$
2: For $i = n-1$ down to 0
3:   Compute $T = \mathtt{ECDBL}(T)$ and lines $l_1, l_2$ for $T+T$
4:   $f \leftarrow f^2 \times \frac{l_1(Q+S) \times l_2(S)}{l_1(S) \times l_2(Q+S)}$
5:   If $\ell[i] = 1$ then
6:     Compute $T = \mathtt{ECADD}(T, P)$ and lines $l_1, l_2$ for $T+P$
7:     $f \leftarrow f \times \frac{l_1(Q+S) \times l_2(S)}{l_1(S) \times l_2(Q+S)}$
8: return $f$
```

Table 3. Miller's Algorithm

Here the line l_1 passes two points T and P if $T \neq P$, and is the tangent at point T if $T = P$. The line l_2 is the vertical line to the x-axis through $T + P$. We call the procedures in Step 3 and Step 4 as TDBL, and in Step 5 and Step 6 as TADD. A computation of TADD/TDBL is divided into three stages, the computation of ECADD/ECDBL and coefficients for l_1, l_2, the evaluation of $l_1(Q+S), l_1(S), l_2(Q+S), l_2(S)$, and the update of f.

3.2 Straightforward Implementation

Let us estimate the computing time of the Tate pairing when points are represented by the Jacobian coordinate $\mathcal{J}$. Suppose Z-coordinates of the points P are chosen to 1 (this is done by converting to the affine coordinate). We set $T = (X_1, Y_1, Z_1)$ and $P = (X_2, Y_2, 1)$. We also set $S = (\mathbf{x}_S, \mathbf{y}_S, 1)$, $Q + S = (\mathbf{x}_{Q+S}, \mathbf{y}_{Q+S}, 1)$ as the affine points over $E(\mathbb{F}_{q^k})$.

An element of $\mathbb{F}_{q^k}$ is represented as a bold character in the following. We denote a computing time of a multiplication and a squaring in $\mathbb{F}_{q^k}$ as M_k and S_k, respectively. As the extension field $\mathbb{F}_{q^k}$ is represented as a k-dimensional vector space over $\mathbb{F}_q$, a naive implementation provides $M_k = \mathcal{O}(k^2 M)$, where M is the computation time of a multiplication of $\mathbb{F}_q$. The multiplication of elements between $\mathbb{F}_{q^k}$ and $\mathbb{F}_q$ requires kM.

Computation of TADD: The lines l_1^{add} and l_2^{add} for $P + T$ ($T \neq \pm P$) are given by

$$\begin{aligned} l_1^{\text{add}}(\mathbf{x}, \mathbf{y}) &= Z_3(\mathbf{y} - Y_2) - (Y_2 Z_1^3 - Y_1)(\mathbf{x} - X_2), \\ l_2^{\text{add}}(\mathbf{x}, \mathbf{y}) &= Z_3^2 \mathbf{x} - X_3, \end{aligned}$$

where $(X_3, Y_3, Z_3) = P + T$. A computation of $P + T$ requires $8M + 3S$, and during the computation, the coefficient $R = Y_2 Z_1^3 - Y_1$ has been computed. Thus we only need to compute $Z_3 Y_2, R X_2, Z_3^2$, which requires $2M + 1S$. For the evaluation of $l_1^{\text{add}}(Q)$ and $l_2^{\text{add}}(Q)$ for $Q = (\mathbf{x}_Q, \mathbf{y}_Q, 1) \in E(\mathbb{F}_{q^k})$, we require $3k$ multiplications in $\mathbb{F}_q$. Then, at last, we update $f = \mathbf{f}$ by

$$\begin{aligned} \mathbf{a} &= \mathbf{a} \times l_1^{\text{add}}(Q+S) \times l_2^{\text{add}}(S), \\ \mathbf{b} &= \mathbf{b} \times l_1^{\text{add}}(S) \times l_2^{\text{add}}(Q+S), \end{aligned}$$

where $\mathbf{f} = \mathbf{a}/\mathbf{b}$ is the quotient of two values $\mathbf{a}, \mathbf{b}$ of $\mathbb{F}_{q^k}$. It requires 4 multiplications in $\mathbb{F}_{q^k}$. Thus a TADD requires $\texttt{TADD} = (8M + 3S) + (2M + 1S) + 2(3kM) + 4M_k = 4M_k + (6k + 10)M + 4S$.

Computation of TDBL: Similarly, the lines for $T + T$ are given by

$$l_1^{\mathrm{dbl}}(\mathbf{x}, \mathbf{y}) = (Z_4 Z_1^2 \mathbf{y} - 2Y_1^2) - (3X_1^2 + aZ_1^4)(Z_1^2 \mathbf{x} - X_1)$$
$$l_2^{\mathrm{dbl}}(\mathbf{x}, \mathbf{y}) = Z_4^2 \mathbf{x} - X_4,$$

where $(X_4, Y_4, Z_4) = T + T$. A computation of $T + T$ requires $4M + 6S$, and computation of coefficients requires $3M + 1S$. For an evaluation, we require $3k$ multiplications in $\mathbb{F}_q$. An update is computed by

$$\mathbf{a} = \mathbf{a}^2 \times l_1^{\mathrm{dbl}}(Q + S) \times l_2^{\mathrm{dbl}}(S),$$
$$\mathbf{b} = \mathbf{b}^2 \times l_1^{\mathrm{dbl}}(S) \times l_2^{\mathrm{dbl}}(Q + S),$$

which requires 4 multiplications and 2 squiring in $\mathbb{F}_{q^k}$. Thus a TDBL requires $\texttt{TDBL} = (4M + 6S) + (3M + 1S) + 2(3kM) + 4M_k + 2S_k = 4M_k + 2S_k + (6k + 7)M + 7S$.

4 Improvements

In this section we describe how to improve the computation time of the Tate pairing $\langle P, Q \rangle = (f_P(Q + S)/f_P(S))^{(q^k-1)/\ell}$. In the Miller's algorithm, we need three stages to update f; (1) computation of ECDBL(ECADD) and coefficients of l_1, l_2, (2) evaluation of $l_1(Q+S), l_1(S), l_2(Q+S), l_2(S)$, and (3) update of f. All computation in (1) are in $\mathbb{F}_q$, while in (2),(3) are in $\mathbb{F}_{q^k}$. We investigate complete formulas of TADD and TDBL assembled by arithmetics of the definition field $\mathbb{F}_q$ and its extension field $\mathbb{F}_{q^k}$.

4.1 Coordinate System

In the computations of ECADD, ECDBL and coefficients of the lines l_1, l_2, we need many squaring of Z-coordinates. This implies that a new coordinate (X, Y, Z, Z^2) matches the computation. We call this representation by the *simplified Chudonovsky-Jacobian coordinate* $\mathcal{J}^s$, in which (X, Y, Z) represents a point in the Jacobian coordinate.

In ECADD, our coordinate $\mathcal{J}^s$ requires $8M + 3S$ as same as the Jacobian coordinate $\mathcal{J}$ and the Chudonovsky-Jacobian coordinate $\mathcal{J}^C$. However, coefficients of l_1, l_2 are computed in $2M$ because we have Z_3^2 in the corrdinate. Thus we require $(8M + 3S) + 2M = 10M + 3S$ for ECADD and coefficient computations. Similarily, we require $(4M + 6S) + 3M = 7M + 6S$ for ECDBL and coefficient computations with our coordinate. A comparison is in Table 4.

	ECADD	Coeff.	Total	ECDBL	Coeff.	Total
$\mathcal{J}$	$8M+3S$	$2M+1S$	$10M+4S$	$4M+6S$	$3M+1S$	$7M+7S$
$\mathcal{J}^C$	$8M+3S$	$2M$	$10M+3S$	$5M+6S$	$3M$	$8M+6S$
$\mathcal{J}^s$	$8M+3S$	$2M$	$10M+3S$	$4M+6S$	$3M$	$7M+6S$

Table 4. Comparison of computing times of ECADD(ECDBL) and coefficients

4.2 Direct Computation of $l_1(Q+S) \times l_2(S)$

In TADD, lines $l_1^{\text{add}}, l_2^{\text{add}}$ are given by

$$\begin{aligned} l_1^{\text{add}}(\mathbf{x},\mathbf{y}) &= Z_3(\mathbf{y}-Y_2) - R(\mathbf{x}-X_2) = a\mathbf{x}+b\mathbf{y}+c, \\ l_2^{\text{add}}(\mathbf{x},\mathbf{y}) &= Z_3^2\mathbf{x} - X_3 = d\mathbf{x}+e, \end{aligned}$$

where we requires $2M+1S$ for coefficients computation as before. Here we have

$$l_1^{\text{add}}(Q+S)\times l_2^{\text{add}}(S) = ad(\mathbf{x}_{Q+S}\mathbf{x}_S)+bd(\mathbf{y}_{Q+S}\mathbf{x}_S)+ae\mathbf{x}_{Q+S}+be\mathbf{y}_{Q+S}+cd\mathbf{x}_S+ce,$$

and we need $6M$ for coefficients and $5kM$ for evaluation if $\mathbf{x}_{Q+S}\mathbf{x}_S$ and $\mathbf{y}_{Q+S}\mathbf{x}_S$ are pre-computed. For $l_1^{\text{add}}(S) \times l_1^{\text{add}}(Q+S)$, we compute

$$l_1^{\text{add}}(S)\times l_2^{\text{add}}(Q+S) = ad(\mathbf{x}_S\mathbf{x}_{Q+S})+bd(\mathbf{y}_S\mathbf{x}_{Q+S})+ae\mathbf{x}_S+be\mathbf{y}_S+cd\mathbf{x}_{Q+S}+ce,$$

which requires more $4kM$, because we have already computed all coefficients and $\mathbf{x}_{Q+S}\mathbf{x}_S$. After computing $l_1^{\text{add}}(Q+S)\times l_2^{\text{add}}(S)$ and $l_1^{\text{add}}(S)\times l_2^{\text{add}}(Q+S)$, the update requires only $2M_k$. Thus we need $\texttt{TADD}^s = (8M+2S)+(2M+1S)+6M+5kM+4kM+2M_k = 2M_k+(9k+16)M+3S$.

Similar avoidance can be possible for TDBL, which requires $\texttt{TDBL}^s = (4M+5S)+(3M+1S)+6M+5kM+4kM+2M_k+2S_k = 2M_k+2S_k+(9k+13)M+6S$. We summarize these results in the following table.

	TADD	TDBL
Straightforward Method	$4M_k+(6k+10)M+4S$	$4M_k+2S_k+(6k+7)M+7S$
Improved Method	$2M_k+(9k+16)M+3S$	$2M_k+2S_k+(9k+13)M+6S$

Table 5. Computing times of a TADD and a TDBL

4.3 Iterated TDBL

For a point $P \in E(\mathbb{F}_q)$, computing 2^wP is called the *w-iterated ECDBL*. A w-iterated ECDBL can be computed by applying ECDBL w times successively, but it may be faster by sharing intermediate results if we call it as one function. Indeed, Itoh et al. proposed a fast algorithm (Table 6) for a w-iterated ECDBL in the Jacobian coordinate [ITTTK99], which requires $4wM+(4w+2)S$.

This idea can be easily applied to our situation. We show a fast algorithm for the *w-iterated TDBL* in the following. We represent $\mathbf{f}=\mathbf{a}/\mathbf{b}$ as the quotient of two elements $\mathbf{a},\mathbf{b} \in \mathbb{F}_{q^k}$. Suppose $2^iP = (X_i, Y_i, Z_i)$ are computed by the

```
iECDBL $(4wM + (4w+2)S)$
Input: $P_0 = (X_0, Y_0, Z_0),\ w$
Output: $P_w = 2^w P_1 = (X_w, Y_w, Z_w)$
$W_0 \leftarrow aZ_0^4$
$M_0 \leftarrow 3X_0^2 + W_0$
$S_0 \leftarrow 4X_0Y_0^2$
$X_1 \leftarrow M_0^2 - 2S_0$
$Y_1 \leftarrow M_0(S_0 - X_1) - 8Y_0^4$
$Z_1 \leftarrow 2Y_0Z_0$
for($i = 1$ to $w-1$){
  $W_i \leftarrow 2(8Y_{i-1}^4)W_{i-1}$
  $M_i \leftarrow 3X_i^2 + W_i$
  $S_i \leftarrow 4X_iY_i^2$
  $X_{i+1} \leftarrow M_i^2 - 2S_i$
  $Y_{i+1} \leftarrow M_i(S_i - X_{i+1}) - 8Y_i^4$
  $Z_{i+1} \leftarrow 2Y_iZ_i$
}
```

Table 6. Iterated ECDBL in the Jacobian coordinate

iterated ECDBL for $i \geq 1$, The lines $l_1^{(i)}$ and $l_2^{(i)}$ are recursively defined by the equations:

$$l_1^{(i)}(\mathbf{x},\mathbf{y}) = (Z_iZ_{i-1}^2\mathbf{y} - 2Y_{i-1}^2) - M_{i-1}(Z_{i-1}^2\mathbf{x} - X_{i-1}),$$
$$l_2^{(i)}(\mathbf{x},\mathbf{y}) = Z_i^2\mathbf{x} - X_i,$$

where $Z_i = 2Y_{i-1}Z_{i-1}, M_{i-1} = 3X_{i-1}^2 + aZ_{i-1}^4$. Here we require $3M + 1S$ for coefficients $Z_iZ_{i-1}^2,\ M_{i-1}Z_{i-1}^2, M_{i-1}X_{i-1}, Z_i^2$, because we have computed Z_{i-1}^2. The update of the iterated TDBL is similarly computed by the direct computation technique. Thus we have

$$\texttt{iTDBL}^s(w) = 2wM_k + 2wS_k + (9k+13)wM + (5w+1)S$$

4.4 Combining with Previous Methods

In this section, we discuss the combination of our techniques with previous two techniques: the first is to use the random element $S \in E(\mathbb{F}_q)$ ([GHS02]) and the other one is to use the condition $\ell \not| (q-1)$ ([BKLS02]). They aim at enhancing the computation time of the Tate pairing for supersingular curves. However, these choices of the parameters can improve the efficiency of computing the Tate pairing for genera curves. We estimate how the choices can make the Tate pairing faster combining with the methods from the previous sections.

$S \in E(\mathbb{F}_q)$: We can choose $S \in E(F_q)$ instead of $S \in E(F_{q^k})$ [GHS02]. Let D_Q be the divisor from the class $(Q) - (\mathcal{O})$. Then $D_Q \sim (Q+S) - (S)$ for any $S \in E(\mathbb{F}_{q^k})$, and we can choose $S \in E(\mathbb{F}_q)$. A problem might occur during the computation of the Tate pairing. The calculation of TDBL and TADD for points T, P should be equal to neither of $\pm S, \pm(S+Q)$. If $S \in E(\mathbb{F}_q)$ is randomly chosen, the error probability is negligible, because we compute ℓP using the addition chain. The number of the intermediate points arisen from the addition chain of ℓ are bounded in the polynomial order of $\log \ell$, and the possible choice of S is in the exponential order of $q > \ell$. We denote by $\texttt{TADD}_{S \in E(\mathbb{F}_q)}$ and $\texttt{TDBL}_{S \in E(\mathbb{F}_q)}$ the computation time of $\texttt{TADD}$ and $\texttt{TDBL}$ for $S \in E(\mathbb{F}_q)$, respectively.

We first consider $\texttt{TADD}_{S \in E(\mathbb{F}_q)}$. If we choose $S \in E(\mathbb{F}_q)$, the values of $l_1(S), l_2(S)$ are in subfield $\mathbb{F}_q$.

$$l_1^{\mathrm{add}}(S) = Z_3(y_S - Y_2) - R(x_S - X_2),$$
$$l_2^{\mathrm{add}}(S) = Z_3^2 x_S - X_3,$$

where $R = Y_1 - Y_2 Z_1^3$. The coefficient computation requires $2M + 1S$ and the evaluation requires only $3M$. We estimated that the evaluation of $l_1(Q+S), l_2(Q+S)$ for $Q + S \in E(\mathbb{F}_{q^k})$ requires $3kM$.

$$\mathbf{a} = \mathbf{a} \times l_1^{\mathrm{add}}(Q+S) \times l_2^{\mathrm{add}}(S)$$
$$\mathbf{b} = \mathbf{b} \times l_1^{\mathrm{add}}(S) \times l_2^{\mathrm{add}}(Q+S),$$

The updating of $\mathbf{a}, \mathbf{b}$ requires $2M_k + 2kM$ due to $l_2^{\mathrm{add}}(S), l_1^{\mathrm{add}}(S) \in \mathbb{F}_q$. In total we need $2M_k + (5k+13)M + 3S$ for computing $\texttt{TADD}_{S \in E(\mathbb{F}_q)}$. Similarly computing a TDBL requires $\texttt{TDBL}_{S \in E(\mathbb{F}_q)} = 2M_k + 2S_k + (5k+10)M + 6S$, and w-iterated TDBL requires $2wM_k + 2wS_k + (5k+10)wM + (5w+1)S$.

We summarize the results in the following table.

拡大していない場合の計算量

	Computing times ($S \in E(F_q)$)
TADD	$2M_k + (5k+13)M + 3S$
TDBL	$2M_k + 2S_k + (5k+10)M + 6S$
w-iterated TDBL	$2wM_k + 2wS_k + (5k+10)wM + (5w+1)S$

Table 7. Computing times of TADD/TDBL/iterated TDBL ($S \in E(\mathbb{F}_q)$)

$q - 1$ is not divisible by ℓ: The prime q must satisfies $\ell | (q^k - 1)$, where ℓ is the divisor of $\#E(\mathbb{F}_q)$. The Tate pairing computes $\alpha^{(q^k-1)/\ell}$ in the final step, where $\alpha \in \mathbb{F}_{q^k}$. If we choose ℓ with $\ell \not| (q-1)$, then we have $a^{(q^k-1)/\ell} = 1$ for $a \in \mathbb{F}_q$. This observation was pointed out for supersingular curves in the paper [BKLS02].

The condition of $\ell \not| (q-1)$ can be checked in the parameter generation stage. When we combine this condition with $S \in E(\mathbb{F}_q)$, the computation of the Tate pairing can be speeded up. We denote by $\texttt{TADD}_{S \in E(\mathbb{F}_q)}^{\ell \not| (q-1)}$ and $\texttt{TDBL}_{S \in E(\mathbb{F}_q)}^{\ell \not| (q-1)}$ the computation time of $\texttt{TADD}$ and $\texttt{TDBL}$ for $S \in E(\mathbb{F}_q)$ with condition $\ell \not| (q-1)$, respectively.

We first consider $\texttt{TADD}_{S \in E(\mathbb{F}_q)}^{\ell \not| (q-1)}$. We assume $S \in E(\mathbb{F}_q)$. If we choose $S \in E(\mathbb{F}_q)$, the values of $l_1(S), l_2(S)$ are in subfield $\mathbb{F}_q$. Thus these values can be discarded of the evaluation. We can update $\mathbf{f} = \mathbf{a}/\mathbf{b}$ as follows:

$$\mathbf{a} = \mathbf{a} \times l_1^{\text{add}}(Q+S)$$
$$\mathbf{b} = \mathbf{b} \times l_2^{\text{add}}(Q+S).$$

The evaluation of $l_1(Q+S), l_2(Q+S)$ for $Q+S \in E(\mathbb{F}_{q^k})$ requires $(3k+2)M+1S$. The updating of $\mathbf{a}, \mathbf{b}$ require $2M_k$. Consequently we need $2M_k+(3k+10)M+3S$ for computing a TADD under assumptions $S \in E(\mathbb{F}_q)$ and $\ell \not| (q-1)$. Similarly, computing a TDBL requires $\texttt{TDBL}_{S\in E(\mathbb{F}_q)}^{\ell \not| (q-1)} = 2M_k + 2S_k + (3k+7)M + 6S$, and a w-iterated TDBL requires $2wM_k + 2wS_k + (3k+7)wM + (5w+1)S$.

	Computing times ($S \in E(\mathbb{F}_q)$ and $\ell \not| (q-1)$)
TADD	$2M_k + (3k+10)M + 3S$
TDBL	$2M_k + 2S_k + (3k+7)M + 6S$
w-iterated TDBL	$2wM_k + 2wS_k + (3k+7)wM + (5w+1)S$

Table 8. Computing times of TADD/TDBL/iterated TDBL ($S \in E(\mathbb{F}_q), \ell \not| (q-1)$)

4.5 Application to Supersingular Curves

distortion map の計算量がある？？

In this section, we discuss the improvements for supersingular elliptic curves. We combine the methods proposed in reference [BKLS02] with our methods in the previous section. According to [Men93], the trace of the supersingular curve over $\mathbb{F}_p$ ($p > 3$) equals to 0, that is we have the extension degree $k = 2$.

In the following we consider a supersingular curve defined by the equation $y^2 = x^3 + ax$ over $\mathbb{F}_p$ ($p \equiv 3 \pmod 4$), which has a distortion map $\Phi : (x, y) \to (-x, iy) \in \mathbb{F}_{p^2}$, where $i^2 = -1$. In this case, the computation time of a multiplication M_2 and a squaring S_2 in the extension field $\mathbb{F}_{p^2}$ can be achieved $M_2 = 4M$ and $S_2 = 2M$, where M and S are the computation time of a multiplication and a squaring in the prime field $\mathbb{F}_q$, respectively.

When a point Q is computed using the distortion map [BKLS02,Jou02], we can make the computation of the Tate pairing much faster. One reason is that we can choose $Q = (x, y)$, where one of x, y is the element of subgroup $\mathbb{F}_q$. The other reason is that we do not have to generate a point $Q \in E(\mathbb{F}_{q^2})$ — the point Q is easily converted from a point in $E(\mathbb{F}_q)$.

We estimate the running time of TADD, TDBL and w-iTDBL under the assumption $S \in E(\mathbb{F}_p)$. If we use the distortion map, condition $\ell \not| (p-1)$ is automatically satisfied. We denote by $\texttt{TADD}_{S\in E(\mathbb{F}_p)}^{\Phi}$, $\texttt{TDBL}_{S\in E(\mathbb{F}_p)}^{\Phi}$, and w-$\texttt{iTDBL}_{S\in E(\mathbb{F}_p)}^{\Phi}$, the computation time of $\texttt{TADD}$, $\texttt{TDBL}$, and w-$\texttt{iTDBL}$ for $S \in E(\mathbb{F}_p)$ with torsion map Φ, respectively.

Because the x-coordinate of $\Phi(Q+S)$ is an element of $\mathbb{F}_p$, $l_2^{\text{add}}(Q+S) \in \mathbb{F}_p$ holds. Thus we do not have to compute $l_2^{\text{add}}(Q+S)$ due to $\ell \not| (p-1)$. We can update $\mathbf{f} = \mathbf{a}/\mathbf{b}$ as follows:

$$\mathbf{a} = \mathbf{a} \times l_1^{\text{add}}(Q+S).$$

Here we have a representation $l_1^{\text{add}}(Q+S) = g\mathbf{y} + h$ for some $g, h \in \mathbb{F}_p$. The evaluation of $l_1^{\text{add}}(Q+S)$ requires $4M$. The updating of $\mathbf{a}$ requires $1M_2 = 4M$. In total we need $1M_2 + (2k+10)M + 3S = 18M + 3S$ for computing $\mathtt{TADD}^{\Phi}_{S \in E(\mathbb{F}_p)}$. Similarly, computing $\mathtt{TDBL}^{\Phi}_{S \in E(\mathbb{F}_p)}$ requires $16M + 6S$, and $w\text{-}\mathtt{iTDBL}^{\Phi}_{S \in E(\mathbb{F}_p)}$ requires $16wM + (5w+1)S$, respectively.

If we implement the Tate pairing over supersingular curve $y^2 = x^3 + ax$.

	Computing times ($S \in E(\mathbb{F}_p)$ and Φ)
TADD	$18M + 3S$
TDBL	$16M + 6S$
iterated TDBL	$16wM + (5w+1)S$

Table 9. Computing times of TADD/TDBL/iterated TDBL for $y^2 = x^3 + ax$.

5 Comparison

In this section, we compare the computing times of the Tate pairing. In order to assure the security of the pairing based primitives, $|\ell| \geq 160$ and $|q^k| \geq 1024$ [GHS02], where $|x|$ denotes the bit length of x. So we used 5 pairs of parameters $(k, |q|) = (2,512), (3,342), (4,256), (5,205), (6,171)$. For each parameter, we randomly generate 1000 ℓs and compute the Tate pairing with the NAF representation [IEEE]. Algorithms are as follows: (0) Straight-forward implementation, (1) Direct computation in $\mathcal{J}^s$, (1i) Direct computation in $\mathcal{J}^s$ with iterated TDBL, (2) Direct computation in $\mathcal{J}^s$ with $S \in E(\mathbb{F}_q)$, (2i) Direct computation in $\mathcal{J}^s$ with $S \in E(\mathbb{F}_q)$ with iterated TDBL, (3) Direct computation in $\mathcal{J}^s$ with $S \in E(\mathbb{F}_q)$ and $\ell \not| (q-1)$, (3i) Direct computation in $\mathcal{J}^s$ with $S \in E(\mathbb{F}_q)$ and $\ell \not| (q-1)$ with iterated TDBL. Timing data are summarized in Table 10. We assume that $M_k = k^2 M, S_k = k^2 S, 1S = 0.8M$ in the table, here M denotes the computing time of a multiplication in $\mathbb{F}_q$ (So it is of no use to compare two computing times in different column).

	Computing times of the Tate pairing (Estimation)				
	$k=2, \|q\|=512$	$k=3, \|q\|=342$	$k=4, \|q\|=256$	$k=5, \|q\|=205$	$k=6, \|q\|=171$
(0)	31122.1M	35413.0M	41002.6M	47290.9M	53905.7M
(1)	33308.4M	33677.9M	35944.5M	39135.9M	42761.1M
(1i)	33035.4M	33495.7M	35808.1M	39026.5M	42670.0M
(2)	25793.0M	26828.8M	29451.0M	32841.7M	36595.0M
(2i)	25520.1M	26646.6M	29314.6M	32732.4M	36503.9M
(3)	21010.5M	22719.3M	25691.7M	29284.2M	33169.4M
(3i)	20737.6M	22537.1M	25555.2M	29174.8M	33078.3M
(4)	14143.2M	———	———	———	———
(4i)	13870.3M	———	———	———	———

Table 10. Comparison of computing times

As k become larger, the direct computation become more efficient. If $k = 6$, the direct computation (1) is about 20.7% faster than the straight-forward implementation (0), and the direct computation with other techniques (3i) is about 38.6% faster. When $k = 2$, the direct computation looks inefficient, namely it makes the computation slower. Still our coordinate $\mathcal{J}^s$ and the iterated TDBL work well. Indeed, if we use $\mathcal{J}^s$ and the iterated TDBL (but not the direct computation), the estimation is $30292.6M$ for $k = 2$ which is about 2.6% faster than (0).

We also give an estimated computation time (4) for a supersingular curve $y^2 = x^3 + ax$ discussed in section 5.1. In this case, the distortion map works very significantly and the computing time is very short. Still our iterated TDBL makes it about 2.0% faster.

6 Concluding Remarks

We proposed several improvements of computing the Tate pairing of the elliptic curves over finite fields $\mathbb{F}_q$ with $(q = p^m, p > 3)$. The proposed algorithms can be applicable not only the supersingular curves but also general elliptic curves. The proposed methods are specially effective upon the elliptic curves that has large MOV degree $(k > 2)$. For $k = 6$, the proposed scheme is about 20% faster than the previously fastest algorithm.

Acknowledge

The authors would like to thank anonymous refrees for their helpful comments and suggestions.

References

[BF01] D.Boneh, and M.Franklin, "Identity-based encryption from the Weil pairing", *CRYPTO 2001*, LNCS 2139, pp.213-229, Springer-Verlag, 2001.

[BKLS02] P.Barreto, H.Kim, B.Lynn, and M.Scott, "Efficient Algorithms for Pairing-Based Cryptosystems", *CRYPTO 2002*, LNCS 2442, pp.354-368, Springer-Verlag, 2002.

[BLS01] D.Boneh, B.Lynn, and H.Shacham, "Short Signatures from the Weil Pairing", *ASIACRYPT 2001*, LNCS 2248, pp.514-532, Springer-Verlag, 2001.

[Cop83] D.Coppersmith, "Evaluating Logarithms in $GF(2^n)$", *STOC 1984*, pp.201-207, 1983.

[CMO98] H.Cohen, A.Miyaji and T.Ono, "Efficient elliptic curve exponentiation using mixed coordinates", *Asiacrypt'98*, LNCS 1514, pp.51-65, Springer-Verlag, 1998.

[DEM02] R.Dunport, A.Enge, and F.Morain, "Building curves with arbitrary small MOV degree over finite prime fields", Cryptology ePrint Archive, Report 2002/094, 2002.

[FMR99] G.Frey, M.Müller, and H.Rück, "The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems", *IEEE Trans. on Information Theory*, vol.45, pp.1717-1718, 1999.

[Gal01] S.D.Galbraith, "Supersingular Curves in Cryptography", *Asiacrypt 2001*, LNCS 2248, pp.495-513, Springer-Verlag, 2001.

[GHS02] S.D.Galbraith, K.Harrison, and D.Soldera, "Implementing the Tate pairing", *ANTS V*, LNCS 2369, pp.324-337, Springer-Verlag, 2002.

[Hes02] F.Hess, "Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings", Cryptology ePrint Archive, Report 2002/012, 2002.

[IEEE] IEEE P1363, Standard Specifications for Public-Key Cryptography, 2000.

[ITTTK99] K.Itoh, M.Takenaka, N.Torii, S.Temma, and Y.Kurihara, "Fast Implementation of Public-Key Cryptography on DSP TMS320C6201", *CHES'99*, LNCS 1717, pp.61-72, 1999.

[Jou00] A.Joux, "A One Round Protocol for Tripartite Diffie-Hellman", *ANTS IV*, LNCS 1838, pp.385-393, Springer-Verlag, 2000.

[Jou02] A.Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems (survey)", *ANTS V*, LNCS 2369, pp.20-32, Springer-Verlag, 2002.

[Kob87] N.Koblitz, "Elliptic curve cryptosystems", *Math. of Comp.*, vol.48, pp.203-209, 1987.

[Men93] A.Menezes, "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993.

[Mil86] V.Miller, "Use of elliptic curves in cryptography", *CRYPTO'85*, LNCS 218. p.417-426, Springer-Verlag, 1986.

[MNT01] A.Miyaji, M.Nakabayashi, and S.Takano, "New explicit conditions of elliptic curve traces for FR-reduction", *IEICE Trans. Fundamentals*, E84-A(5), May, 2001.

[MOV93] A.Menezes, T.Okamoto, and S.Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field", *IEEE Trans. on Information Theory*, vol.39, pp.1639-1646, 1993.

[NIST] National Institute of Standards and Technology, Recommended Elliptic Curves for Federal Government Use, in the appendix of FIPS 186-2.

[OP01] T.Okamoto, P.Pointcheval, "The Gap Problems: a new class of problems for the security of cryptographic primitives", *PKC 2001*, LNCS 1992, pp.104-118, Springer-Verlag, 2001.

[Pat02] K.G.Paterson, "ID-based Signatures from Pairings on Elliptic Curves", Cryptology ePrint Archive, Report 2002/004, 2002.

[PS02] D.Page, and N.Smart, "Hardware Implementation of Finite Fields of Characteristic Three", to appear in the proceedings of *CHES 2002*.

[Sma01] N.P.Smart, "An Identity Based Authentificated Key Agreement Protocol Based on the Weil Pairing", Cryptology ePrint Archive, Report 2001/111, 2001.

[SEC] Standards for Efficient Cryptography Group (SECG), Specification of Standards for Efficient Cryptography. `http://www.secg.org`

[SOK00] R.Sakai, K.Ohgishi, and M.Kasahara, "Cryptosystems Based on Pairing", *2000 Symposium on Cryptography and Information Security (SCIS 2000)*, Okinawa, Japan, Jan. 26-28, 2000.

[SW02] N.P.Smart, and J.Westwood, "Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three", Cryptology ePrint Archive, Report 2002/114, 2002.

[Ver01] E.R.Verheul, "Self-Blindable Credential Certificates from the Weil pairing", *ASIACRYPT 2001*, LNCS 2248, pp.533-551, Springer-Verlag, 2001.