Implementing Pairing-Based Cryptosystems

Zhaohui Cheng and Manos Nistazakis
School of Computing Science, Middlesex University
White Hart Lane, London N17 8HR, UK.
{m.z.cheng, e.nistazakis}@mdx.ac.uk

Abstract: Pairing-based cryptosystems have been developing very fast in the last few years. As the key primitive, pairing is also the heaviest operation in these systems. The performance of pairing affects the application of the schemes in practice. In this report, we summarise the formulas of the Tate pairing operation on elliptic curves in different coordinate systems and describe a few observations of improving the pairing performance in some special cases.

1 Introduction

Starting from the proposal of using pairing in cryptography [8], especially after the work of [13] in the key exchange protocol and the first practically provable identity-based encryption scheme [1], in the last few years, the pairing-based cryptosystem has been one of most exciting fields in cryptography research. Many interesting schemes have been proposed to serve some security requirements which have never been met before. Meanwhile, due to the complexity of pairing operation, to improve the performance of the schemes is also of great concern. There are two types of pairing known so far that satisfy the special requirement needed in these cryptosystems, i.e., the Weil pairing and the Tate pairing. As the Tate pairing is about twice faster than the Weil pairing, here we only discuss the Tate pairing.

Pairing can be computed using Miller’s algorithm [16][8]. Many techniques have been exploited to dramatically improve the performance of the algorithm, such as [3][4][10][12][17][5]. Generally the following factors would affect the pairing performance, the coefficients of the elliptic curves, the Hamming weight of the subgroup orders and the (two) input points (one of them is the output of morphisms on an input of a symmetric pairing to achieve non-degeneracy in many schemes). From different angles, the pairing computation could be optimised. For example, the work in [3][4][17] mainly focus on reducing the large unit operations such as to remove the cost of denominator computation and denominator squaring in Miller’s algorithm, while the authors in [5] find that the times of Tate double-and-add operation in the algorithm could be reduced. The work of [12] tries to simplify the computation of the basic double-and-add operation, while the authors of [11] work on how to speed-up the operation of basic multiplication in the underlying field.

In this report, we summarise the formulas of the basic Tate double-and-add operation in different coordinate systems and describe a few observations of improving the pairing performance in some special cases. The report is organised as follows. First we briefly introduce the Tate pairing and Miller’s algorithm to compute pairings and the optimization so far. Then the Tate double-and-add formulas to evaluate the pairing in different coordinate systems are presented. The optimization for special cases is described and complexity of different coordinate systems is compared. Finally we discuss using the pre-computed table to speed-up the pairing in the affine system.

2 The Tate Pairing

For simplicity, here we only consider the normal cases used in the cryptosystems. For the elliptic curve $E$ over the field $\mathbb{F}_q$, let $r$ be a prime and $r \nmid \#E(\mathbb{F}_q)$. Let $E(\mathbb{F}_q)[r]$ denote the elements of $E(\mathbb{F}_q)$ of order dividing $r$ ($r$-torsion points, and if
assume that the infinity point \( O \) is an \( r \)-torsion point, then these points form a cyclic group), and let \( \mu_r = \{x \in F_{q^k} : x^r = 1\} \), where \( k \) is the smallest positive integer such that \( r \mid q^k - 1 \). \( k \) is referred as the embedding degree or MOV degree \([15]\). Then there are non-degenerate bilinear pairings such that

\[
\langle \cdot, \cdot \rangle_r : E(F_q)[r] \times E(F_q)/rE(F_q) \to F_{q^k}^*/(F_{q^k}^*)^r
\]

and

\[
\tau_r : E(F_q)[r] \times E(F_q)/rE(F_q) \to \mu_r
\]

\( \langle \cdot, \cdot \rangle_r \) is called the Tate-Lichtenbaum pairing and \( \tau_r \) is called the modified Tate-Lichtenbaum pairing and we have \( \tau_r(P, Q) = \langle P, Q \rangle_{r}^{q^k-1} \) \([8][20]\). If \( k > 1 \) (i.e., \( r \mid q - 1 \)), then \( \tau_r(P, Q) = \langle (P, Q)_{(q-1)/d} \rangle \) where \( (q-1)/d = q^{k-1} \). The modified Tate pairing satisfies the following properties, (so does the Tate pairing with the corresponding ones).

1. (bilinearity) \( \tau_r(P_1 + P_2, Q) = \tau_r(P_1, Q) + \tau_r(P_2, Q) \) and \( \tau_r(P, Q_1 + Q_2) = \tau_r(P, Q_1) + \tau_r(P, Q_2) \).

2. (non-degeneracy) For each \( P \neq O \in E(F_q)[r] \), there exists \( Q \in E(F_q) \) such that \( \tau_r(P, Q) \neq 1 \).

The Tate pairing \( \langle P, Q \rangle_r \) can be constructed as follows. First find the function \( f_P \) whose divisor (see \([8]\) for details of divisor) is equal to \( r[P] - r[O] \) and then compute a divisor \( D_Q \) which is equivalent to \([Q] - [O] \) and the support of \( D_Q \) does not include \( P \) and \( O \). This can be done with great success probability by randomly choosing \( S \in E(F_q) \) and setting \( D_Q = [Q + S] - [S] \). \( \langle P, Q \rangle_r \) is defined as

\[
\langle P, Q \rangle_r = f_P(D_Q) = f_P(Q + S)/f_P(S).
\]

The Tate pairing can be computed using Miller’s algorithm. Miller’s algorithm basically uses the double-and-add algorithm similar to the point scalar and we call it the Tate double-and-add operation. We need to compute the function whose divisor is equal to \( D_P = r[P + R] - r[R] \) (\( R \) could be \( O \)). Let \( D_P = j[P + R] - j[R] - [jP] + [O] \). Because \( \text{sum}(D_P) = O \) and \( \deg(D_P) = 0 \), there exists a function \( f_j \) whose divisor is \( D_P \). Note that

\[
D_{j+k} = (j+k)[P + R] - (j+k)[R] - [(j+k)P] + [O] = D_j + D_k + \text{div}(2ax + by + c)/x^d,
\]

where \( ax + by + c = 0 \) is the line through \( jP \) and \( kP \) (the tangent line if \( jP = kP \)), and \( x + d = 0 \) is the vertical line through \((j+k)P\). The double-and-add algorithm works as follows. Let \( r \)’s binary representation be \( r = (r_t, r_{t-1}, \ldots, r_0) \). Let \( g_{v,P} \) be the straight line through point \( V \) and \( P \) (the tangent line of the curve, if \( V = P \)), and let \( g_{v,P} \) be the vertical line through point \( V \).

**INPUT:** A prime integer \( r \) and two points \( P \in E(F_q)[r] \) and \( Q \in E(F_q) \).

**OUTPUT:** The Tate pairing \( \langle P, Q \rangle_r \).

1. Randomly choose a point \( S = (S_x, S_y) \in E(F_q) \) and set \( Q' = Q + S = (Q_x', Q_y') \).

2. set \( f \leftarrow 1 \) and \( V \leftarrow P \).

3. for \((i = t - 1; i >= 0; i--)\)

   (a) set \( f \leftarrow f^2 \cdot \frac{g_{v,P}(Q_y',Q_x')}{g_{v,P}(S_y,S_x)} \cdot \frac{g_{v,P}(S_x,S_y)}{g_{v,P}(Q_y',Q_x')} \) and \( V \leftarrow 2V \).

   (b) if \((r_i == 1)\) then set \( f \leftarrow f \cdot \frac{g_{v,P}(Q_y',Q_x')}{g_{v,P}(S_y,S_x)} \) and \( V \leftarrow V + P \).

4. return \( f \).

Note that there is no unique value for each Tate pairing (instead, the pairing’s value is a coset), while in the cryptosystems, normally we need a unique representation of the final result in each computation. Hence, in cryptography we normally use the modified Tate pairing, i.e., \( \tau_r(P, Q) = \langle P, Q \rangle_r^{q^k-1} \). As it is more convenient to use two points from the same group in many cryptosystems, the symmetric pairing with a morphism is introduced. A symmetric Tate pairing \( \tau_{\phi} \) is defined as, for \( P, T \in E(F_q)[r] \subseteq E(F_q)[r] \), \( \tau_{\phi}(P, T) = \tau_{\phi}(P, \phi(T)) = \tau_r(P, Q), \) where a non-rational endomorphism \( \phi(T) \) is required to map \( T \) to \( Q \) that is linearly independent from \( P \). This is required to achieve non-degeneracy if \( [a]P + [b]Q = O \) where \( a, b \neq 0 \in \mathbb{Z}_q^* \), then \( \tau_{\phi}(P, Q) = 1 \) if \( k > 1 \), see \([9]\)
Moreover, we can speed-up Miller’s algorithm in this setting in a few ways.

1. In [3][4], it has been demonstrated that if \( P \) and \( Q \) are linearly independent, then 
   
   \[
   \tau_r(P, Q) = f_P(Q) \frac{\Delta}{\delta} \quad \text{(i.e., choose } S = O^1). 
   \]
   
   And the evaluations of \( g_{2V}(S_x), g_{V'}(S_x, S_y), g_{V+P}(S_x) \) and \( g_P(S_x, S_y) \) become unnecessary.

2. Intuitively, when \( k = 2 \) and \( q \) is a large prime, to evaluate \( \tau_r(P, T) = \tau_r(P, \phi(T)) \), if we can find a morphism \( \phi(T) = (f_1(T_x), f_2(T_y)) = (Q_x, Q_y) \) for some functions \( f_1, f_2 \) such that \( f_1(T_x) \in F_q \), then the denominators \( g_{2V}(Q_x) \) and \( g_{V+P}(Q_x) \) become irrelevant in the computation because after applying the power \( (q^k - 1)/r \), these results are diminished to 1. And this can be generalized as that if \( f_1(T_x) \in F_q^{d'} \) where positive integer \( d' \) and \( d < k \), then the denominators \( g_{2V}(Q_x) \) and \( g_{V+P}(Q_x) \) become irrelevant [4].

   Note that \( f_1(T_x) \) is not necessarily in \( F_q \) for different \( k \) and \( d \). For supersingular curves, this type of endomorphism is presented in [3][17].

   And for normal curves, by carefully selecting the group parameter, an asymmetric pairing \( \tau_r(P, Q) \) that \( P, Q \) are not in the same torsion group can be sped up in a similar way [4].

3. Another observation in the algorithm is that by choosing a prime \( r \), e.g. a Solinas prime, with low Hamming weight, the times of invocation of Step 3.b in the algorithm can be reduced. As noticed in [7], the very last addition (Step 3.b) is not necessary. Similarly, if \( p \) is chosen as a generalized Mersenne prime which has low Hamming weight, the modulo operation can be done faster. While we should be careful with the possible side-effect that some prime \( p \) with the special form could be vulnerable to the faster number field sieve.

4. As pointed in [3], the exponentiation in the modified Tate pairing can be sped up as well. Specially if \( k = 2 \) and the normal basis is used, the Lucas ladder can be used [18].

By applying these techniques, the pairing can be computed dramatically faster than using the general Miller’s algorithm.

In [5], the authors found that the general Miller’s algorithm can be improved by reducing the iterations of double-and-add operations. However, this improvement appears to has significance only if the denominator has to be computed.

Finally, as the modified Tate pairing needs an exponentiation operation. When the security level is higher, the cost of this operation becomes paramount. Then we can use Weil pairing instead in some cases to achieve better performance if pre-computation is not feasible.

In this report, we summarise how to perform the double-and-add operation on (supersingular) elliptic curves in different coordinate systems and describe how to optimise this operation in some special cases.

3 Tate Double-And-Add in Coordinate Systems

For simplicity, in the report we only consider the following curve which is the favorite choice in the software implementation of pairing-based cryptosystems. Suppose the elliptic curve \( E \) is defined over the field \( F_q \) where \( q \) is a large prime. Let \( r \) be a prime of 160-bit and \( r | \#E(F_q) \) and a point \( P \in E(F_q)[r] \).

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

Note that \( E \) is not necessarily supersingular.

Miller’s algorithm includes the following operations: the point doubling \( (V = 2V) \), the point addition with a point fixed \( (V = V + P) \), the tangent-line evaluation \( (g_{V'}(Q_x, Q_y)) \), the non-tangent non-vertical line evaluation \( (g_{V,P}(Q_x, Q_y)) \) and probably the vertical line evaluation \( (g_{2V}(Q_x)) \). In the literature, the point addition and doubling algorithms have been investigated with great effort to improve performance, see [6]. Based on the work on point addition and doubling, the Tate double-and-add computation can be optimised as a whole in the
3.1 Addition formulas in affine coordinate

- **Point doubling formulas.** For a point \( V = (x_1, y_1) \), \( 2V = (x_3, y_3) \) is computed by

\[
\lambda = \frac{3x_1^2 + a}{2y_1} \quad (3.1)
\]

\[
x_3 = \lambda^2 - 2x_1 \quad (3.2)
\]

\[
y_3 = \lambda(x_1 - x_3) - y_1 \quad (3.3)
\]

- **Tangent line evaluation formula.** For points \( V = (x_1, y_1) \) and \( Q = (Q_x, Q_y) \), \( gV, V(Q_x, Q_y) \) is evaluated by

\[
u = (Q_y - y_1) - \lambda(Q_x - x_1) \quad (3.4)
\]

- **Point addition formulas.** For two points \( P = (x_1, y_1) \) and \( V = (x_2, y_2) \), \( P + V = R = (x_3, y_3) \) is computed by

\[
\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad (3.5)
\]

\[
x_3 = \lambda^2 - x_1 - x_2 \quad (3.6)
\]

\[
y_3 = \lambda(x_1 - x_3) - y_1 \quad (3.7)
\]

- **Non-tangent-non-vertical line evaluation formula.** For points \( P = (x_1, y_1) \), \( V = (x_2, y_2) \) and \( Q = (Q_x, Q_y) \), \( gV, P(Q_x, Q_y) \) is evaluated by

\[
u = (Q_y - y_1) - \lambda(Q_x - x_1) \quad (3.8)
\]

- **Vertical line evaluation formula.** For points \( V = (x_3, y_3) \) and \( Q = (Q_x, Q_y) \), \( gV(Q_x) \) is computed as

\[
v = Q_x - x_3 \quad (3.9)
\]

- **Numerator squaring.**

\[
g_i = g_{i-1}^2 \cdot u_i \quad i > 1 \quad (3.10)
\]

\[
g_0 = 1
\]

- **Denominator squaring.**

\[
h_i = h_{i-1}^2 \cdot v_i \quad i > 1 \quad (3.11)
\]

\[
h_0 = 1
\]

In the affine system, in each iteration of double-and-add operation, one modulo inverse operation is required, which is very computationally expensive compared with multiplication. Hence, some projective coordinate systems are developed to remove the inverse operation by using more multiplications instead. Two major coordinate systems, the Jacobian and the modified Jacobian system, have better performance on the point doubling operation.

In the sequel, we describe the Tate double-and-add operation in these two systems by considering the choice of coefficient of the curve and the group order \( r \).

3.2 Addition formulas in Jacobian coordinate

For Jacobian coordinates, denoted by \( J^1 \), by applying the variable changes \( x = X/Z^2 \) and \( y = Y/Z^3 \), the new curve equation is

\[
E : Y^2 = X^3 + aXZ^4 + bZ^6
\]

In the following computation, we assume that \( a = 0 \) or is small.

- **Point doubling formula-I.** Let \( V = (X_1, Y_1, Z_1) \). \( 2V = R = (X_3, Y_3, Z_3) \) is computed as follows.

\[
\lambda_1 = Z_1^2 \\
\lambda_2 = 3X_1^2 + a\lambda_1^2 \\
Z_3 = 2Y_1Z_1 \\
\lambda_3 = 2Y_1^2 \\
\lambda_4 = 2X_1\lambda_3 \\
X_3 = \lambda_2 - 2\lambda_4 \\
\lambda_5 = 2\lambda_3^2 \\
Y_3 = \lambda_2(\lambda_4 - X_3) - \lambda_5
\]

\[
= \frac{1}{3M + 6S}
\]

In the report, \( M \) stands for multiplication operation, while \( S \) means squaring. Note that if \( a = 0 \), the cost is \( 3M + 4S \) by saving the cost of computing \( \lambda_1 \) and \( \lambda_2^2 \).

- **Tangent line evaluation formula-I.** The counterpart of equation 3.4 in the Jacobian system is

\[
u = \frac{Q_y}{Z_1^2} - \frac{\lambda_2}{Z_3}(Q_x - \frac{X_1}{Z_1}) \quad (3.12)
\]

Because \( Z_1, Z_3 \in F_q \), after applying the power \( q - 1 \) on \( u, Z_1^2Z_3 \) (the least common divisor of \( Z_1^2 \) and \( Z_3 \))
will be diminished as 1, we can convert the above formula to

\[ u = (Z_3Z_4^2Q_y - 2Y_1^2) - \lambda_2(Z_1^2Q_x - X_1) \]
\[ = (Z_3Z_4^2Q_y - \lambda_3) - \lambda_2(Z_1^2Q_x - X_1) \]

(3.13)

Note that in the morphism \( \phi(T) \), \( f_1(T_x) \) is not necessary in \( F_q \), i.e., \( Q_a \) in formula 3.13 is not necessary in \( F_q \). If \( Q_a \in F_q^* \) is represented using the normal basis \( \lambda_3(\lambda_1Q_x - X_1) \) in the formula needs \( k + 2 \) multiplications in \( F_q \). This evaluation can be simplified if \( a = 0 \). When \( a = 0 \), we can compute the point doubling and the tangent line evaluation in another way. We denote the Jacobian coordinate system using the following point doubling formula-II as \( J^I \).

- **Point doubling formula-II.** Let \( V = (X_1, Y_1, Z_1) \). \( 2V = R = (X_2, Y_2, Z_2) \) is computed as follows.

\[
\begin{align*}
\lambda_1 &= 3X_1^2 & 1S \\
\lambda_2 &= \lambda_1X_1 & 1M \\
\lambda_3 &= 2Y_1^2 & 1S \\
\lambda_4 &= 2\lambda_3^2 & 1S \\
X_3 &= X_1(3\lambda_2 - 4\lambda_3) & 1M \\
Y_3 &= \lambda_2(6\lambda_3 - 3\lambda_2) - \lambda_4 & 1M \\
Z_3 &= 2Y_1Z_1 & 1M \\
&= 4M + 3S
\end{align*}
\]

- **Tangent line evaluation formula-II.** Now the evaluation can make use of the intermediate result \( \lambda_2 \) to save one multiplication as

\[
\begin{align*}
u &= (Z_3Z_4^2Q_y - 2Y_1^2) - \lambda_1(Z_1^2Q_x - X_1) \\
&= (Z_3Z_4^2Q_y - \lambda_3) - (\lambda_1Z_1^2Q_x - \lambda_2) \\
\end{align*}
\]

(3.14)

- **Curve addition formula.** Let \( P = (X_1, Y_1, Z_1) \) and \( V = (X_2, Y_2, Z_2) \). Note that it always holds that \( Z_1 = 1 \). \( P + V = R = (X_3, Y_3, Z_3) \) is computed by

\[
\begin{align*}
\lambda_1 &= X_1Z_2^2 & 1M + 1S \\
\lambda_2 &= \lambda_1 - X_2 & 2M \\
\lambda_3 &= Y_1Z_2^2 & 2M \\
\lambda_4 &= \lambda_3 - Y_2 & 2M \\
\lambda_5 &= \lambda_1 + X_2 & 2M \\
\lambda_6 &= \lambda_3 + Y_2 & 2M \\
\lambda_7 &= \lambda_3^2 & 1S \\
X_3 &= \lambda_2^2 - \lambda_5\lambda_7 & 1M + 1S \\
Y_3 &= (\lambda_8\lambda_4 - \lambda_6\lambda_7\lambda_2)/2 & 3M \quad (3.15) \\
&= 8M + 3S
\end{align*}
\]

- **Non-tangent-non-vertical line evaluation formula.** Because \( x_1 \) (resp. \( y_1 \)) is always the \( x \)-coordinate (resp. \( y \)-coordinate) of point \( P \) and the corresponding coordinates in the Jacobian system is \((X_1, Y_1, 1)\), we can compute the counterpart of equation 3.8 as

\[ u = Z_3(Q_y - Y_1) - \lambda_4(Q_x - X_1) \]

(3.15)

- **Vertical line evaluation formula.** For the same reason, the counterpart of equation 3.9 in the Jacobian coordinate system is

\[ v = Z_3^3Q_x - X_3 \]

(3.16)

As noticed in [12] that \( Z_3^3 \) is used both in the vertical line evaluation and the point addition (the point doubling and the tangent line evaluation as well if \( a \neq 0 \)), one squaring operation can be saved by passing this intermediate result cross the consecutive Tate double-and-add computations if the vertical line evaluation is required. So in the new coordinate system, a point is presented as \((X, Y, Z, Z^2)\), and we denote it as \( J^2 \). If \( a = 0 \), we can combine the coordinate system \( J^2 \) and the doubling formula-II, denoted by \( J^3 \), to provide better performance in the case that variable \( Q_x \) in the evaluation formulas is not an element of \( F_q \).

### 3.3 Addition formulas in the modified Jacobian coordinate

If the coefficient \( a \) is a large integer in \( F_q \), we have to consider the cost of computing \( a\lambda_1^2 \) in the doubling formula of the Jacobian system. As demonstrated in [6], the modified Jacobian coordinate system, denoted by \( J^3 \), can save two squaring operations to double a point when \( a \) is large. Although,
the general point addition in the modified Jacobian system is more complex compared with the Jacobian system, if the point addition is only needed to be computed with trivial times, the modified Jacobian system could profit the pairing computation. By choosing special scalars $r$ with low Hamming weights, e.g., the Solinas prime, the general point addition could become trivial in the pairing computation. A point in the system is presented as $(X, Y, Z, aZ^3)$.

- **Point doubling formula.** Let $V = (X_1, Y_1, Z_1, aZ_1^3)$. 2$V = R = (X_3, Y_3, Z_3, aZ_3^3)$ is computed as follows.

\[
\begin{align*}
\lambda_1 &= 3X_1^2 + aZ_1^4 && 1S \\
\lambda_2 &= 2Y_1^2 && 1S \\
\lambda_3 &= 2X_1\lambda_2 && 1M \\
X_3 &= \lambda_1^2 - 2\lambda_3 && 1S \\
\lambda_4 &= 2\lambda_2^2 && 1S \\
Y_3 &= \lambda_1(\lambda_3 - X_3) - \lambda_4 && 1M \\
Z_3 &= 2Y_1Z_1 && 1M \\
aZ_3^3 &= 2a\lambda(aZ_1^4) && 1M \\
\end{align*}
\]

- **Tangent line evaluation formula.** Now the tangent line evaluation can be computed by

\[
\begin{align*}
u &= (Z_3Z_1^2Q_y - 2Y_1^2) - \lambda_1(Z_1^2Q_x - X_1) \\
&= (Z_3Z_1^2Q_y - \lambda_2) - \lambda_1(Z_1^2Q_y - X_1) \\
&= (Z_3Z_1^2Q_y - \lambda_3 - X_3) - \lambda_4 \\
&= (Z_3Z_1^2Q_y - \lambda_3 - X_3 - \lambda_4) \\
\end{align*}
\]  

\[
\lambda_1 = X_1Z_1^2 \\
\lambda_2 = 1 - X_2 \\
\lambda_3 = Y_1Z_1^2 \\
\lambda_4 = \lambda_3 - Y_2 \\
\lambda_5 = \lambda_1 + X_2 \\
\lambda_6 = \lambda_3 + Y_2 \\
Z_3 = Z_2\lambda_2 \\
\lambda_7 = \lambda_2^2 \\
X_3 = \lambda_3^2 - \lambda_5\lambda_7 \\
\lambda_8 = \lambda_3\lambda_7 - 2X_3 \\
Y_3 = (\lambda_8\lambda_4 - \lambda_6\lambda_7\lambda_2)/2 \\
\lambda_9 = Z_3^4 \\
aZ_3^3 = a\lambda_9^3 \\
\]

The non-tangent-non-vertical line evaluation formula (resp. the vertical line evaluation formula) is same as formula 3.15 (resp. formula 3.16).

### 3.4 Computation Complexity Without Pre-computation

Assume the normal basis is used and at least one of the coordinates $(Q_x, Q_y)$ of $Q$ has to be represented by $k$ elements in $F_q$. For the result of [3][4][17], here we assume that the vertical line evaluation is not necessary. Hence we do not consider the advantage of using the coordinate systems $J^2$ and $J^3$. Without considering the cost of squaring the numerator, Table 1 summarises the computation complexity of curves based on different value for coefficient $\alpha$, different output of the morphism applied on one input of the pairing and the Hamming weight of the subgroup order $r$. In the table, HHW stands for High Hamming Weight (HHW), while LHW means Low Hamming Weight. From Table 1, we can find that the curve $E : y^2 = x^3 + b$ with the morphism $\phi(T) = Q$ that $Q_x \in F_q$, has the best implementation performance to evaluate the Tate pairing $\hat{\tau}_r(P, T) = \tau_r(P, \phi(T)) = \tau_r(P, Q)$ in the (variant) coordinate systems without pre-computing. More general condition could be that $Q_x \in F_{q^k}$ that $d \mid k$ when $P \in E(F_{q^k})[r]$. If $Q_x \notin F_q$ and coefficient $a = 0$, we can use $J^1\tau$ to save one squaring in each double-and-add iteration compared with using $J^1\tau$. When coefficient $a$ of the curve is large and the used torsion group order $r$ has low Hamming weight, e.g., a Solinas prime of the form $r = 2^s + 2^p + 1$, the modified Jacobian system could be used to save a squaring in each iteration compared with using $J^1\tau$.

The significance of these improvements depends on the embedding degree and the Hamming weight of $r$. For example, if $k = 2$ and $r$ is a Solinas prime (hence the cost of Step 3.b in Miller’s algorithm is trivial), and assume the cost ratio of $S/M$ is 0.85, then the saving is about 5% (resp. 4.5%) using $J^1\tau$ (resp. $J^3\tau$) compared with using $J^1\tau$, where one squaring and one multiplication in $F_q$ to compute the numeration squaring are counted.

### 4 Table-based Implementation

Although pairing in the projective systems is relatively faster than in the affine system if the in-
version is required in each iteration of the double-
and-add operation, the cost is still very high. An
observation of the pairing-based system is that in
many cases the scalar \( r \) and one point of the input
are fixed in Miller’s algorithm and we can make use
of this attribute to pre-compute some intermediate
results to speed-up the whole process. For exam-
ple, in the affine system, we can pre-compute \( \lambda \)’s
and if the memory space is sufficient, the point ad-
dition (or doubling) result in each step and (or) the
coefficients of the lines as well and store them into
a table. Now if \( Q_x \in F_q \), by retrieving the required
data from the table, each iteration only needs at
most \( kM \) in the underlying field. This would signifi-
cantly improve the algorithm’s performance. If the
memory is limited, by only pre-computing and stor-
ing \( \lambda \)’s, the algorithm needs at most \((k + 1)M + 1S\)
to compute each numerator and the table size is
about \( r \log q \) bits if \( r \) has very low Hamming weight,
such as in the form \( r = 2^\alpha + 2^\beta + 1 \) (for the form
\( r = 2^\alpha - 2^\beta - 1 \), we can use the signed represen-
tation of \( r \).) Note that \( \hat{\tau}(P, Q) = \hat{\tau}(Q, P) \) for
\( P, Q \in E(F_q)[r] \) if the endomorphism is applied.
Hence, if the algorithm is implemented to support
the pre-computed table of the first input of a pair-
ing, we can simply exchange the inputs.

5 Concluding Remarks

It is important to improve the performance of pair-
ing to promote the wide deployment of pairing-
based cryptosystems. In the report, the formu-
las to evaluate the Tate double-and-add operation
on elliptic curves in different coordinate systems
are summarised and a few observations to speed-
up pairing in special cases are described. Latest
work [7][2][14] shows that an even faster algorithm
on small characteristic (2 and 3) supersingular el-
liptic curves and hyperelliptic curves exists.

References

[1] D. Boneh and M. Franklin. “Identity Based En-
cryption from The Weil Pairing”, Advances in
heigeartaigh and M. Scott. “Efficient Pair-
ing Computation on Supersingular Abelian Va-
rieties”, Cryptology ePrint Archive, Report
2004/375.
Scott. “Efficient Algorithms for Pairing-Based
Cryptosystems”, Advances in Cryptology –
Crypto 2002, Lecture Notes on Computer Sci-
“On the Selection of Pairing-Friendly Groups”,
Selected Areas in Cryptography – SAC 2003,
Lecture Notes on Computer Science 3006, pp.
of Miller’s Algorithm for Computing Weil/Tate
Pairing”, Cryptology ePrint Archive, Report
2004/065.


