Bilinear Pairing Computation Using the Extended Double-Base Chains Algorithm

Abdulwahed M. Ismail, Mohamad Rushdan MD Said, Kamel Ariffin Mohd Atan and Isamiddin S. Rakhimov

Institute for Mathematical Research (INSPEM)
University Putra Malaysia
Serdang, 43300, Selangore, Malaysia
wahid963@yahoo.com
mrushdan@putra.upm.edu.my
kamel@putra.upm.edu.my
isamiddin@science.upm.edu.my

Abstract. Elliptic curve (EC) pairings have been the focus of attention of researchers and cryptographers, especially after identity-based cryptosystems (IBC) were proposed in 2001. The Weil and Tate pairing is considered as the most important pairings used in cryptographical protocols and their applications. The computation efficiency of the Weil and Tate pairings mainly depends on the efficiency of the EC scalar multiplications algorithms used. In this paper, we compute the Tate pairing using multi-base number representation (MBNR) system in scalar multiplication instead of using binary representation as used in Miller’s algorithm and in the double-base (DB) chain used by Changan Zhao et al. We show that using doubling, tripling and quintupling in scalar multiplication, computation of the Tate pairing and its applications can be significantly enhanced.

Keywords: Bilinear pairings, Tate pairing, Miller’s Algorithm, Elliptic curves cryptography, Multi-number representation system

1. Introduction

In 1993, Menezes, Okamato and Vanstone (MOV) [21] used the Weil pairing to convert the elliptic curves discrete logarithm problem (ECDLP) into a discrete
logarithm problem (DLP) in some extensions of the base field. In the same year, Frey and Rück [24] used the Tate pairing for the same purpose as MOV, but on a wider range of ECs. In 2000, Joux [23] used EC pairings to propose a cryptosystem which creates and exchanges the secret key between three entities instead of two entities as in the Diffie Hellman Protocol [8]. The idea of IBC was first proposed by Shamir in 1984 [22]. The method uses the identities of entities as their public keys, such that no certification is needed from any trusted authority. Later, in 2000, Sakai, Ohgishi and Kashara [26] used EC pairings in IBC, which were proposed originally by Shamir. However, his signature scheme was without encryption and key sharing protocols until 2001.

In Section 2 of this paper, some basic and necessary background on the ECs group law, EC scalar multiplication, bilinear pairings with some related divisors, and rational function are provided. In Section 3, Tate pairing computation is presented. Section 4 discusses our proposed algorithm, with experimental results and efficiency considerations made. Section 5 concludes this paper.

2. Background

2.1. Elliptic Curves. Let \( p \) be a prime number, \( m \) a positive integer, and \( F_{p^m} \) the finite field with \( q = p^m \) elements. \( p \) is said to be the characteristic of \( F_{p^m} \), and \( m \) is its extension degree. For curves over finite fields of characteristic more than three, we consider the simple form of the elliptic curve equation:

\[
E : y^2 = x^3 + ax + b
\]

with \( a, b \) in \( F_q \) and \( 4a^3 + 27b^2 \neq 0 \). The group used for the cryptosystem is the group of points on the curve over \( F_q \). If represented in affine coordinates, the points have the form \( (x, y) \), where \( x \) and \( y \) are in \( F_q \) and satisfy the equation of the curve. They also have a distinguished point \( O \) (called the point at infinity), which acts as the identity for the group. The addition of any two distinct points on the elliptic curve will be done by using some formulas as shown below. When \( P = (x_P, y_P) \) and \( Q = (x_Q, y_Q) \) are not negative of each other, then \( P + Q = R \).

We first need to find \( \lambda \) as the slope of the line through \( P \) and \( Q \). More details about the EC group law can be found in [2], [12] and [5].

\[
\lambda = \frac{y_P - y_Q}{x_P - x_Q}
\]

\[
x_R = \lambda^2 - x_P - x_Q
\]
Bilinear pairing computation

\[ y_R = -y_P + \lambda(x_P - x_R) \]

The doubling operation for any point in the elliptic curve group requires some steps (the result is the same point when doubling \(2P\), or adding \(P + P\)). Let us say we want to double the point \(P = (x_P, y_P)\), when \(y_P\) is not 0. We have \(2P = D\), where \(D = (x_D, y_D)\) and

\[
\lambda = \frac{3x_P^2 + a}{2y_P} \\
x_D = \lambda^2 - 2x_P \\
y_D = -y_P + \lambda(x_P - x_D)
\]

For each doubling of point \(P\), we need the above procedure to obtain \(2P\), \(2(2P)\), \(2(2(2P))\), and so on. Here, we note that to compute \(lP\) for any \(l\) integer, we need a number of doublings and additions until we obtain \(P + P + \ldots + P\) \((l\)-times). The scalar \(l\) in elliptic curve cryptosystems is considered as a secret key, which usually has very long bits. Such point multiplication arithmetic requires much memory and time for running and implementing necessary cryptographical operations.

Points are added using a geometric group law, which can be expressed algebraically through rational functions involving \(x\) and \(y\). Whenever two points are added, forming \(P + Q\), or whenever a point is doubled, forming \(2P\), these formulas are evaluated at the cost of some number of multiplications, squarings, and divisions in the field. For example, to double a point in affine coordinates using the short Weierstrass form, requires 1 multiplication, 2 squarings, and 1 division in the field, not counting multiplication by 2 or 3 [2]. To add two distinct points in affine coordinates requires 1 multiplication, 1 squaring, and 1 division in the field. Performing a doubling and an addition \(2P + Q\) requires 2 multiplications, 3 squarings and 2 divisions if the points are added as \((P + P) + Q\), that is we first double \(P\) and then add \(Q\) [1].

Improving the efficiency of scalar multiplication in elliptic curves is one of the main interests of many researchers in the field of cryptology. The techniques proposed so far use different techniques for representing the scalar \(l\), which clearly show different levels of computational speed and security. Binary representation is extended to signed binary representation and it is the Non-Adjacent Form NAF algorithm. Other well-known techniques such as the Window method and Montgomery method bring about much improvement in terms of the efficiency of the EC arithmetic [12] and [5].

2.2. Multi-Base Number Representation (MBNR).
2.2.1. **Double Base Chain.** An important contribution by [13] was a new ternary/binary method to perform efficient scalar multiplication. This method evaluates the expressions of the form $6P \pm Q$, which can be computed as $2(3P) \pm Q$ or $3(2P) \pm Q$. When using the short Weierstrass form $y^2 = x^3 + ax + b$, the latter takes an extra inversion but saves five (field) multiplications. For binary curves, the costs are $3I + 4S + 11M$ and $2I + 6S + 16M$, so the trade-off is $1I$ for $2S + 5M$.

A similar idea was suggested in [14] when an integer $k$ is represented in the double-base number system (DBNS) as the sum or difference of mixed powers of two and three.

2.2.2. **Multi-Base Chain.** The above so-called DBNS has been generalized to multi-base number representation by the authors in their recent paper [17]. They proposed two efficient formulas for computing $(5P)$, when $P$ is an elliptic curve point over prime and binary finite fields. This work led to a new scalar multiplication algorithm, which represents the scalar using three bases of 2, 3 and 5 and computes the scalar multiplication very efficiently.

**Definition 1.** A multiple representation $l = \sum s_i 2^{b_i} 3^{t_i} 5^{r_i}$ using the bases \{2,3,5\} is called a step multi-base number representation (SMBR), where each exponent \{b_i\}, \{t_i\} and \{r_i\} refers to separate monotonic decreasing sequences.

The lengths of MBNR are shorter than those of DBNS and are also more redundant, as the number of representations of $n$ grows very fast based on the number of base elements. In the example shown in [17], the number 50 has 72 DBN representations using the bases (2 and 3), while it has 489 MBN representations using the bases (2, 3 and 5). The special MBNR is more suitable for scalar multiplication algorithms than the general MBNR. According to the experiments conducted by the authors of [17], the multi-base algorithm performs faster and more competitively as compared to other sequential scalar multiplication algorithms. In algorithm 4 for computing scalar multiplications over binary finite fields, the required curve operations can be calculated as $b_i$ doublings, $t_i$ triplings and $r_i$ quintauplings. The number of curve additions is the same as the number of terms in the chain. Whenever the components of the binary and ternary are not zero, we can use double-and-add and triple-and-add operations instead of curve addition.

Let $b_{\text{max}}$, $t_{\text{max}}$ and $r_{\text{max}}$ be the highest power of 2,3 and 5 in the SMBR, respectively. In the table below, we show some examples of 3-base number representation for integers of 160 bits. $m$ is denoted to represent the number of terms in the chain. In this work, we consider point $P$ in affine coordinates for Tate pairing computation, where the cost of quintupling operation ECQNT is
Bilinear pairing computation

<table>
<thead>
<tr>
<th>$b_{\text{max}}$</th>
<th>$t_{\text{max}}$</th>
<th>$f_{\text{max}}$</th>
<th>$m$</th>
<th>$b_{\text{max}}$</th>
<th>$t_{\text{max}}$</th>
<th>$f_{\text{max}}$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>47</td>
<td>14</td>
<td>33</td>
<td>70</td>
<td>24</td>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>55</td>
<td>28</td>
<td>26</td>
<td>38</td>
<td>78</td>
<td>25</td>
<td>18</td>
<td>30</td>
</tr>
<tr>
<td>58</td>
<td>45</td>
<td>13</td>
<td>32</td>
<td>83</td>
<td>25</td>
<td>16</td>
<td>34</td>
</tr>
<tr>
<td>65</td>
<td>35</td>
<td>17</td>
<td>29</td>
<td>88</td>
<td>23</td>
<td>15</td>
<td>37</td>
</tr>
<tr>
<td>69</td>
<td>25</td>
<td>22</td>
<td>25</td>
<td>94</td>
<td>24</td>
<td>12</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 1. Maximum exponents and terms for some 160 bit size integers

9[S]+17[M] ≈ 24.2[M] (using 2(2P) + P) and requires 12[S]+22[M] ≈ 31.6[M] (using 2P + 3P) [17]. We borrow the requires of computing other operations from [13][1]. Elliptic curves point addition ECADD in affine coordinates requires (1I+2M+1S), while doubling ECDBL requires (1I+2M+2S) and tripling ECTRL costs (1I+7M+4S).

2.3. Bilinear Pairings. The map $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing function, where $G_1$ is an additive abelian group with identity element 0. $G_2$ is a multiplicative cyclic group of order $l$ with identity element 1. Let $l$ be a positive integer. $G_1 = \langle P \rangle$ has the order $l$ such that $[l]P = 0$ for all $P \in G_1$.

Let $E$ be an elliptic curve over a field $K$. Let $P \in E(F_q)$ be a point of prime order $l$. The embedding degree $k$ is the smallest integer such that $l$ must divide $q^k - 1$. The set of $l$-torsion points of $E$ is denoted by $E[l]$ and is defined as \{ $P \in E(F_q)$ | $[l]P = \mathcal{O}$ \}.

2.3.1. Divisors. Let $E$ be an elliptic curve over a field $K$. For each point $P \in E(\bar{K})$, a divisor $D$ is a formal sum of zeros and poles on an elliptic curve $E$. The divisor $D$ is an element of the free abelian group (denoted by $\text{Div}(E)$) generated by the set of points of $E(\bar{K})$. Given a divisor $D = \sum_{P \in E} l_P(P)$, the degree of $D$ is defined by $\text{deg}(D) = \sum_{P \in E} l_P$. The subgroup of divisors of degree 0 is selected to be used in computing pairings, namely, $\text{Div}^0(E) = \{ D \in \text{Div}(E) : \text{deg}(D) = 0 \}$. More details can be found in each of [18] [6][7].

For a nonzero rational function $f$ over $E$, we define $\text{div}(f) = \sum_{P \in E} \text{ord}_P(f)(P)$. It turns out that $\text{div}(f)$ is an element in $\text{Div}^0(E)$ and is called a principal divisor. A characterization of principal divisors is as follows: $D = \sum_{P \in E} l_P(P) \in \text{Div}^0(E)$ is principal iff $\sum_{P \in E} l_P P = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity.

**Theorem 1.** [7] Let $E$ be an elliptic curve over a field $K$. Let $D = \sum_{P} l_P(P)$ be a degree zero divisor on $E$. Then $D \sim 0$, which means there exists a function $f$ such that $D = \text{Div}(f)$ if and only if $D = \sum_{P} [l_P]P = \mathcal{O}$ on $E$.

The relation $\sim$ on $\text{Div}^0(E)$ is defined as $D_1 \sim D_2$ if $D_1 - D_2$ is principal. The support of a divisor $D = \sum_{P \in E} l_P(P)$ is the set of points $P$ with $l_P \neq 0$. If $f$
is a nonzero rational function such that \( \text{div}(f) \) and \( D \) have disjoint supports, we can extend the evaluation of \( f \) at \( D \) by defining \( f(D) = \Pi_{P \in E} f(P)^{l_P} \).

2.3.2. Tate Pairings. Let \( l \) be an integer which is prime to \( p = \text{char}(K) \) if \( p > 0 \), and \( E[l] = \{ P \in E(K) : lP = O \} \). If \( P, Q \in E[l] \), there exist \( D_Q \in \text{Div}^0(E) \) such that \( D_Q \sim (Q) - (O) \). It follows there exist functions \( f_P \) such that \( \text{div}(f_P) = lD_P = l(P) - l(O) \). This means that \( f_p \) has a zero of order \( l \) at \( P \) and a pole of order \( l \) at \( O \), and that there are no other zeros and poles. Let \( R \in E[l] \) such that \( R \notin \{ O, P, -Q, P - Q \} \), and let \( D_Q = (Q + R) - (R) \). \( R \) is supposed to be selected in a way that \( D_Q \) and \( \text{div}(f_P) \) have disjoint support. Then the formula of the reduced Tate pairing of order \( l \) is the map 
\[
e_l : E(F_q)[l] \times E(F_{q^k})[l] \rightarrow F_{q^k}^*
\]
defined as follows [6][25]:
\[
e_l(P, Q) = f_P(D_Q)^{(q^k - 1)/l} = (f_P(Q + R)/f_P(R))^{(q^k - 1)/l}
\]
The Tate pairing satisfies the following properties:

- (Bilinearity) \( e_l(P + R, Q) = e_l(P, Q) \cdot e_l(R, Q) \) and \( e_l(P, Q + S) = e_l(P, Q) \cdot e_l(P, S) \) for all \( P, R \in E(F_q)[l] \) and all \( Q, S \in E(F_{q^k})[l] \). Then \( e_l(sP, Q) = e_l(P, sQ) = e_l(P, Q)^s \) for any integer \( s \).
- (Non-degeneracy) If \( e_l(P, Q) = 1 \) for all \( Q \in E(F_{q^k})[l] \), then \( P = O \). Alternatively, for each \( P \neq O \), there exists \( Q \in E(F_{q^k})[l] \) such that \( e_l(P, Q) \neq 1 \).
- (Compatibility) Let \( l = hl' \). If \( P \in E(F_q)[l] \) and \( Q \in E(F_{q^k})[l'] \), then \( e_{l'}(hP, Q) = e_l(P, Q)^{h} \).

3. Tate Pairing Computation Using Miller’s Algorithm

Tate pairing is more efficient for use in cryptography because its computation is much faster than other pairings, like Weil pairings [7]. Miller’s algorithm for computing the Tate pairing is of interest for the purpose of cryptographical use. Let \( P \in E(F_q)[l] \) and \( Q \in E(F_{q^k})[l] \) be linearly independent points, and let \( N = \#E(F_q) \), and \( e_l(P, Q) \) the Tate pairing. For sufficient details about the computation of \( e_l(P, Q) \) using Miller’s algorithm, please refer to [6] and [25].

**Theorem 2.** (Miller’s formula). Let \( P \) be a point on \( E(F_q) \) and \( f_c \) be a function with divisor \( (f_c) = c(P) - (cP) - (c - 1)(O) \), \( c \in Z \). For all \( a, b \in Z \),
\[
f_{a+b}(Q) = f_a(Q) \cdot f_b(Q) \cdot g_{a+b,P}(Q)/g_{a+b,P}(Q)
\]
Notice that \((f_0) = (f_1) = 0\), so that \(f_0(Q) = f_1(Q) = 1\). Furthermore, 
\[ f_{a+1}(Q) = f_a(Q) \cdot \frac{g_{aP,P}(Q)}{g_{a+1}P(P(Q))} \] 
and 
\[ f_{2a}(Q) = f_a(Q)^2 \cdot \frac{g_{aP,aP}(Q)}{g_{2aP}(P)} \].

Let the binary representation of \(l \geq 0\) be \(l = (l_t,...,l_1,l_0)\), where \(l_i \in \{0,1\}\) and \(l_t \neq 0\). Miller’s algorithm computes \(f_P(Q) = f_l(Q)\), \(Q \neq \mathcal{O}\) by coupling the above formulas with the double-and-add method to calculate \(lP\). More formally, the Miller’s algorithm can be written as follows:

**Algorithm 1.** (Miller’s algorithm\[18\] \[25\])

**INPUT:** Integer \(l = \sum_{i=0}^t b_i2^i\) with \(b_i \in \{0,1\}\) and \(b_t \neq 0\). Miller’s algorithm computes 
\(f_P(Q) = f_l(Q)\), \(Q \neq \mathcal{O}\) by coupling the above formulas with the double-and-add method to calculate \(lP\). More formally, the Miller’s algorithm can be written as follows:

(input: An integer \(l = \sum_{i=1}^m s_i2^{b_i}3^{t_i}5^{r_i}\), with \(s_i \in \{1,-1\}\), such that \(0 \leq b_1 < b_2 < ... < b_m\), \(t_1 \geq t_2 \geq ... \geq t_m \geq 0\), \(r_1 \geq r_2 \geq ... \geq r_m \geq 0\), \(q \geq b_i \forall i\); and \(P = (x_P,y_P) \in E(F_q)[l]\) and \(Q = (x_Q,y_Q) \in E(F_q)[l]\)

**OUTPUT:** \(e_l(P,Q)\)

1. \(T \leftarrow P, f - 1 \leftarrow \frac{1}{x_q - x_P}\);
   If \(s_1 = 1\), then \(f_1 \leftarrow 1\); else \(f_1 \leftarrow f_{-1}\)
2. for \(i = 1,2,...,m - 1\) do
   2.1 \(u \leftarrow b_i - b_{i+1}\)
       \(v \leftarrow t_i - t_{i+1}\)
       \(w \leftarrow r_i - r_{i+1}\)
   2.2 if \(u = 0\) then
      2.2.1 for \(j = 1,...,v\) do
      2.2.2 \(f_1 \leftarrow f_1^{3LT}Q \cdot L_{2T}(Q)\), \(T \leftarrow 3T\)
   2.3 else
      2.3.1 if \(v = 0\) then

4. **Proposed Algorithm**

We propose a Tate pairing computation method in the form of an algorithm, as follows:

**Algorithm 2.** Miller’s Algorithm Using MBNR.

(input: An integer \(l = \sum_{i=1}^m s_i2^{b_i}3^{t_i}5^{r_i}\), with \(s_i \in \{1,-1\}\), such that \(0 \leq b_1 < b_2 < ... < b_m\), \(t_1 \geq t_2 \geq ... \geq t_m \geq 0\), \(r_1 \geq r_2 \geq ... \geq r_m \geq 0\), \(q \geq b_i \forall i\); and \(P = (x_P,y_P) \in E(F_q)[l]\) and \(Q = (x_Q,y_Q) \in E(F_q)[l]\)

**OUTPUT:** \(e_l(P,Q)\)

1. \(T \leftarrow P, f - 1 \leftarrow \frac{1}{x_q - x_P}\);
   If \(s_1 = 1\), then \(f_1 \leftarrow 1\); else \(f_1 \leftarrow f_{-1}\)
2. for \(i = 1,2,...,m - 1\) do
   2.1 \(u \leftarrow b_i - b_{i+1}\)
       \(v \leftarrow t_i - t_{i+1}\)
       \(w \leftarrow r_i - r_{i+1}\)
   2.2 if \(u = 0\) then
      2.2.1 for \(j = 1,...,v\) do
      2.2.2 \(f_1 \leftarrow f_1^{3LT}Q \cdot L_{2T}(Q)\), \(T \leftarrow 3T\)
   2.3 else
      2.3.1 if \(v = 0\) then


2.3.2 for \( j = 1, \ldots, u \) do
2.3.3 \( f_1 \leftarrow f_1^2 L_{T,T}(Q) \), \( T \leftarrow 2T \)
2.3.4 else
2.4 if \( u = o \) and \( v = 0 \) then
2.4.1 for \( j = 1, \ldots, w \) do
2.4.2 \( f_1 \leftarrow f_1^5 \frac{L_{T,T}(Q) L_{2T,3T}(Q)}{V_{2T}(Q) V_{5T}(Q)}, T \leftarrow 5T \)
2.4.3 else
2.4.4 for \( j = 1, \ldots, u \) do
2.4.5 \( f_1 \leftarrow f_1^2 \frac{L_{T,T}(Q) L_{2T,3T}(Q)}{V_{2T}(Q) V_{5T}(Q)} \), \( T \leftarrow 2T \)
2.4.6 for \( j = 1, \ldots, v \) do
2.4.7 \( f_1 \leftarrow f_1^3 \frac{L_{T,T}(Q) L_{2T,3T}(Q)}{V_{2T}(Q) V_{5T}(Q)}, T \leftarrow 3T \)
2.4.8 for \( j = 1, \ldots, w \) do
2.4.9 \( f_1 \leftarrow f_1^5 \frac{L_{T,T}(Q) L_{2T,3T}(Q)}{V_{2T}(Q) V_{5T}(Q)}, T \leftarrow 5T \)
2.5 if \( s_{i+1} = 1 \) then \( f_1 \leftarrow f_1^1 \frac{L_{T,P}(Q)}{V_{T+P}(Q)}, T \leftarrow T + P \)
2.6 else \( f_1 \leftarrow f_1, f_{-1}^1 \frac{L_{T,P}(Q)}{V_{T-P}(Q)}, T \leftarrow T - P \)
3. return \( f_1^{(q^k-1)/l} \)

4.1. Experimental Results. To obtain the results of the above algorithm, we apply the formula for computing the Tate pairing of elliptic curves over finite fields higher than characteristic three. Numbers with 160 bit size represented with bases 2, 3 and 5 are used in Miller’s algorithm instead of binary representation. Before we list the results, we first show in Table 2 the costs of each operation used in the proposed Tate pairing computation algorithm, neglecting the cost of field addition and subtraction. We denote the cost of inversion, squaring and multiplication in \( F_q^* \) by \( I \), \( S \) and \( M \), respectively. We also denote the above costs in \( F_q^{*k} \) by \( I_k \), \( S_k \) and \( M_k \) \((M_k \approx k^{1.6} M)\), while the multiplication between \( F_q^* \) and \( F_q^{*k} \) is denoted by \( M_b = kM \), where \( k \) is an embedding degree taken as 4, 6 and 8\[16\]. To reduce the inversion operation in computing the rational function \( f \) we replace \((x_Q - x_1)^{-1}\) by \( \overline{x_Q} - x_1 \). Since \( x_Q \in F_q^* \), its conjugate can be denoted as \( \overline{x_Q} \) and \( \overline{x_1} = x_1 \in F_q^* \[16\].

In the operations for computing the rational function \( f \), we employ the same method as used by a previous work in \[16\]. For addition, subtraction, doubling and tripling, we show the list of costs required for each operation. For the parts related to our algorithm, we find the cost of TQNT. We will show the calculation in detail and compare our result with the previous algorithms mentioned above. The main formula is
\[
\begin{array}{|c|c|c|}
\hline
\text{Operation} & \text{Cost} & \text{Precomputed Cost} \\
\hline
\text{TADD} & M_k + 2.5M_b + 1I + 3M + 1S & 2M_k + 7M_{k/2} + 1I_{k/2} \\
\text{TSUB} & M_k + 1I + (2k + 3)M + 1S & 2M_k + I_k \\
\text{TDBL} & M_k + S_k + 3.5M_b + 1I + 4M + 2S & 2M_k \\
\text{TTRL} & 3M_k + S_k + 2M_b + 1I + 9M + 4S & S_k \\
\hline
\end{array}
\]

Table 2. Cost of required operations to compute Tate pairings

Let \( T = (x_1, y_1), T_2 = 2T = (x_2, y_2), T_3 = 3T = (x_3, y_3), \) and \( T_5 = 5T = (x_5, y_5) \) be in \( E(F_q) \). Notice that the line \( L_{T,T} \) passes through the points \( T \) and \(-2T \) with slope \( \lambda_1 \). Thus, we can have the equation as \((y_Q + y_2) - \lambda_1(x_Q - x_2)\). The line \( L_{T,2T} \) passes through \( T \) and \( 2T \) with slope \( \lambda_2 \). Hence, its equation is \((y_Q - y_2) - \lambda_2(x_Q - x_2)\). The other line \( L_{2T,3T} \) passes through \( 2T \) and \( 3T \) with a slope \( \lambda_3 \) in order to have its equation as \((y_Q - y_2) - \lambda_3(x_Q - x_2)\). For the vertical lines \( V_{2T}, V_{3T} \) and \( V_{5T} \) forming the denominator of the rational function \( f \), it is clear that \( V_{2T} \) passes through \( 2T \) and \(-2T \) shown as \( x_Q - x_2 \). \( V_{3T} \) passes through \( 3T \) and \(-3T \) shown as \( x_Q - x_3 \), and finally, \( V_{5T} \) passes through \( 5T \) and \(-5T \) shown as \( x_Q - x_5 \).

The TQNT algorithm can be written as follows:

\begin{enumerate}
\item \( T_5 = (x_5, y_5) \rightarrow \text{ECQNT} \)
\item \( l_1 \leftarrow \frac{L_{T,T}(Q) L_{T,2T}(Q) L_{2T,3T}(Q)}{V^2_{2T}(Q) V_{3T}(Q) V_{5T}(Q)} \)
\item \( l_2 \leftarrow \frac{L_{T,T}(Q) L_{2T,3T}(Q)}{V_{2T}(Q)} \)
\item \( l_3 \leftarrow \frac{V_{5T}(Q) V_{5T}(Q)}{} \)
\item \( f_1 \leftarrow (f_1^2)^2 \cdot f_1 \cdot l_1 \cdot l_2 \cdot l_3 \)
\item return \( f_1 \)
\end{enumerate}

We calculate the cost of \( l_1 \) in step 2 in the above algorithm, which was simplified by Chang An Zhau et al. in [16] to the following form:

\( l_1 = x_Q^2 + ((\lambda_1(\lambda_1 + \lambda_2) - 2x_1)x_Q - (\lambda_1 + \lambda_2)(y_Q - y_1) - (\lambda_1(\lambda_1 + \lambda_2) - x_1)x_1. \)

The cost of computing \( \lambda_1(\lambda_1 + \lambda_2) \) and \((\lambda_1(\lambda_1 + \lambda_2) - x_1)x_1 \) requires 2M, and computing \((\lambda_1(\lambda_1 + \lambda_2) - 2x_1)x_Q \) and \((\lambda_1 + \lambda_2)(y_Q - y_1) \) requires 2Mb. Therefore, we need 2Mb + 2M for computing \( l_1 \). Notice that \( x_Q^2 \) can be precomputed.

To compute \( l_2 \), we use the same technique to simplify the expression as follows:
\[ l_2 = x_Q^2 + ((\lambda_1(\lambda_1 + \lambda_3) - 2x_1)x_Q - (\lambda_1 + \lambda_3)(y_Q - y_1) - (\lambda_1(\lambda_1 + \lambda_3) - x_1)x_Q \]

The cost of computing \( \lambda_1(\lambda_1 + \lambda_3) \) and \( (\lambda_1(\lambda_1 + \lambda_3) - x_1)x_Q \) requires 2M,
and computing \( ((\lambda_1(\lambda_1 + \lambda_3) - 2x_1)x_Q \) and \( (\lambda_1 + \lambda_3)(y_Q - y_1) \) requires 2Mb.
Therefore, we need 2Mb + 2M for computing \( l_2 \).

To compute \( l_3 = (x_Q - x_3)(x_Q - x_5) = (x_Q - x_3)(x_Q - x_5) \)
\[ = x_Q^2 - x_Qx_5 - x_Qx_3 + x_3x_5 = x_Q^2 - x_Q(x_5 - x_3) + x_3x_5 \]
requires \( 1M_b + 1M \).

Therefore, for computing \( (f_2^1)^2 \cdot f_1 \cdot l_1 \cdot l_2 \cdot l_3 \), we require
\[ 4M_k + S_k + 5M_b + 5M \]

\( x_Q^2 \) and \( x_Q \) are already precomputed. Thus, the total cost of TQNT is
\[ 4M_k + S_k + 5M_b + 5M + \text{ECTRL} \]

### 4.2. Efficiency.

We check the efficiency of using multi-base number representation in computing Tate pairings by considering the order \( l \) of point \( P \in E(F_q) \), which has at least 160 bit length, while \( q^k \) has at least 1000 bits to be secure for use in cryptographical systems. If the total pre-computed cost is \( T_{\text{pre}} = 6M_k + S_k + I_k + 7M_{k/2} + I_{k/2} \),
then by taking \( M_4 = 9M, M_6 = 18M, M_8 = 27M, S = 0.8M, I = 10M, M_b = kM, \) and \( I_k = I + k^2 M \) [16], the total cost of our method is
\[
\begin{align*}
\text{pre} & = 6M_k + S_k + I_k + 7M_{k/2} + I_{k/2}, \\
& = (b_{\text{max}}TDBL + t_{\text{max}}TTRL + r_{\text{max}}TQNT + \frac{m}{2}(TADD + TSUB)) + T_{\text{pre}} \\
& = ((b_{\text{max}} + 3t_{\text{max}} + 4r_{\text{max}} + m + 6)M_k + ((b_{\text{max}} + t_{\text{max}} + r_{\text{max}} + 1)S_k + (2b_{\text{max}} + 2t_{\text{max}} + 5r_{\text{max}} + \frac{5}{4}m)M_b + (b_{\text{max}} + t_{\text{max}} + m)I + (4b_{\text{max}} + 9t_{\text{max}} + 5r_{\text{max}} + (k + 3)m)M) \\
& \quad + (2b_{\text{max}} + 4t_{\text{max}} + 12r_{\text{max}} + m)S + I_k + 7M_{k/2} + I_{k/2}.
\end{align*}
\]

The results show that the new algorithm is faster than the previous Miller’s algorithm using binary, signed binary and DB chain representation. The comparison is listed in the following table.
Bilinear pairing computation

The compared Algorithms | $k = 4$ | $k = 6$ | $k = 8$
--- | --- | --- | ---
compared with [19] | 35.3% | 40.1% | 42.2%
compared with [20] | 13.3% | 10.9% | 9.4%
compared with [16] | 4.5% | 2.9% | 4.0%

Table 4. Comparing the new method with some other methods in computing Tate pairings

5. Conclusion

This work proposes a new algorithm for computing Tate pairings, which has many applications in cryptographical protocols. This algorithm uses multibase representation in the scalar using 2, 3 and 5 as bases, which leads to a faster EC scalar multiplication over finite fields. Affine coordinates have been considered due to their efficiency in cryptography. Our study’s results show significant improvement in terms of cost saving in computing Tate pairing. Overall, these indicate that the new approach significantly enhances pairing-based applications. The proposed approach can be applied in computing pairings over supersingular curves and different finite fields.

Acknowledgments: The authors gratefully acknowledge the financial support received from the Malaysian Ministry of Science, Technology, and Innovation (MOSTI) under the Sciencefund Grant Project Number 01-01-04-SF0945.

References


Received: January, 2010