Efficient Tate Pairing Computation Using Double-Base Chains

Chang'an Zhao, Fangguo Zhang and Jiwu Huang

[1] Department of Electronics and Communication Engineering,
Sun Yat-Sen University, Guangzhou 510275, P.R.China
[2] Guangdong Key Laboratory of Information Security Technology,
Guangzhou 510275, P.R.China
chinazhaok@yahoo.com.cn
isszhfg@mail.sysu.edu.cn
isshjw@mail.sysu.edu.cn

Abstract. Pairing-based cryptosystems have been developing very fast in the last few years. The efficiencies of the cryptosystems are determined by the computation of the Tate pairing. In this paper a new efficient algorithm based on double-base chain for computing the Tate pairing is proposed for odd characteristic $p > 3$. The inherent sparseness of double-base number system reduces the computational cost for computing the Tate pairing evidently. It is 9% faster than the previous fastest method for MOV degree k=6.

Keywords: Tate pairing, Elliptic curves, Double-base chains.

1 Introduction

The bilinear pairings, namely the Weil pairing and the Tate pairing of algebraic curves, are important tools for research on algebraic geometry. The early applications of the bilinear pairings in cryptography were used to evaluate the discrete logarithm problem. For example, the MOV attack [22] (using Weil pairing) and FR attack [15] (using Tate pairing) reduce the discrete logarithm problem on some elliptic curves or hyperelliptic curves to the discrete logarithm problem in a finite field. However, the bilinear pairings have been found various applications in cryptography since the publication of the works of Joux [20] and Sakai [26]. For example, there have been many papers on identity based encryption [9, 28, 6], short signatures [10, 29, 7], group signatures [8], and many more. Pairing-based cryptosystems are currently one of the most active areas of research in elliptic curve cryptography. More details about research on pairings can be found at Paulo Barreto's pairing-based crypto lounge [3].

However, the pairing computations are often the bottleneck to realize cryptographic applications practically. Therefore, the fast implementations of these pairings have become a subject of active research areas in elliptic curve cryptography. As the Tate pairing is about twice faster than the Weil pairing generally, here we discuss the Tate pairing mainly.

The computation of the Tate pairing can be performed using an algorithm first presented by Miller [23]. There are many improvements based in some manner on Miller's algorithm proposed. Generally the following factors would affect the efficiency of computing the pairings: the base in Miller's algorithm, the coefficients of the elliptic curves, the Hamming weight of the subgroup orders and the relation of two input points. From different angles, many techniques have been exploited to dramatically improve the performance of the algorithm. The work in [2, 17, 13] mainly focus on reducing the large unit operations such as to remove the cost of denominator computation and denominator squaring. Eisenträer, et al. [14] combine an improved point multiplication with a parabola substitution, enables them to get an improvement to Miller's algorithm for general elliptic curves. The authors in [5] find that the times of Tate double-and-add operation in the algorithm could be reduced. Izu and Takagi [19] apply the iterated double Tate pairing and the polynomial expansion trick to the computation of Tate pairing. Recently, Kobayashi et al. introduce a pseudo-arithmetic method to computing the Tate pairing in [21]. Some special pairings for efficient fast computations are introduced, such as eta pairing [4] and ate pairing [18].

In our paper, we propose a new modified Miller's algorithm based on a representation of the multiplier as a sum of mixed powers of 2 and 3, called the double-base number system(DBNS). The sparseness of this representation leads to fewer multiplicative operations than some general modifications of Miller's algorithm. The new algorithm for computing the pairing can be applied into supersingular and ordinary elliptic curves. Here the prime order of the elliptic curve subgroup is selected randomly since a prime modulus p of low Hamming weight leads to the possible lowering of discrete-logarithm security [25], while P and Q in the pairing $e_l(P, Q)$ are chosen randomly and have no special relations in our algorithm since they can be implemented widely in many cryptographic protocols for security considerations.

This paper is organized as follows. Section 2 introduces basic mathematical concepts of the Tate pairing, Miller's algorithm and DBNS. Section 3 describes the modified Miller's algorithm based on double-base chain. Section 4 discusses the efficiency of the new algorithm. In Section 5 we summarize our work.

2 Mathematical Preliminaries

2.1 The Tate Pairing

Let $q = p^m$, where $p > 3$ is a prime number and m is a positive integer. F_q is a finite field with q elements. p is the characteristic of F_q, and m is its extension degree.

Let E be an elliptic curve defined over a finite field F_q.

$$E : y^2 = x^3 + ax + b$$

where a, $b \in F_q$, $4a^3 + 27b^2 \neq 0 \in F_q$.

Let l be a prime integer which is coprime to q. l is the order of a point $P \in E(F_q)$. Let $l \dagger (q-1)$, and k be the smallest positive integer satisfying $l|(q^k-1)$. k is also called the embedding degree or MOV degree. We denote the l-torsion points on the elliptic curve as E[l].

A divisor is a formal sum of points on the elliptic curve $E(F_{q^k})$(see [27]). The degree of a divisor $D = \sum_P a_P(P)$ is the sum $deg(D) = \sum_P a_P$. Let f be a function on the elliptic curve and let $D_R = \sum_R a_R(R)$ which satisfies $deg(D_R) = 0$. Then we define $f(D_R) = \prod_R f(R)^{a_R}$.

Let $P \in E(F_q)[l]$ and $Q \in E(F_{q^k})[l]$. D_P and D_Q are the divisors of E, satisfying $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$, respectively. Let f_P be a function on the elliptic curve whose divisor is $div(f_P) = lD_P$.

The reduced Tate pairing e_l is a mapping

$$e_l : E(F_q) \cap E[l] \times E(F_{q^k}) \cap E[l] \to F^*_{q^k}$$

defined as

$$e_l(P, Q) = (f_P(D_Q))^{(q^k-1)/l}.$$

By Theorem 1 in [2], one can define the reduced Tate pairing as

$$e_l(P, Q) = f_P(Q)^{(q^k-1)/l}.$$

The Tate pairing satisfies the following properties:

1. ***Well-defined*** $e_l(\mathcal{O}, Q) = 1$ for all $Q \in E(F_{q^k})$ and $e_l(P, Q) = 1$ for all $P \in E(F_q) \cap E[l]$, $Q \in lE(F_{q^k})$.
2. ***Bilinearity*** For any $P \in E(F_q)[l]$, $Q \in E(F_{q^k})$ and any integer n, $e_l(nP, Q) = e_l(P, nQ) = e_l(P, Q)^n$.
3. ***Non-degeneracy*** If $e_l(P, Q) = 1$ for all $Q \in E(F_{q^k})$, then $P = \mathcal{O}$.

2.2 Miller's Algorithm

The main part of the computation of the Tate pairing is calculating $f_P(Q)$. Miller [23] first proposed an efficient algorithm for computing the Tate pairing. The main idea of Miller's algorithm is to combine the 'double-add' algorithm for elliptic curve point multiplication with an evaluation of the straight lines passing the related points used in the addition process. Now we describe Miller's original algorithm simply.

For $P, Q \in E(F_{q^k})$, we define $l_{P,Q}$ to be the equation of the line through points P and Q (if $P = Q$, then $l_{P,Q}$ is the tangent to the curve at P or Q, and if one of P or Q is the point at infinity $\mathcal{O}$, then the $l_{P,Q}$ is a vertical line at the other point). We define v_P to be the equation of the line between P and $\mathcal{O}$ if P is not equal to $\mathcal{O}$.

Let P be a point in $E(F_q)[l]$ and f_j be a function on the elliptic curve with its divisor $(f_j) = j(P) - (jP) - (j-1)\mathcal{O}$, $j \in Z$. Then for $i, j \in Z$, we have

$$f_{i+j}(Q) = f_i(Q)f_j(Q)l_{iP,jP}(Q)/v_{(i+j)P}(Q).$$

Note that $(f_0) = (f_1) = 0$, so that $f_0(Q) = f_1(Q) = 1$. A second useful remark is that $f_i = -(f_{-i}) - (v_{iP})$ for $i < 0$. Thus we have $f_i(Q) = 1/(f_{-i}(Q)v_{iP}(Q))$. Particularly, we have $f_{-1}(Q) = 1/(x_Q - x_P)$ for $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.

Miller's original algorithm is described as follows.

Miller's original algorithm

```
Input An prime integer $l = \sum_{i=0}^{n} l_i 2^i$, where $l_i \in \{0,1\}$. $P \in E(F_q)[l]$ and
    $Q \in E(F_{q^k})[l]$
Output $e_l(P,Q)$
1. $T \leftarrow P$, $f_1 \leftarrow 1$
2. for $i = n-1, n-2, ..., 1, 0$ do
 2.1 $f_1 \leftarrow f_1^2 \cdot \frac{l_{T,T}(Q)}{v_{2T}(Q)}$, $T \leftarrow 2T$
 2.2 if $l_i = 1$ then
 2.3 $f_1 \leftarrow f_1 \cdot \frac{l_{T,P}(Q)}{v_{T+P}(Q)}$, $T \leftarrow T + P$
3. return $f_1^{(q^k-1)/l}$
```

2.3 Double-base Number System

The double-base number system (DBNS) [12] is a representation for integers. It can reduce plus or minus signs for representing an positive integer highly. Every positive integer n can be represented as the sum of {2, 3}-integers (i.e., numbers of the form $2^b 3^t$) as

$$k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i}$$

where $s_i \in \{-1, 1\}, b_i,\ t_i \geq 0$.

The work of [12] gives an efficient greedy algorithm for representing k sparsely using double-base chain. Let b_{max} and t_{max} be the highest power of 2 and 3 in double-base chain, respectively. We cite Table 7 in [12] generated by using 10000 randomly chosen 160-bit integers.

Table 1. Average number of terms and different largest binary and ternary exponents

b_{max}	t_{max}	m	b_{max}	t_{max}	m
57	65	45	95	41	37
76	53	38	103	36	39

3 Miller's Algorithm Using Double-base Chain

In general, affine coordinates are more efficient than projective coordinates[16, 21] in the computation of the Tate pairing. Therefore affine coordinate system is employed in our algorithm. We will borrow the point tripling [11] and the point quadrupling [1] in affine coordinates for computing the Tate pairing efficiently.

Point tripling algorithm [11] and quadrupling algorithm [1] in affine coordinates are described respectively as follows.

Point Tripling Algorithm in [11]

```
Input $P(x_1, y_1) \in E(F_q)$
Output $3P = (x_3, y_3)$
1. $t_1 \leftarrow (2y_1)^2$, $m \leftarrow 3x_1^2 + a$, $s \leftarrow m^2$
2. $d_1 \leftarrow (3x_1) \cdot t_1 - s$
3. $d \leftarrow (2y_1) \cdot d_1$; $I \leftarrow d^{-1}$
4. $\lambda_1 \leftarrow d_1 \cdot I \cdot M$; $\lambda_2 \leftarrow -\lambda_1 + t_1^2 \cdot I$
5. $x_3 \leftarrow (\lambda_2 - \lambda_1) \cdot (\lambda_2 + \lambda_1) + x_1$
6. $y_3 \leftarrow \lambda_2 \cdot (x_1 - x_3) - y_1$
```

It is easily checked that the cost of point tripling algorithm is 1I+7M+4S.

Point Quadrupling Algorithm in [11]

```
Input $P(x_1, y_1) \in E(F_q)$
Output $2P = (x_2, y_2), 4P = (x_4, y_4) \in E(F_q)$
1. $m \leftarrow 3x_1^2 + a$, $s \leftarrow x_1 \cdot (2y_1)^2$, $t \leftarrow (2y_1)^4$
2. $e \leftarrow 2m \cdot (3s - m^2) - t$, $E \leftarrow (2y_1) \cdot e$,
3. $I \leftarrow E^{-1}$, $(2y_1)^{-1} \leftarrow e \cdot I$, $(2y_2)^{-1} \leftarrow t \cdot I$
4. $\lambda_1 \leftarrow m \cdot (2y_1)^{-1}$,
5. $x_2 \leftarrow \lambda_1^2 - 2x_1$, $y_2 \leftarrow \lambda_1 \cdot (x_1 - x_2) - y_1$
6. $\lambda_2 \leftarrow (3x_2^2 + a) \cdot (2y_2)^{-1}$
7. $x_4 \leftarrow \lambda_2^2 - 2x_2$, $y_4 \leftarrow \lambda_2 \cdot (x_2 - x_4) - y_2$
```

It is easily checked that the cost of point quadrupling algorithm is 1I+9M+7S.

We denote the computational cost of point addition, doubling, tripling and quadrupling in affine coordinates on the elliptic curve as ECADD, ECDBL, ECTRL and ECQDL respectively. Table 2 summarizes these simple operations on $E(F_q)$.

Table 2. Cost of simple operations on $E(F_q)$

operation	cost
ECADD	$1I + 2M + 1S$
ECDBL	$1I + 2M + 2S$
ECTRL [11]	$1I + 7M + 4S$
ECQDL [1]	$1I + 9M + 7S$

3.1 Miller's Algorithm Using Double-base Chain

Let l be a positive integer represented in double-base chain. Let $P \in E(F_q)[l]$ and $Q \in E(F_{q^k})[l]$. Miller's algorithm based on double-base chain is described as follows.

Miller's Algorithm using Double-Base Chain

```
Input An integer $l = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i}$, with $s_i \in \{-1, 1\}$, and such that $b_1 \geq b_2 \geq ... \geq b_m \geq 0$, and $t_1 \geq t_2 \geq ... \geq t_m \geq 0$; and $P = (x_P, y_P) \in E(F_q)[l]$ and $Q = (x_Q, y_Q) \in E(F_{q^k})[l]$
Output $e_l(P, Q)$
1. $T \leftarrow P$, $f_{-1} = 1/(x_Q - x_P)$; if $s_1 = 1$, then $f_1 \leftarrow 1$; else $f_1 \leftarrow f_{-1}$
2. for $i = 1, ..., m-1$ do
 2.1 $u \leftarrow b_i - b_{i+1}, v \leftarrow t_i - t_{i+1}$
 2.2 if $u = 0$ then
  2.2.1 for $j = 1, ..., v$ do
  2.2.2   $f_1 \leftarrow f_1^3 \cdot \frac{l_{T,T}(Q) l_{T,2T}(Q)}{v_{2T}(Q) v_{3T}(Q)}$, $T \leftarrow 3T$
 2.3 else
  2.3.1 if $v = 0$ then
  2.3.2   for $j = 1, ..., u$ do
  2.3.3     $f_1 \leftarrow f_1^2 \cdot \frac{l_{T,T}(Q)}{v_{2T}(Q)}$, $T \leftarrow 2T$
  2.3.4 else
  2.3.5   for $j = 1, ..., u$ do
  2.3.6     $f_1 \leftarrow f_1^2 \cdot \frac{l_{T,T}(Q)}{v_{2T}(Q)}$, $T \leftarrow 2T$
  2.3.7   for $j = 1, ..., v$ do
  2.3.8     $f_1 \leftarrow f_1^3 \cdot \frac{l_{T,T}(Q) l_{T,2T}(Q)}{v_{2T}(Q) v_{3T}(Q)}$, $T \leftarrow 3T$
 2.4 if $s_{i+1} = 1$, then $f_1 \leftarrow f_1 \cdot \frac{l_{T,P}(Q)}{v_{T+P}(Q)}$, $T \leftarrow T + P$
 2.5 else $f_1 \leftarrow f_1 \cdot f_{-1} \frac{l_{T,-P}(Q)}{v_{T-P}(Q)}$, $T \leftarrow T - P$
3. return $f_1^{(q^k-1)/l}$
```

It seems that our new algorithm would be fast because of the sparseness of an integer represented in double-base chain. In the following section we will analyze the complexity of the addition, subtraction, doubling and tripling part of computing the Tate pairing. In Section 5 We will show that our modified Miller's algorithm is more efficient than the previous fastest method indeed.

4 The Complexity of Computing the Tate Pairing

We neglect the cost of field additions and subtractions, as well as the cost of multiplication by small constants. The computational cost of a multiplication, a squaring, an inversion in F_q^* can be denoted the same as above. The computational cost of a multiplication, a squaring, and an inversion in $F_{q^k}^*$ can be represented by M_k, S_k and I_k. A multiplication between elements in $F_{q^k}^*$ and F_q^* is denoted as M_b. We will take $M_k = k^{1.6}M$ and $M_b = kM$ in the efficiency consideration.

The computational cost of the addition, subtraction, doubling and tripling part of the computation of the Tate pairing is represented by TADD, TSUB, TDBL and TTRL, respectively. We consider their computational cost in detail in this section. Some useful remarks are given as below.

For an element $g \in F_{q^k}$, the conjugates of g is denoted by $\overline{g}$. Notice that we can replace $(x_Q - x_1)^{-1}$ by $\overline{x_Q} - x_1$ since $\overline{x_1} = x_1 \in F_q^*$ in the computation of the Tate pairing generally. This replacement is similar to the pseudo-inversion method in [21] and leads to the reducibility of inversive operations in Miller's algorithm. In the sequel, we analyze the complexity of TADD, TSUB, TDBL, TTRL, and iterated TDBL respectively.

4.1 The Computational Cost of TADD

Let $T(x_1, y_1)$, $P(x_P, y_P) \in E(F_q)$. TADD algorithm is described as below.

```
TADD Algorithm
  Input $f \in F_{p^k}^*$,  $T(x_1,y_1), P(x_P,y_P) \in E(F_q)$, $Q(x_Q,y_Q) \in E(F_{q^k})$
  Output the updated $f$
  1. $T_5(x_5,y_5) \leftarrow$ ECADD$(T,P)$
  2. $l_1 \leftarrow (y_Q - y_P) - \lambda(x_Q - x_P)$
  3. $l_2 \leftarrow x_Q - x_5$
  4. $f \leftarrow f \cdot l_1\overline{l_2}$
  5. return $f$
```

If we take advantage of the polynomial expansion method [21, 19] and the fact that $\overline{x_5} = x_5$, we have

$$\begin{aligned} l_1\overline{l_2} &= ((y_Q - y_P) - \lambda(x_Q - x_P))(\overline{x_Q} - x_5) \\ &= (y_Q - y_P)\overline{x_Q} - \lambda(x_Q - x_P)\overline{x_Q} - x_5(y_Q - y_P) + \lambda\ x_5(x_Q - x_P). \end{aligned}$$

Notice that $(y_Q - y_P)\overline{x_Q}$ and $(x_Q - x_P)\overline{x_Q}$ can be precomputed. Calculating λx_5 requires $1M$ and calculating three scalar multiplications, $\lambda \cdot (x_Q - x_P)\overline{x_Q}$, $x_5 \cdot (y_Q - y_P)$, $\lambda x_5 \cdot (x_Q - x_P)$ requires $3M_b$. In addition, computing $f \cdot l_1\overline{l_2}$ requires $1M_k$. Therefore the total cost of TADD is $M_k + 3M_b + M +$ ECADD as discussed in [21]. If the pseudo-multiplication [21] method is applied, it can be reduced to $M_k + 2.5M_b + M +$ ECADD with $k \geq 3$. The precomputed cost of TADD is $2M_k + 7M_{k/2} + I_{k/2}$.

4.2 The Computational Cost of TSUB

Let $T(x_1, y_1)$, $P = (x_P, y_P) \in E(F_q)$. Since

$$f_{-1}l_1(Q) = \frac{(y_Q + y_P) - \lambda(x_Q - x_P)}{x_Q - x_P} = \frac{y_Q + y_P}{x_Q - x_P} - \lambda,$$

we can substitute $f_{-1}l_1(Q)$ with $\frac{y_Q+y_P}{x_Q-x_P} - \lambda$ in the subtraction part of the computation of the Tate pairing. Notice that $\frac{y_Q+y_P}{x_Q-x_P}$ can be precomputed. TSUB algorithm is described as follows.

TSUB Algorithm

```
Input $f \in F^*_{p^k}$,$T(x_1, y_1), P(x_P, y_P) \in E(F_q)$, $Q(x_Q, y_Q) \in E(F_{q^k})$
Output the updated $f$
1. $T_6(x_6, y_6) \leftarrow \mathrm{ECADD}(T, -P)$
2. $f_{-1}l_1 \leftarrow \frac{y_Q+y_P}{x_Q-x_P} - \lambda$
3. $l_2 \leftarrow x_Q - x_6$
4. $f \leftarrow f \cdot (f_{-1}l_1)\overline{l_2}$
5. return $f$
```

It is easily checked that the total cost of TSUB is $M_k + (2k+1)M + \mathrm{ECADD}$ if the polynomial expansion method [21, 19] is applied. The precomputed cost of TSUB is $2M_k + I_k$.

4.3 The Computational Cost of TDBL

Let $T(x_1, y_1),\ T_2 = 2T = (x_2, y_2) \in E(F_q)$. TDBL algorithm is described in the following.

TDBL Algorithm

```
Input $T = (x_1, y_1) \in E(F_q)$, $Q = (x_Q, y_Q) \in F_{q^k}$, and $f \in F^*_{q^k}$
Output the updated $f$
1. $T_2 = (x_2, y_2) \leftarrow \mathrm{ECDBL}(T)$
2. $l_1 \leftarrow (y_Q - y_1) - \lambda(x_Q - x_1)$
3. $l_2 \leftarrow x_Q - x_2$
4. $f \leftarrow f^2 \cdot (l_1\overline{l_2})$
5. return $f$
```

The total cost of TDBL is $M_k + S_k + 4M_b + 2M + \mathrm{ECDBL}$ by using the polynomial expansion method [19, 21]. If the pseudo-multiplication [21] method is applied, it can be reduced to $M_k+S_k+3.5M_b+2M+\mathrm{ECDBL}$. The precomputed cost of TDBL is $2M_k$.

4.4 The Computational Cost of TTRL

In this part we study the computational of TTRL. TTRL algorithm is described as follows.

TTRL Algorithm

```
Input $T = (x_1, y_1) \in E(F_q)$, $Q = (x_Q, y_Q) \in F_{q^k}$, and $f \in F^*_{q^k}$
Output the updated $f$
1. $T_3 = (x_3, y_3) \leftarrow \mathrm{ECTRL}(T)$
2. $l_1 \leftarrow \frac{l_{T,T}(Q)l_{T,2T}(Q)}{v_{2T}(Q)}$
3. $l_2 \leftarrow x_Q - x_3$
4. $f \leftarrow f^3 \cdot (l_1\overline{l_2})$
5. return $f$
```

Let $T = (x_1, y_1)$, $T_3 = 3T = (x_3, y_3) \in E(F_q)$. Let $T_2 = 2T = (x_2, y_2)$, where $x_2 = \lambda_1^2 - 2x_1$ as an intermediate. We apply the parabola method [14] to computing TTRL. Notice that $l_{T,T}$ passes through T and $-2T$, whose slope is λ_1, and $l_{T,2T}$ passes through T and $2T$, whose slope is λ_2. We obtain

$$\begin{aligned}
\frac{l_{T,T}(Q)l_{T,2T}(Q)}{v_{2T}(Q)} =& \frac{((y_Q+y_2)-\lambda_1(x_Q-x_2))((y_Q-y_2)-\lambda_2(x_Q-x_2))}{x_Q-x_2} \\
=& \frac{y_Q^2-y_2^2}{x_Q-x_2} + \lambda_1\lambda_2(x_Q-x_2) - \lambda_1(y_Q-y_2) - \lambda_2(y_Q+y_2) \\
=& x_Q^2+x_2x_Q+x_2^2+a+\lambda_1\lambda_2(x_Q-x_2)-\lambda_1(y_Q-y_2)-\lambda_2(y_Q+y_2) \\
=& x_Q^2+(x_2+\lambda_1\lambda_2)x_Q-(\lambda_1+\lambda_2)y_Q+x_2^2+a-\lambda_1\lambda_2x_2+(\lambda_1-\lambda_2)y_2 \\
=& (x_Q-x_1)(x_Q+x_1+x_2+\lambda_1\lambda_2)-(\lambda_1+\lambda_2)(y_Q-y_1)+x_1^2+x_1x_2 \\
& + \lambda_1\lambda_2x_1-(\lambda_1+\lambda_2)y_1+x_2^2+a-\lambda_1\lambda_2x_2+(\lambda_1-\lambda_2)y_2 \\
=& (x_Q-x_1)(x_Q+x_1+x_2+\lambda_1\lambda_2)-(\lambda_1+\lambda_2)(y_Q-y_1) \\
& + \frac{y_1^2-y_2^2}{x_1-x_2}+\lambda_1\lambda_2(x_1-x_2)+(\lambda_1-\lambda_2)y_2-(\lambda_1+\lambda_2)y_1 \\
=& (x_Q-x_1)(x_Q+x_1+x_2+\lambda_1\lambda_2)-(\lambda_1+\lambda_2)(y_Q-y_1) \\
& + (\lambda_1-\lambda_2)((y_2+y_1)-\lambda_1(x_1-x_2)) \\
=& (x_Q-x_1)(x_Q+x_1+x_2+\lambda_1\lambda_2)-(\lambda_1+\lambda_2)(y_Q-y_1)
\end{aligned}$$

by using $y_2 = \lambda_1(x_1 - x_2) - y_1$.

We can expand the above polynomial in powers of x_Q . The expansive polynomial is

$$\begin{aligned}
l_1 =& x_Q^2 + (x_2+\lambda_1\lambda_2)x_Q - (\lambda_1+\lambda_2)(y_Q-y_1) - x_1(x_1+x_2+\lambda_1\lambda_2) \\
=& x_Q^2 + ((\lambda_1(\lambda_1+\lambda_2)-2x_1)x_Q - (\lambda_1+\lambda_2)(y_Q-y_1) - (\lambda_1(\lambda_1+\lambda_2)-x_1)x_1.
\end{aligned}$$

Computing $\lambda_1 \cdot (\lambda_1+\lambda_2)$ and $(\lambda_1(\lambda_1+\lambda_2)-x_1)\cdot x_1$ requires $2M$, and computing $((\lambda_1(\lambda_1+\lambda_2)-2x_1)\cdot x_Q$ and $(\lambda_1+\lambda_2)\cdot(y_Q-y_1)$ requires $2M_b$. Therefore we need $2M_b + 2M$ for computing l_1. Notice that x_Q^2 can be precomputed.

We require $1S_k + 3M_k$ for computing $f \cdot f^2 \cdot l_1 \cdot \overline{l_2}$. Thus the total cost of TTRL is $3M_k + S_k + 2M_b + 2M +$ ECTRL. The precomputed cost is $1S_k$.

4.5 The Computational Cost of Iterated TDBL

The iterated TDBL algorithm was first proposed in the work [19] of Izu. The work of [21] gave an improved iterated TDBL algorithm. We give a modified iterated TDBL algorithm by using point quadrupling in affine coordinates.

Iterated TDBL Algorithm

```
  Input  $f \in F_{q^k}$,$T = (x_1, y_1) \in E(F_q)$,  and  $Q = (x_Q, y_Q) \in F_{q^k}$
  Output  the updated $f$
  1. $T_2(x_2, y_2) = 2T_1, T_4(x_4, y_4) = 4T_1 \leftarrow$ ECQDL($T$)
```

```
2. $l_1 \leftarrow y_Q - y_1 - \lambda_1(x_Q - x_1)$
3. $l_2 \leftarrow x_Q - x_2$
4. $f^{'} \leftarrow f^2 \cdot l_1 \cdot \overline{l_2}$
5. $l_1^{'} \leftarrow y_Q - y_2 - \lambda_2(x_Q - x_2)$
6. $l_2^{'} \leftarrow x_Q - x_4$
7. $f \leftarrow f^{'2} \cdot l_1^{'} \cdot \overline{l_2^{'}}$
```

Now we study the computation of f in iterated TDBL. Since

$$\text{the updated } f = f^{'2} \cdot l_1^{'} \cdot \overline{l_2^{'}} = \quad f^4 \cdot l_1^2 \cdot \overline{l_2^2} \cdot l_1^{'} \cdot \overline{l_2^{'}} = (f^2 \cdot l_1)^2 \cdot l_1^{'} \cdot \overline{l_2^2 l_2^{'}}$$

We can apply the polynomial expansion method in $l_2^2 l_2^{'}$. Then

$$l_2^2 l_2^{'} = (x_Q - x_2)^2 (x_Q - x_4) = x_Q^3 - (x_4 + 2x_2)x_Q^2 + (2x_2x_4 + x_2^2)x_Q - x_2^2 x_4$$

Since x_2^2 is computed in ECQDL(T_1), $x_2 \cdot x_4$ and $x_2^2 \cdot x_4$ require 2M. $(x_4 + 2x_2) \cdot x_Q^2$ and $(2x_2x_4 + x_2^2) \cdot x_Q$ require $2M_b$. Therefore, the total cost of iterated TDBL is

$$3M_k + 2S_k + 4M_b + 2M + \text{ECQDL}.$$

It is obviously more efficient than the iterated TDBL in [21] if the cost of 1ECQDL is lower than the cost of 2ECDBL.

We summarize these results in the following table.

Table 3. Cost of elementary operations for computing the Tate pairing

operation	cost
TADD	$M_k + 2.5M_b + 1I + 3M + 1S$
TSUB	$M_k + 1I + (2k+3)M + 1S$
TDBL	$M_k + S_k + 3.5M_b + 1I + 4M + 2S$
Iterated TDBL	$3M_k + 2S_k + 4M_b + 1I + 11M + 7S$
TTRL	$3M_k + S_k + 2M_b + 1I + 9M + 4S$

5 Efficiency Consideration

In this section we put our emphasis on studying the efficiency of new proposed algorithm and comparing it with the previous methods in [21] and [19]. Since it is necessary that the order l of elliptic curve subgroup have at least 160 bits and q^k have at least 1000 bits for security and efficiency in many cryptographic protocols, we mainly give our efficiency analysis for $log_2 l = 160$.

5.1 Efficiency of Our Proposed Algorithm

The total precomputed cost of new proposed algorithm is

$$T_{pre} = 6M_k + S_k + I_k + 7M_{k/2} + I_{k/2}.$$

We assume that plus signs and minus signs appear on average if l is represented in double-base chain. The total cost of our algorithm is

$$\begin{aligned}
&b_{max}\mathrm{TDBL} + t_{max}\mathrm{TTRL} + \frac{m}{2}(\mathrm{TADD} + \mathrm{TSUB}) + T_{pre}\\
=&((b_{max} + 3t_{max} + m + 6)M_k + ((b_{max} + t_{max} + 1)S_k\\
&+ (\frac{7}{2}b_{max} + 2t_{max} + \frac{5}{4}m)M_b + (b_{max} + t_{max} + m)I\\
&+ (4b_{max} + 9t_{max} + (k+3)m)M + (2b_{max} + 4t_{max} + m)S + I_k + 7M_{k/2} + I_{k/2}.
\end{aligned}$$

Here we take $M_4 = 9M$, $M_6 = 18M$, $M_8 = 27M$, $S = 0.8M$, $I = 10M$, $M_b = kM$, $I_k = I + k^2M$. We estimate the cost of new proposed algorithm by Table 1 since the 160 bit length of l is enough for security in many cryptographical protocols. Notice that we do not use the iterated TDBL technique [19, 21] for simplicity.

Table 4. Cost of new proposed algorithm by Table 1

b_{max}	t_{max}	m	cost(k=4)	cost(k=6)	cost(k=8)
57	65	45	8287M	12744M	17222M
76	53	38	8350M	12554M	17085M
95	41	37	8395M	12552M	17052M
103	36	39	8493M	12676M	17186M

5.2 Comparisons

In this part, we compare our algorithm with the previous methods in [19] and [21] mainly.

If we assume that k is a randomly chosen n-bit integer, it is well known that the double-and-add algorithm requires n doubling and $n/2$ additions on average. Using the signed digit representation, the average density of non-zero digits is reduced to 1/3. Therefore the computational cost of Miller's original algorithm on average is the sum of nTDBL and $\frac{n}{2}$TADD. The computational cost of signed Miller's algorithm is

$$n\mathrm{TDBL} + \frac{n}{3}(\frac{1}{2}\mathrm{TADD} + \frac{1}{2}\mathrm{TSUB}).$$

The computational cost of Miller's algorithm described in [19] is

$$\begin{aligned}&(n-1)(2M_k+2S_k+(3k+7)M+6S)+\frac{n}{2}(2M_k+(3k+10)M+3S)\\=&(3n-2)M_k+(2n-2)S_k+(\frac{9n-6}{2}k+12n-7)M+\frac{15n-12}{2}S.\end{aligned}$$

The computational cost of signed Miller's algorithm described in [21] is

$$\begin{aligned}&(n-1)\mathrm{TDBL}+\frac{n}{6}(\mathrm{TADD}+\mathrm{TADD})+T_{pre}\\=&(\frac{4}{3}n+5)M_k+(n-1)S_k+I_k+7M_{k/2}+1I_{k/2}\\&+(\frac{17n-14}{4}k+(5n-4))M+(\frac{7}{3}n-2)S+(\frac{4}{3}n-1)I.\end{aligned}$$

Now we compare our algorithm with the proposed methods in [21] and [19]. The assumptions of $M_k,\ I$ and n are the same as above. The computational cost of these algorithms is shown in the following Table.

Table 5. Cost comparison of proposed algorithms

Algorithm	cost(k=4)	cost(k=6)	cost(k=8)
new proposed algorithm	8350M	12554M	17085M
signed Miller's algorithm in [21]	9196M	13685M	18121M
Miller's algorithm in [19]	12328M	20353M	28379M

Table 5 implies that our proposed algorithm is 9% faster and 62% faster than the previous proposed algorithms in [21] and [19] for $k = 6$, respectively. the sparseness of DBNS leads to fewer multiplications than the previous methods in [21, 19]. We also point out that our proposed algorithm can be applied for $k = 2$.

6 Conclusion

In this paper, A new algorithm based on the double-base chain is proposed for computing the Tate pairing. The proposed algorithm can be applied to supersingular and ordinary elliptic curves. It is more efficient than the previous proposed algorithms in [21] and [19].

References

1. D. Adachi and T. Hirata, *Refined computations for points of the form 2^kP based on Montgomery Trick*, IEICE Trans.Fundamentals. Vol.E89-A, No,1, pp.334-339, Jan 2006.

2. 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, pp. 354-368. Springer-Verlag, 2002.
3. Paulo Barreto's pairing-based crypto lounge,
http://paginas.terra.com.br/informatica/paulobarreto/pblounge.html
4. P.S.L.M. Barreto, S. Galbraith, C. ÓhÉigeartaigh, and Michael Scott, *Efficient pairing computation on supersingular abelian varieties.* Cryptology ePrint Archive, Report 2004/375.
5. I. Blake, K. Murty and G. Xu, *Refinements of Miller's algorithm for computing Weil/Tate pairing*, Cryptology ePrint Archive, Report 2004/065.
6. D. Boneh, X. Boyen, *Secure identity based encryption without random oracles*, Advances in Cryptology-Crypto'2004, LNCS 3152, pp.443-459. Springer-Verlag, 2004.
7. D. Boneh and X. Boyen, *Short signatures without random oracles*, Advances in Cryptology-Eurocrypt 2004, LNCS 3027, pp.56-73, Springer-Verlag, 2004.
8. D. Boneh, X. Boyen, H. Shacham, *Short group signatures.* Advances in Cryptology-Crypto'2004, LNCS 3152, pp.41-55. Springer-Verlag, 2004.
9. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing.* Advances in Cryptology-Crypto'2001, LNCS 2139, pp. 213-229. Springer-Verlag, 2001.
10. D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairing*, Advances in Cryptology-Asiacrypt 2001, LNCS 2248, pp.514-532, Springer-Verlag, 2001.
11. M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery, *Trading inversions for multiplications in elliptic curve cryptography*, Designs, Codes and Cryptography. Vol 39, No 2, pp.189-206, May 2006.
12. V.S. Dimitrov, L. Imbert, and P.K. Mishra, *Efficient and secure elliptic curve point multiplication using double-base chains.* Advances in Cryptology-Asiacrypt'2005, LNCS 3788, pp.59-78. Springer-Verlag, 2005.
13. I. Dutta, and H.-S. Lee, *Tate pairing implementation for hyperelliptic curves* $y^2 = x^p - x + d$. Advances in Cryptology-Asiacrypt'2003, LNCS 2894, pp.111-123. Springer-Verlag, 2003.
14. K. Eisenträer, K. Lauter, and P.L. Montgomery, *Fast elliptic curve arithmetic and improved Weil pairing evaluation.* CT-RSA 2003, LNCS 2612, pp.343-354. Springer-Verlag, 2003.
15. G. Frey and H. Rück, *A remark concerning* m*-divisibility and the discrete logarithm in the divisor class group of curves.* Math. Comp., 52, pp.865-874, 1994.
16. S.D. Galbraith, *Pairings - Advances in elliptic curve cryptography.* Cambridge University Press, 2005.
17. S.D. Galbraith, K. Harrison, and D. Soldera. *Implementing the Tate pairing.* Algorithm Number Theory Symposium - ANTS V, LNCS 2369, pp. 324-337. Springer-Verlag, 2002.
18. F. Hess and N.P. Smart and F. Vercauteren, *The Eta Pairing revisited*, Cryptology ePrint Archive, Report 2006/110.
19. T. Izu and T. Takagi, *Efficient computations of the Tate pairing for the large MOV degrees*, ICISC'02, LNCS 2587, pp.283-297. Springer-Verlag, 2003.
20. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, Algorithm Number Theory Symposium - ANTS IV, LNCS 1838, pp.385-394. Springer-Verlag, 2000.
21. T. Kobayashi, K. Aoki, H. Imai, *Efficient algorithms for Tate pairing*, IEICE Trans. Fundamentals, VOL.E89-A, NO.1 Jan 2006.

22. A. Menezes, T. Okamoto, and S. Vanstone. *Reducing elliptic curve logarithms to logarithms in a finite field.* IEEE Transaction on Information Theory, 39: 1639-1646, 1993.
23. V.S. Miller. *Short programs for functions on curves.* unpublished manuscript, 1986.
24. P.L. Montgomery. *Speeding the Pollard and elliptic curve methods of factorization.* Math. Comp., vol.48, pp.243-264,1987.
25. J.A. Muir, D.R. Stinson, *On the low hamming weight discrete logarithm problem for nonadjacent representations*, Applicable Algebra in Engineering, Communication and Computing, vol 16, pp.461-472, Jan 2006.
26. R. Sakai, K. Ohgishi, M. Kasahara, *Cryptosystems based on pairing*, SCIC 2000-C20, Jan. 2000. Okinawa, Japan.
27. J.H. Silverman, *The arithmetic of elliptic curves.* Springer-Verlag, New York, 1986.
28. B. Waters, *Efficient identity-based encryption without random oracles*, In Advances in Cryptology Eurocrpt 2005, LNCS 3494, pp. 114-127, Springer-Verlag, 2005.
29. F. Zhang, R. Safavi-Naini and W. Susilo, *An efficient signature scheme from bilinear pairings and its applications*, PKC 2004, LNCS 2947, pp.277-290, Springer-Verlag, 2004.