Comparing Implementation Efficiency of Ordinary and Squared Pairings

Christine Abegail Antonio[1], Satoru [2] and Ken Nakamula[3]
Department of Mathematics, Tokyo Metropolitan University
Minami-Osawa, Hachioji-shi
Tokyo, Japan
[1] abby@tnt.math.metro-u.ac.jp
[2] satoru@tnt.math.metro-u.ac.jp
[3] nakamula@tnt.math.metro-u.ac.jp

Abstract

In this paper, we will implement a standard probabilistic method of computing bilinear pairings. We will compare its performance to a deterministic algorithm introduced in [5] to compute the *squared Tate/Weil* pairings which are claimed to be 20 percent faster than the standard method. All pairings will be evaluated over pairing-friendly ordinary elliptic curves of embedding degrees 8 and 10 and a supersingular curve of embedding degree 6. For these curves, we can make the algorithm to compute both the ordinary Weil and Tate pairings deterministic and optimizations to improve the algorithms are applied. We will show that the evaluation of *squared Weil* pairing is, indeed, faster than the ordinary Weil pairing even with optimizations. However, evaluation of the *squared Tate* pairing is not faster than the ordinary Tate pairing over the curves that we used when optimizations are applied.

key words and phrases. bilinear pairings, squared Weil/Tate pairing, cryptography, pairing-friendly curves

1 Introduction

Recently, increasing interest has been focused on the efficient computations of bilinear pairings for cryptographic use. Two of the most researched type of bilinear pairings are the Weil and the Tate pairings. The original polynomial time algorithm to evaluate these pairings was due to Miller [13] for the case of elliptic curves. Since the algorithm is probabilistic, a lot of researches are based on improving his method, e.g., making it deterministic. The papers [10] and [1] proposed a fast computation of the Tate pairing by improving Miller's

algorithm. In [11], they suggested that bilinear pairing computation can be improved by choosing suitable fields where the field arithmetic can be made relatively faster.

Furthermore, the security of pairing based cryptosystems depends on (1) finding curves whose order is divisible by a large prime such that generic attacks on smooth group orders (Pohlig-Hellman attacks) can be resisted, and on (2) the discrete logarithm in the field where the evaluation of Tate/Weil pairing lies should be computationally infeasible. At the same time, such field should lie in a 'low' degree field so that pairings are easily computable. Pairing-friendly elliptic curves address these problems.

A randomly chosen elliptic curve tends to have a large embedding degree which is not useful in pairing based cryptosystems. One type of curves that is proven to be effective is supersingular curves. They are guaranteed to have small embedding degree ($k \leq 6$) and Menezes, Okamoto and Vanstone [12] showed a complete classification for these curves. In [17] and [18], Tanaka, proposed a method of constructing an ordinary (nonsupersingular) pairing-friendly elliptic curves with embedding degree 8 by improving the technique proposed by Brezing and Weng which can be found in [6]. In [7], Freeman provided a technique on constructing ordinary pairing-friendly curves of prime order with embedding degree 10.

A comparison of the performance of the Tate pairing over MNT curves [14] and supersingular curves are discussed by D. Page et al. in [15]. In [4], they proposed an efficient implementation of the Tate pairing over Barreto-Naehrig curves. In this paper, using supersingular elliptic curves over a field of characteristic 3 and the curves generated by Freeman and Tanaka, we will implement the basic algorithm of Miller to compute bilinear pairings. We will compare its performance to a deterministic algorithm to compute the *squared Tate/Weil* pairing which was proposed in [5]. We will show that even by adding optimizations on the standard algorithm, the squared Weil pairing is much faster than the Weil pairing. However, the Tate pairing is comparable to the squared Tate pairing when optimizations are applied and details of this fact are discussed in Section 2.2.

This paper is organized as follows. Section 2 gives the notations that we will use in this paper, a brief mathematical background of bilinear pairings, in particular, the Tate/Weil pairings and the squared Tate/Weil pairings and the algorithms to compute them. We will also introduce some optimizations to make the algorithms efficient. Section 3 will give details on pairing friendly elliptic curves that we will use in our implementations. In Section 4 we will give our numerical results. Section 5 gives the observations we draw from this experiment.

Our Contribution

We will compare the computational efficiency of pairing evaluations on ordinary curves of embedding degrees 8 and 10 over large prime fields and on a supersingular curve of embedding degree 6 over a field of characteristic 3. We

will focus on the optimizations that can be applied to the standard algorithm, particularly on Miller's loop and the use of distortion maps for the supersingular case.

We will show a method on how the standard algorithms can also be made deterministic under a natural setting, i.e., since we are implementing over curves of embedding degree greater than 1, we can choose one of our points to come from $E(\mathbb{F}_q)$ and the other point to come from $E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. This assures us that we will not get a trivial value in our pairing computations.

The parameters that we will use for the pairing-friendly curves will be presented. We chose ordinary curves of embedding degrees 8 and 10 because as of the time this paper is being written, there are still no published works on the comparison of the implementations of pairings on these embedding degrees. As for our choice of supersingular curves, a lot of researches have shown that the best choices for parameters are in characteristic three, e.g. [9] and [11].

Our main objective is to be able to compare the implementation efficiency of squared pairings and ordinary pairings over curves of embedding degrees 8 and 10 and supersingular curves of embedding degree 6. However, with this experiment, we were able to conclude based on our results, that the squared Tate pairing is not necessarily faster than the ordinary Tate pairing with optimizations.

2 Notations and Algorithms

Let E be an elliptic curve over a finite field $\mathbb{F}_q$. A *divisor* D is defined as $D = \sum_{P \in E} m_P(P)$, $m_P \in \mathbb{Z}$ and $m_P = 0$ for almost all points P. These divisors form an additive group $\mathbf{D}$(E). We say that a divisor D is principal if there exists a rational function f such that m_P gives the order of vanishing of f at P. These principal divisors are of degree 0 and form a subgroup of the group $\mathbf{D}^0$(E) of degree zero divisors. The support of a divisor $D = \sum_{P \in E} n_P(P)$, denoted by supp($D$), is the set of points P with $n_P \neq 0$. See [8] for a detailed discussion on divisors.

2.1 The Weil and the Tate Pairings

Let n=$|E(\mathbb{F}_q)|$ and $r \in \mathbb{Z}^+$ such that r is relatively prime to the characteristic of the field $\mathbb{F}_q$. We denote by k the *embedding degree* or the *security multiplier*, i.e., the smallest positive integer such that $r|q^k - 1$. A pairing is a function which maps bilinearly, a pair of elliptic curve points P, Q to an element in the finite multiplicative group ${\mathbb{F}_{q^k}}^*$. Two of the most commonly used pairings are the Weil and the Tate pairings.

We denote by $E(\mathbb{F}_{q^k})[r]$ the group of r-torsion points of an elliptic curve and let $\mu_r = \{x \in \mathbb{F}_{q^k} | x^r = 1\}$, or the r-th roots of unity in $\mathbb{F}_{q^k}$. For a pair of r-torsion points P, Q, we let f_P and f_Q be rational functions with divisors $(f_P) = rD_P$ and $(f_Q) = rD_Q$, where $D_P \sim (P) - (P_\infty)$, $D_Q \sim (Q) - (P_\infty)$ with

$\mathrm{supp}(D_Q) \cap \mathrm{supp}(D_P) = \emptyset$ and P_∞ denotes the point at infinity on E. The Weil pairing is a map

$$\begin{array}{rcccc} e_r(*,*) & : & E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] & \rightarrow & \mu_r. \\ & & (P,Q) & \mapsto & f_P(D_Q)/f_Q(D_P) \end{array}$$

On the other hand, let the support of D_Q be disjoint from the support of (f_P). Choose an $S \in E(\mathbb{F}_q)$ such that $S \notin \langle P \rangle \cup \langle P - Q \rangle \cup \langle -Q \rangle$ and let $D_{Q'} = (Q + S) - (S)$. Then the Tate pairing is a map

$$\begin{array}{rcccc} < *, * >_r & : & E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) & \rightarrow & {\mathbb{F}_{q^k}}^*/{\mathbb{F}_{q^k}}^{*r} \\ & & (P,Q) & \mapsto & f_P(D_{Q'}) \end{array}$$

The value of the Tate pairing is independent of the choice of S.

Note that Tate pairing is not unique because it evaluates as an element of one of the cosets of ${\mathbb{F}_{q^k}}^*/{\mathbb{F}_{q^k}}^{*r}$. To produce a unique value, we raise the output to the power $(q^k - 1)/r$. This process is called the *final exponentiation*.

The Weil pairing requires two applications of Miller's algorithm to produce a bilinear map, whereas the Tate pairing requires only one application and a final exponentiation. Since the Tate pairing is less expensive to implement, it is much commonly used in practice. The disadvantage to these standard algorithms is that it may fail for random choices of P and Q because it may happen that the tangent line may pass through Q, or the vertical function may be evaluated to be zero. In the next subsection, we will present an optimization where we can make both algorithms deterministic.

2.2 Optimizations

The first known technique to compute both pairings was due to Miller [13]. In this paper, we added a few optimizations in the basic Miller algorithm in our programs and they are as follows. See [8] and [10] for efficient computation of pairings.

1. To avoid division in $\mathbb{F}_{q^k}$, we can replace the rational function f in Miller's loop by f_1/f_2, where f_1 and f_2 correspond to the numerator and the denominator of f, respectively. We can perform a single division at the end of the loop.

2. Recall that we are evaluating pairings over $\mathbb{F}_{q^k}$. For embedding degree $k > 1$, we can choose one of our points to come from $E(\mathbb{F}_q)$ and the other point to come from $E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. Under this natural setting, we are sure that we will not get a trivial value in our pairing computations which will make our algorithm deterministic.

3. For the case of the original Weil pairing, for $k > 1$ and r is prime, we choose $(f_P) = r(P) - r(P_\infty)$ and $(f_Q) = r(Q) - r(P_\infty)$ and we choose 2 random points S, T and equate $D_P = (P + S) - (S)$, $D_Q = (Q + T) - (T)$. To optimize, we choose $P \in E(\mathbb{F}_q)$, $P \neq P_\infty$ and $Q \in E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. With this choice, $E(\mathbb{F}_{q^k})[r] = \langle P, Q \rangle$. So we choose $R \in \{aP - bQ | 1 < a, b < r\}$ and

take $D_Q = (Q + R) - (R)$. Then $\text{supp}(f_P) \cap \text{supp}(D_Q) = \emptyset$ which is what we want. Similarly, we choose $D_P = (P - R) - (-R)$ and we can easily check that $\text{supp}(f_Q) \cap \text{supp}(D_P) = \emptyset$. Therefore, we only need to choose a single point R so that we will not get a trivial value in our computations.

4. For the case of the Tate pairing, if $k > 1$ and r is prime, then r does not divide $q - 1$ so all elements of $\mathbb{F}_q^*$ map to 1 when we raise it to $(q^k - 1)/r$. Therefore we can ignore all terms in the algorithm that evaluates at $\mathbb{F}_q^*$. For example, to compute the Tate pairing in Algorithm IX.1 of [8], it is unnecessary to choose a random point S in the standard algorithm since l, v and S are defined over $\mathbb{F}_q$. Therefore, there will be no need to compute for $l(S)$ and $v(S)$ in Miller's loop of the original algorithm because it maps to 1 after the final exponentiation [8]. In fact, there is no need to choose a random point S at all. This will make the program run faster.

5. The use of distortion maps for supersingular curves appear to speed up pairing evaluations, therefore, in our experiment, we will compare its performance with and without distortion maps. Details on the distortion map of the curve that we will use in our implementations are discussed in Section 3.

In the algorithms, we will use the functions $l_{A,B}$ and v_{A+B}. These are just the lines computed when evaluating the elliptic curve point addition $A+B = C$. The values for these functions are solved using the formulas

$$l_{A,B}(Q) = (y_Q - y_A) - \lambda(x_Q - x_A)$$

and

$$v_C(Q) = (x_Q - x_C)$$

where $A=(x_A, y_A)$, $C=(x_C, y_C)$, $Q=(x_Q, y_Q)$, and λ is the slope of the line through A and B. Note that we first need to compute $A + B$ and from this, we can obtain the slope of the line, λ, through A and B. Notice that to compute $l_{A,B}(Q)$ we need 1 multiplication while no multiplication is needed to evaluate $v_C(Q)$. Furthermore, let

$$r = (1, r_d, ..., r_0)_2, \qquad d = \lfloor \log_2 r \rfloor - 1$$

be the binary expansion of r.

Below are the algorithms to compute the Weil and Tate pairings with optimizations described as in 1-5 above.

Algorithm 1. Optimized Miller's Algorithm for Computing Weil Pairing.

INPUT: $P \in E(\mathbb{F}_q)[r]$, $P \neq P_\infty$, $Q \in E(\mathbb{F}_{q^k})[r] \setminus E(\mathbb{F}_q)[r]$.
OUTPUT: $e_r(P,Q)$
1: Generate a random point $R \in \{aP - bQ | 1 < a, b < r\}$.
2: $T \leftarrow P$, $S \leftarrow Q$, $f \leftarrow 1$,$g \leftarrow 1$.
3: for $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ down to 0 do
4: $f = f^2 \cdot l_{T,T}(Q+R) \cdot v_{2T}(R) \cdot l_{S,S}(-R) \cdot v_{2S}(P-R)$
5: $g = g^2 \cdot v_{2T}(Q+R) \cdot l_{T,T}(R) \cdot v_{2S}(-R) \cdot l_{S,S}(P-R)$
6: $T = 2T$, $S = 2S$
7: if $r_i = 1$ then
8: $f = f \cdot l_{T,P}(Q+R) \cdot v_{T+P}(R) \cdot l_{S,Q}(-R) \cdot v_{S+Q}(P-R)$
9: $g = g \cdot v_{T+P}(Q+R) \cdot l_{T,P}(R) \cdot v_{S+Q}(-R) \cdot l_{S,Q}(P-R)$
10: $T = T + P$, $S = S + Q$
11: end if
12: end for
13: Return f/g.

Algorithm 2. Optimized Miller's Algorithm for Computing Tate Pairing.

INPUT: $P \in E(\mathbb{F}_q)[r]$, $P \neq P_\infty$, $Q \in E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$.
OUTPUT: $\langle P, Q \rangle_r^{(q^k-1)/r}$
1: $T \leftarrow P$, $f \leftarrow 1$.
2: for $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ down to 0 do
3: $f = f^2 \cdot l_{T,T}(Q)$
4: $T = 2T$
5: $g = g^2 \cdot v_T(Q)$
6: if $r_i = 1$ then
7: $f = f \cdot l_{T,P}(Q)$
8: $T = T + P$
9: $g = g \cdot v_T(Q)$
10: end if
11: end for
12: $w \leftarrow (f/g)^{(q^k-1)/r}$
13: Return w.

With these optimizations, we can reduce the number of operations in evaluating both the Weil and the Tate pairings. For the case of the Tate pairing, applying these optimizations make the algorithm a lot more efficient and comparable to the performance of the squared Tate pairing which will be described in Section 2.3 below.

2.3 The Squared Weil and Tate Pairings

In [5], Eisentrager, Lauter and Montgomery introduced a pairing technique called the *squared* Weil and Tate pairings. These can be solved by deterministic

algorithms and they are significantly faster than the standard Tate and Weil pairing algorithms. For a detailed discussion on these pairings, refer to [5].

Let r be a positive integer satisfying the same conditions of the Weil pairing. In our implementations, similar to the optimizations that we did to the ordinary pairings, we take $P \in E(\mathbb{F}_q)[r]$ and $Q \in E(\mathbb{F}_{q^k})[r] \setminus E(\mathbb{F}_q)[r]$. Then the *squared* Weil pairing is defined as

$$e_r(P,Q)^2 = (-1)^r \frac{f_P(Q) \cdot f_Q(-P)}{f_P(-Q) \cdot f_Q(P)}.$$

On the other hand, for $P \in E(\mathbb{F}_q)[r]$ and $Q \in E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$, we define the *squared* Tate pairing as

$$< P, Q >_r^{2\cdot((q^k-1)/r)} = \frac{f_P(Q)}{f_P(-Q)}.$$

Given below are the deterministic algorithms to compute the *squared* Weil and the *squared* Tate pairings. For details on the optimizations that can be applied for both algorithms, refer to [5].

Algorithm 3. Algorithm for Computing Squared Weil Pairing.

INPUT: $P \in E(\mathbb{F}_q)[r]$, $P \neq P_\infty$, $Q \in E(\mathbb{F}_{q^k})[r] \setminus E(\mathbb{F}_q)[r]$.
OUTPUT: $e_r(P,Q)^2$
1: $T \leftarrow P$, $S \leftarrow Q$, $f \leftarrow 1$, $g \leftarrow 1$
2: for $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ down to 0 do
3: $f = f^2 \cdot l_{T,T}(Q) l_{S,S}(-P)$
4: $g = g^2 \cdot l_{T,T}(-Q) l_{S,S}(P)$
5: $T = 2T$, $S = 2S$
6: if $r_i = 1$ then
7: $f = f \cdot l_{T,P}(Q) l_{T,Q}(-P)$
8: $g = g \cdot l_{S,P}(-Q) l_{S,Q}(P)$
9: $T = T + P$, $S = S + Q$
10: end if
11: end for
12: Return f/g

Algorithm 4. Algorithm for Computing Squared Tate Pairing.

INPUT: $P \in E(\mathbb{F}_q)[r]$, $P \neq P_\infty$, $Q \in E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$.

OUTPUT: $< P, Q >_r^{2\cdot((q^k-1)/r)}$

1: $T \leftarrow P$, $f \leftarrow 1$, $g \leftarrow 1$.
2: for $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ down to 0 do
3: $\quad f = f^2 \cdot l_{T,T}(Q)$
4: $\quad g = g^2 \cdot l_{T,T}(-Q)$
5: $\quad T = 2T$
6: $\quad$ if $r_i = 1$ then
7: $\quad\quad f = f \cdot l_{T,P}(Q)$
8: $\quad\quad g = g \cdot l_{T,P}(-Q)$
9: $\quad\quad T = T + P$
10: $\quad$ end if
11: end for
12: $w \leftarrow (f/g)^{(q^k-1)/r}$
13: Return w.

Notice that the vertical lines v_{A+B} which requires computation in any algorithm of the Weil/Tate pairings do not appear in Algorithms 3 and 4 because $v_{A+B}(Q)$ and $v_{A+B}(-Q)$ (or $v_{A+B}(P)$ and $v_{A+B}(-P)$)are equal and therefore, can be cancelled out. Furthermore, the algorithms will terminate and output the correct answer in polynomial time.

3 Pairing Friendly Elliptic Curves

The security of pairing-based cryptosystems depend on the infeasibility of the Discrete Logarithm Problem (DLP) on the groups $E(\mathbb{F}_q)$ and ${\mathbb{F}_{q^k}}^*$. In this regard, we choose the parameters q and k such that DLP on these two groups are hard to compute and $|E(\mathbb{F}_q)|$ has a large prime factor r. For example, a pairing is considered secure against today's best attacks when $r \sim 2^{160}$ and $k = 6$ to 10 as in [7]. Elliptic curves with small k and a large r are called 'pairing-friendly elliptic curves' and they are classified into two groups, namely, supersingular and ordinary.

Supersingular elliptic curves have embedding degree k= 1, 2, 3, 4 or 6 and for $k \leq 3$, they must be defined on fields of characteristic not equal to 2 or 3. They also have distortion maps which make computations much faster and more efficient. Many researches have shown that the best choices for parameters are in characteristic three, e.g. [9] and [11], so for our experiment, we will use the popular curve over the field $\mathbb{F}_{3^{163}}$, given by the equation

$$E : Y^2 = X^3 - X + 1.$$

The order of the group is given by $|E(\mathbb{F}_{3^{163}})| = 3^{163} + 3^{(163+1)/2} + 1$ and the large prime divisor r is 253 bits. We will use the following distortion map to make computations more efficient.

$$(X, Y) \longmapsto (\alpha - X, iY),$$

where $i \in \mathbb{F}_{3^2}$, i satisfies $i^2 = -1$ and $\alpha \in \mathbb{F}_{3^3}$, $\alpha^3 - \alpha - 1{=}0$.

There are some disadvantages in using supersingular curves. With the advent of supercomputers, they may not be secure enough to use in cryptosystems since the embedding degrees are bounded and there are efficient algorithms for computing the DLP in small characteristic fields. Furthermore, there are limited number of parameter choices for the case of fields of characteristics 2 and 3.

In that case, we can turn our attention on another type of pairing-friendly curves that we can use — ordinary elliptic curves. Many papers have been published on the construction of this kind of curves and a survey on this was written by Freeman, Scott and Teske [6].

In [17], Tanaka developed an algorithm to generate pairing-friendly elliptic curves of embedding degree 8 over finite prime fields by improving the method of Brezing and Weng (see [6] and [17] for further details). We chose this curve because the author gave an explicit method and examples to generate pairing friendly curves of embedding degree 8. The curve can be generated using the following parameters.

$$\begin{aligned} t(x) &= -8x^2 - 108x^2 - 54x - 8 \\ r(x) &= 82x^4 + 108x^3 + 54x^2 + 12x + 1 \\ q(x) &= 379906x^6 + 799008x^5 + 705346x^4 + 333614x^3 + 88945x^2 \\ &\quad +12636x + 745 \\ n(x) &= q(x) + 1 - t(x), \end{aligned}$$

where t is the trace of Frobenius of the curve, r is the large prime which divides the order of the group, q is the characteristic of the finite field and n is the number of points of the elliptic curve E over $\mathbb{F}_q$.

With $x = 24000000000010394$, ($\log_2 x{=}54.4$), we get an elliptic curve in [17] with the following parameters which we will use in our programs. Note that the large prime divisor r of the group order is 224 bits. It is easy to verify that this curve is pairing-friendly once the parameters are given numerically as follows.

$$E : Y^2 \equiv X^3 + aX \pmod{q} \quad (a \neq 0)$$

$$\begin{aligned} t &= -113356800000147285043200063789391713609209096429146 0 \\ r &= 272056320000471307161600306182614014808404525177076771 \\ &\quad 93482845476817 \\ q &= 726011672004446604951703464791789328991217313776602768 \\ &\quad 811505320697581567547878422987036476401963225900 69 \\ n &= 726011672004446604951703464791789328991217313776602780 \\ &\quad 014718532071231007186788480192620783732287286881530 \\ a &= 1/2 \end{aligned}$$

Freeman, in [7] proposed a way to generate a whole family of pairing-friendly ordinary elliptic curves of embedding degree 10 with prime orders. This is an addition to the work of Miyaji, Nakabayashi and Takano[14], who gave a complete characterization of ordinary elliptic curves with embedding degrees 3, 4 and 6 of prime order and Barreto–Naehrig[2] who provided a method to produce curves of prime order with embedding degree 12. We chose this particular curve because this is the only good example for embedding degree 10.

We can generate a 'pairing-friendly' elliptic curve of embedding degree 10 with prime order by using the following parameters.

$$\begin{aligned} t(x) &= 10x^2 + 5x + 3 \\ r(x) &= 25x^4 + 25x^3 + 15x^2 + 5x + 1 \\ q(x) &= 25x^4 + 25x^3 + 25x^2 + 10x + 3 \end{aligned}$$

where the notations are the same as the ones we used for Tanaka's curves, except that $r(x) = n(x)$. If the equation $u^2 - 15Dv^2 = -20$ has a solution with $u \equiv 5 \pmod{15}$, then (t, r, q) represents a family of curves of embedding degree 10. For details on the construction and the algorithm he used, see [7].

For our experiment, we will use a 234-bit curve (published example [7])

$$E : Y^2 = X^3 + AX + B$$

which was generated with D= 1227652867 with the following parameters.

$$\begin{aligned} r &= 182116508039694720644932643473759500459342546966570904207262300432038 03 \\ q &= 182116508039694720644932643473759500459342546966570904207262300432038 03 \\ A &= -3 \\ B &= 157486680949134011847779644735228590869008312749229489733206849959032 75 \end{aligned}$$

4 Results

We implemented six algorithms, two of which are the standard Weil/Tate pairing algorithms and the remaining four are the ones discussed above. Their efficiency are measured over three pairing friendly elliptic curves, two of which are ordinary and one is supersingular. We generated 10 pairs of points for each curve as inputs in our programs. Our programs are written in MAGMA and ran on an AMD Opteron, 2GHZ dual core machine with 4GB of memory. Below are the tables for the average timings of the algorithms in seconds. Note that the timings for the Tate/Squared Tate pairings do not include the final exponentiation.

Timings for the Weil/squared Weil Pairings

	Tanaka	Freeman	SS(w/o dist.)	SS(w/dist.)
Weil Pairing	2.091	2.373	75.998	31.719
Weil Pairing (w/ opt.)	0.479	0.538	12.286	5.980
Squared Weil pairing	0.307	0.341	9.403	3.473

Timings for the Tate/squared Tate Pairings

	Tanaka	Freeman	SS(w/o dist.)	SS(w/dist.)
Tate Pairing	0.349	0.370	14.935	14.806
Tate Pairing (w/ opt.)	0.085	0.091	2.359	1.813
Squared Tate pairing	0.085	0.091	2.384	2.109

5 Observations

We have demonstrated the first comparison of the Weil/Tate pairings and squared Weil/Tate pairings on supesingular curves over characteristic 3 field and ordinary curves of embedding degrees 8 and 10. Based on the timings we gathered in our implementations, we can see that pairing evaluation on supersingular curves using distortion maps are much faster than the pairing evaluation without distortion maps. However, they appear to be much less efficient compared to the ordinary curves we chose of embedding degrees 8 and 10. We believe that this has to do with the choice of the extension fields where we evaluate the pairings. The polynomial we used to generate the extension field $\mathbb{F}_{q^k}$ for the ordinary elliptic curves have less hamming weight than the supersingular case. This is a problem with supersingular curves over characteristic 2 or 3 fields. There are only a small number of fields to choose from and we need an element of luck in our search for a suitable field [8].

We also observed that the squared Weil pairing is much efficient to compute compared to the Weil pairing, even when optimizations are applied. On the other hand, the efficiency of the squared Tate pairing is comparable to the Tate pairing when optimizations are applied for the ordinary curves. However, for the case of supersingular curves, the squared Tate pairing is not faster than the optimized Tate pairing. The reason for this can be checked by comparing the loop from line 2 to line 11 of both Algorithms 2 and 4 in our source code. The squared Tate pairing requires one more multiplication compared to the optimized Tate pairing which takes more time to compute.

We would like to point out that this experiment does not consider security implementations other than the bit size of the order of the r-torsion subgroup, the finite field $\mathbb{F}_q$ and the extension field $\mathbb{F}_{q^k}$. We are basing our conclusions on the computational efficiency of the pairings, the pairing-friendly curves that we used and the underlying field to which we evaluate the pairings.

References

[1] P. Barreto, H. Kim, B. Lynn, M. Scott. Efficient Algorithms for Pairing Based Cryptosystems. *Advances in Cryptology - Crypto 2002*, pp. 354-3-68, LNCS vol. 2442, Springer-Verlag, 2002.

[2] P. Barreto, M. Naehrig, Pairing-Friendly Elliptic Curves of Prime Order, in: SAC 2005, ed. B. Preneel, S. Tavares, LNCS vol. 3897, pp. 319–331, Springer-Verlag 2006.

[3] D. Boneh, M. Franklin, Identity Based Encryption Scheme from the Weil Pairing, in CRYPTO '01, Springer LNCS 2248 (2001), pp. 213–239.

[4] A.J. Devegili, M.Scott, R. Dahab, Implementing Cryptographic Pairings over Barreto-Naehrig Curves, in: Pairing-Based Cryptography Pairing 2007. LNCS vol. 4575, pp. 197–207, Springer-Verlag 2007.

[5] K. Eisentrager, K. Lauter, P. Montgomery. Improved Weil and Tate Pairing for Elliptic and Hyperelliptic Curves, in Algorithmic Number Theory Symposium, ANTS VI, LNCS vol. 3076, pp. 169–183, Springer-Verlag 2004.

[6] D. Freeman, M. Scott, E. Teske, A Taxonomy of Pairing-Friendly Elliptic Curves, preprint, 2006. Available at http://math.berkeley.edu/dfreeman/papers/taxonomy.pdf

[7] D. Freeman, Constructing Pairing Friendly Elliptic Curves with Embedding Degree 10,In: Algorithmic Number Theory, LNCS vol. 4076, pp. 452–465, Springer Heidelberg, 2006.

[8] S. Galbraith, Pairings, in: Advances in Elliptic Curve Cryptography, pp. 183–213, Cambridge University Press, 2005.

[9] S. Galbraith, Supersingular Curves in Cryptography, in Advances in Cryptology - ASIACRYPT 2001. Springer-Verlag, LNCS 2248, pp. 495–513, 2001,

[10] S. Galbraith, K. Harrison, D. Soldera. Implementing the Tate Pairing. Algorithmic Number Theory, 5th International Symposium ANTS-V, LNCS vol. 2369, pp. 20–32, Springer-Verlag, 2002.

[11] K. Harrison, D. Page, N.P. Smart, Software Implementation of Finite Fields of Characteristic 3, for Use in Pairing Based Cryptosystem. In: LMS Journal of Computation and Mathematics, London, vol.5(1), pp.181–193, London Mathematical Society, London 2002.

[12] A. Menezes, T. Okamoto, S. Vanstone, Reducing Elliptic Curve Logarithms to Logarithms In A Finite Field. In: Proceedings 23rd Annual ACM symposium on Theory of Computing, pp. 80–89. ACM press, New York (1991)

[13] V. Miller, Short Programs for Functions on Curves, unpublished manuscript, 1986. Available at http://crypto.stanford.edu/miller/miller.pdf.

[14] A. Miyaji, M. Nakabayashi, S. Takano, New Explicit Conditions of Elliptic Curve Traces for FR-Reduction, IEICE Taransactions on Fundamentals, pp. 1234–1243, E84–A(5)(2001).

[15] D. Page, N.P. Smart, F. Vercauteren, A Comparison of MNT Curves and Supersingular Curves, AAECC vol 17, pp. 379–392, Springer-Verlag 2006.

[16] M Scott, P. Barreto. Generating More MNT Elliptic Curves. *Designs, Code and Cryptography*, Vol. 38, No.2, pp.209–217, 2006

[17] S. Tanaka, More Constructing Pairing-Friendly Elliptic Curves for Cryptography, Masters Thesis, Tokyo Metropolitan Univerity (2007). Available at www.tnt.math.metro-u.ac.jp/labo/master/2006/satoru/thesis2006r1.pdf (in japanese).

[18] S. Tanaka, K. Nakamula, More Constructing Pairing-Friendly Elliptic Curves for Cryptography, to appear in Transactions of the Japan Society for Industrial and Applied Mathematics (JSIAM), vol. 17, No. 4, 2007. (in japanese)