# Computing Pairings Using x-Coordinates Only

Steven D. Galbraith[1]* and Xibin Lin[2]**

[1] Mathematics Department, Royal Holloway University of London
Egham, Surrey, TW20 0EX, UK
[2] School of Mathematics and Computational Science
Sun-Yat Sen University Guangzhou, 510275, P.R.China
Steven.Galbraith@rhul.ac.uk
linxibin@mail2.sysu.edu.cn

**Abstract.** To reduce bandwidth in elliptic curve cryptography one can transmit only $x$-coordinates of points (or $x$-coordinates together with an extra bit). For further computation using the points one can either recover the $y$-coordinates by taking square roots or one can use point multiplication formulae which use $x$-coordinates only.
We consider how to efficiently use point compression in pairing-based cryptography. We give a method to compute compressed Weil pairings using $x$-coordinates only. We also show how to compute the compressed Tate and ate pairings using only one $y$-coordinate. Our methods are more efficient than taking square roots when the embedding degree is small. We implemented the algorithms in the case of embedding degree 2 curves over $\mathbb{F}_p$ where $p \equiv 3 \pmod 4$ and found that our methods are $10-15\%$ faster than the analogous methods using square roots.

**Keywords:** elliptic curves, pairings, point compression.

## 1 Introduction

To reduce bandwidth in elliptic curve cryptography one can transmit only $x$-coordinates of points (or $x$-coordinates together with an extra bit). This is called point compression (see Section IV.4 of [5] for more details). For further computation using the points one can either recover the $y$-coordinates by taking square roots or one can use point multiplication formulae which use $x$-coordinates only (for example, the formulae due to Montgomery [22]).

Cryptographic parings have many applications in cryptography. To reduce the bandwidth of pairing-based cryptosystems it is natural to use point compression. Indeed, this trick is necessary for certain applications such as the BLS short signature scheme [7]. However, due to the security requirements of elliptic curves for pairing applications (especially when the embedding degree is small) one might be required to work over a large field and so taking square roots might be expensive. For this reason (and also, out of mathematical curiosity) it is natural to ask whether pairings can be computed using $x$-coordinates only.

When we discard $y$-coordinates we are essentially working up to sign. Let $q$ be a prime power and let $e_n$ be a bilinear pairing into the group of $n$-th roots of unity $\mu_n \subseteq \mathbb{F}_{q^k}^*$ where $k$ is even and $n \nmid (q^{k/2}-1)$. We have $e_n(\pm P, \pm Q) = z^{\pm 1}$. Note that the norm of $z$ with respect to $\mathbb{F}_{q^k}/\mathbb{F}_{q^{k/2}}$ is 1 and so $z^{-1}$ is the Galois conjugate of $z$ in $\mathbb{F}_{q^k}/\mathbb{F}_{q^k}$. It follows that the trace $z+z^{-1}$ of the pairing value is independent of the choice of signs. This suggests that it might be possible to compute the trace of the pairing value from the $x$-coordinates only. One of the contributions of this paper is show that this can indeed be done.

Traces of pairing values are already used to compress pairing values [27, 16, 15, 24], so our methods can also be viewed as a form of compressed pairing computation. However, we only consider compression with respect to the quadratic extension $\mathbb{F}_{q^k}/\mathbb{F}_{q^{k/2}}$. If further compression is required then it can be performed after the pairing computation has been performed.

* This author thanks the EPSRC for support.
** This author thanks the Chinese Scholarship Council.

To save time cost of computing pairing value, many methods have been proposed to speed up the basic Miller algorithm, including the denominator elimination method [3], the construction of pairing-friendly elliptic curves [14], and the methods to shorten the Miller loop [2, 17]. Our formulae for pairing computation are essentially independent of these tricks and so our methods apply equally to the ate pairing.

This paper is organized as follows. Section 2 introduces the mathematical preliminaries, including the Miller Algorithm and compressed pairings, Lucas sequences, and the conventional methods to compute the compressed Tate pairing for $k = 2$. Section 3 first describes the method of point multiplication using $x$-coordinates only, then explains how to compute Miller functions (in a certain shape) using $x$-coordinates only. Section 4 shows how to use the results of Section 3 to compute compressed pairings using $x$-coordinates only. Section 5 considers the case where one $y$-coordinate is known (this arises in many cryptographic applications). Section 6 analyses the efficiency of our methods and reports on some implementation comparisons. Section 7 gives our conclusions.

## 2 Mathematical Preliminaries

### 2.1 Square Roots

It is well-known that one can compute square roots in finite fields in polynomial time. A particularly simple case is when $p \equiv 3 \pmod 4$, where $x = a^{(p+1)/4} \pmod p$ is a solution to $x^2 \equiv a \pmod p$. This can be computed efficiently using standard modular exponentiation algorithms, but if $p$ is 512 bits then this still needs around 510 modular squarings and roughly 255 modular multiplications. A small improvement may be obtained using window methods, but as far as we know this method is essentially the most efficient method to compute square roots modulo $p$.

For other values of $p$ the Tonelli-Shanks method can be used, but it is typically less efficient than the above. Hence, computing square roots is a non-trivial operation which can lead to a significant overhead in cryptographic protocols using point compression. In standard elliptic curve cryptography one can work with 160–256 bit primes and the problem may not be so significant. But in pairing-based cryptography, when the embedding degree is small, square root computation could be a considerable burden.

Computing roots is more efficient when working over extension fields. Barreto et al ([3] Section 4) present a more efficient method for square roots in $\mathbb{F}_{p^m}$ where $p \equiv 3 \pmod 4$ and $m > 1$ is odd. In characteristic 2 the problem of recovering the $y$-coordinate in point compression boils down to solving an equation of the form $x^2 + x = a$ in $\mathbb{F}_{2^m}$, which can be very efficiently solved when $m$ is odd (see Section II.2.4 of [5]).

Hence, for pairing-based cryptography using fields of small characteristic it is relatively efficient to recover the $y$-coordinates of compressed points. Therefore, in this situation there is no motivation for developing pairing algorithms using $x$-coordinates only. For this reason we restrict to odd characteristic in this paper.

### 2.2 Miller's Algorithm

Let $E$ be an elliptic curve over $\mathbb{F}_q$, $P$ and $Q$ be points on $E$ of order $n$, and $n \mid (q^k - 1)$. Recall that the Weil pairing (technically this is $1/e_n(P, Q)$ but the literature usually calls the below formulation the Weil pairing) may be computed as

$$e_n(P, Q) = (-1)^n f_{n,P}(Q)/f_{n,Q}(P)$$

where $f_{n,P}$ is a function on $E$ with divisor $n(P) - n(\infty)$. Similarly, the Tate pairing is

$$t_n(P, Q) = f_{n,P}(Q)^{(q^k-1)/n}.$$

One builds the functions $f_{n,P}$ by an interative process known as Miller's algorithm. The key to this process are the rules $f_{1,P} = 1$ and

$$f_{n+m,P} = f_{n,P} f_{m,P} l/v$$

where $l$ is the line between $[n]P$ and $[m]P$ (tangent line if $[n]P = [m]P$) and $v$ is the vertical line through $[n + m]P$. For future reference we now present Miller's algorithm for computing $f_{n,P}(Q)$.

**Algorithm 1: Miller's Algorithm**

```
Input: $P, Q$ and elliptic curve $E$, and $n = \sum_{i=0}^{l} a[i]2^i$.
Output: $f_{n,P}(Q)$
1 $R \leftarrow P$, $f_1 \leftarrow 1$
2 for $i = l-1, l-2, \ldots, 1, 0$ do
2.1 $f_1 \leftarrow f_1^2 \cdot \frac{l_{R,R}(Q)}{v_{2R}(Q)}$, $R \leftarrow 2R$
2.2 if $a[i] = 1$ then
2.3 $f_1 \leftarrow f_1 \cdot \frac{l_{R,P}(Q)}{v_{R+P}(Q)}$, $R \leftarrow R + P$
2.4 end if
2.5 end for
3 return $f_1$
```

### 2.3 Lucas Sequences

Let $z \in \mathbb{F}_{q^2}$ be an element of order dividing $q+1$ and recall the trace map $\mathrm{Tr}(z) = z + z^q$. We briefly recall how to compute $\mathrm{Tr}(z^i)$ efficiently from $\mathrm{Tr}(z)$ using Lucas sequences. Define $V_i = \mathrm{Tr}(z^i)$. The following Lemma recalls the standard facts on $V_i$ (see Section 3.6.3 of [13]).

**Lemma 1.** *The Lucas sequence satisfies:*

1. $V_0 = 2$ *and* $V_1 = \mathrm{Tr}(z)$.
2. $V_{i+1} = \mathrm{Tr}(z)V_i - V_{i-1}$.
3. $V_{2i} = V_i^2 - 2$.
4. $V_{2i+1} = V_i V_{i+1} - \mathrm{Tr}(z)$.

### 2.4 Point Compression in Pairing Based Cryptography

The philosophy of point compression is simple: to reduce the bandwidth one can transmit only the $x$-coordinates of any points required to be sent during runs of a protocol. One can also extend this idea to public keys or other system parameters, but the motivation may be weaker for this data since the bandwidth for set-up may not be so constrained.

There are now two choices to implement the protocol. The first is to recover $y$-coordinates for all points by taking square roots and then perform point multiplication and pairing computations in the conventional way. The alternative is to work with $x$-coordinates only. Pairing computation using $x$-coordinates only is not expected to be faster than conventional methods, but avoiding computing the square roots can lead to an overall improvement in performance.

We now consider a small number of examples of cryptographic protocols using pairings to discuss how point compression can be used.

1. The Joux 3-party key exchange protocol [19]: The data transmitted is the points $[a]P$, $[b]P$ and $[c]P$. We can send $x$-coordinates only. To compute the shared key, user A needs to compute $\mathrm{Tr}(e_n([b]P, [c]P)^a)$. Hence the conventional solution requires two square root operations.
2. Boneh-Lynn-Shacham signatures [7]: The public key is $Q = [a]P$ and the private key is $a$. To sign $m$ we hash to a point $H(m)$ and the signature is $\sigma = [a]H(m)$. To get a short signature it is required to send only the $x$-coordinates for $\sigma$. To verify one must test whether $\mathrm{Tr}(e_n(P, \sigma)) = \mathrm{Tr}(e_n(Q, H(m)))$. Hence the conventional solution required one square root (or two if the public key is also compressed).
3. There are many examples of identity-based encryption schemes using pairings. We recall here the ID-based KEM based on the Sakai-Kasahara approach, which has been submitted to the IEEE P1363 draft standard on identity based cryptography [1].
   The master key is $P' = [s]P$. Given an identity $ID$ the private key is $D_{ID} = [(s + H_1(ID))^{-1}]Q$.
   To encapsulate a key to user $ID$: Choose a random bitstring $m$, set $r = H_3(m)$, $R = P' + [H_1(ID)]P$, $U = [r]R$, $V = m \oplus H_2(e(P, Q))$ and $k = H_4(m)$. The key encapsulation is $(U, V)$, where one could apply point compression to the component $U$. The encapsulated key is $k$.

To recover the key user $ID$ computes $\alpha = e(U, D_{ID})$, $m = H_2(\alpha) \oplus V$, $r = H_3(m)$, $R = P' + [H_1(ID)]P$, if $U \neq [r]R$ then fail, else $k = H_4(m)$. Note that the pairing computation involves the private key (for which both $x$ and $y$ are known) and the point $U$, for which only the $x$-coordinate is known.
The Boneh-Boyen ID-based KEM [8, 10] using point compression of the ciphertext also results in pairing computations where only one $y$-coordinate is known.

### 2.5 Conventional Methods to Compute the Compressed Tate Pairing

For the sake of comparision, we briefly recall some existing methods for computing pairings when the embedding degree $k$ is even. These methods are referred to as the conventional methods. We particularly focus on the case of $k = 2$, since our performance comparison in Section 6 is for this case. The merits of using elliptic curves of embedding degree 2 is discussed by Scott in [26]. We also assume that order of the subgroup of interest has low Hamming weight, hence we only count the cost of doublings in Miller's algorithm.

The first type methods are the ones that apply to general elliptic curves. These include the standard BLKS method in affine coordinates[3], and the methods in jacobian coordinates in [26] [12][18].

The second type is the method using elliptic curves with efficient endomorphisms, which was proposed by Scott in [25]. This method depends on the choice of certain non-supersingular elliptic curves. This method is referred to as Tate using endomorphisms. For details of this case see Appendix C.

When point compression technique is used, the cost of the overall computation of the compressed Tate pairing consists of three main parts: the square root extractions, the Miller loop, and the final exponentiation. The cost of the first and the third parts varies slightly according to the parameters. While the operation of each doubling step in the Miller loop for different methods can be approximately counted, which is summarized in the following Table.

Here and thereafter, $M$, $S$ and $I$ mean multiplication, squaring and inversion in $\mathbb{F}_q$ respectively. In this paper, we also assume $M \approx S$. For concrete examples, we consider $q \equiv 3 \bmod 4$, so it is reasonable to assume that one multiplication in $\mathbb{F}_{q^2}$ costs $3M$, and one squaring in $\mathbb{F}_{q^2}$ costs $2M$.

**Table 1.** Cost of Existing Methods in Miller Step

| Method | Operation |
|---|---|
| Affine Coordinate | $10M + 1I$ |
| Jacobian Coordinate[12] | $19M$ |

## 3 Point Multiplication and Miller Functions Using $x$-Coordinates Only

In this section we present our main technical result, which describes the shape of Miller functions treating the $y$-coordinates of the points as an unknown. We provide simple recurrence formulae which enable efficient computation of these functions. First, to set the scene, we recall how to derive the formulae for point multplication using $x$-coordinates only.

### 3.1 Elliptic Curve Point Multiplication using $x$-Coordinates Only

Let $E : y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{F}_q$ and let $P = (x_P, y_P)$. Let $[i]P = (x_i, y_i)$. From the fact that $-[i]P = [i](-P)$ it follows that $x_i$ depends only on $x_P$ and $i$. The existence of formulae for $x_i$ using $x$-coordinates only is clear from the theory of division polynomials (for example, see the Formulary of Cassels [11]). It follows that for any elliptic curve in odd characteristic there are rational functions $g_{i,1}(x), g_{i,2}(x)$ such that

$$[i]P = (g_{i,1}(x_P), y_P g_{i,2}(x_P)).$$

In general, evaluating such expressions is less efficient than using the standard formulae for the group law on elliptic curves, however Montgomery [22] presented very efficient formulae and a ladder algorithm to compute the $x$-coordinate of $[i]P$ from the $x$-coordinate of $P$.

For our application we will need not just $x_i$ but the value $y_i/y_P$ (this is because we need to compute slopes for the Miller functions). Hence, we cannot just use Montgomery's results. Define $u_i$ to be $y_i/y_P$. Clearly, $x_1 = x_P$ and $u_1 = 1$. The remainder of this section is devoted to giving recurrence formulae for these rational functions.

**Lemma 2.** *Let $(x_i, y_P u_i) = [i](x_P, y_P)$ and suppose that $x_i^3 + Ax_i + B \neq 0$. Then*

$$x_{2i} = \frac{(3x_i^2 + A)^2}{4(x_i^3 + Ax_i + B)} - 2x_i, \quad u_{2i} = u_i \left( \frac{3x_i^2 + A}{2(x_i^3 + Ax_i + B)} (x_i - x_{2i}) - 1 \right)$$

*and, if $x_i \neq x_P$*

$$x_{i+1} = \frac{(x_P^3 + Ax_P + B)(u_i - 1)^2}{(x_i - x_P)^2} - x_i - x_P, \quad u_{i+1} = \frac{(u_i - 1)(x_P - x_{i+1})}{x_i - x_P} - 1.$$

*Proof.* For doubling, the slope is $\lambda = (3x_i^2 + A)/(2y_i)$ so $\lambda^2 = (3x_i^2 + A)^2/(4y_i^2)$ and $x_{2i} = \lambda^2 - 2x_i$. The formula for $x_{2i}$ is immediate.

The $y$-coordinate of $[2](x_i, y_i)$ is $y_{2i} = -(\lambda(x_{2i} - x_i) + y_i)$. For this calculation we write

$$\lambda = \frac{y_i(3x_i^2 + A)}{2y_i^2} = y_P u_i \frac{3x_i^2 + A}{2(x_i^3 + Ax_i + B)}. \tag{1}$$

Hence, using $y_{2i} = y_P u_{2i}$, we have

$$u_{2i} = -u_i \frac{(3x_i^2 + A)(x_{2i} - x_i)}{2(x_i^3 + Ax_i + B)} - u_i$$

which establishes the formula.

For adding $(x_P, y_P)$ to $(x_i, y_i)$, when $x_P \neq x_i$, we have $\lambda = (y_i - y_P)/(x_i - x_P) = y_P(u_i - 1)/(x_i - x_P)$. The formulae follow easily.

Points of order 2 satisfy $u_i = 0$.

## 3.2 Miller Functions Using $x$-Coordinates Only

Let $E : y^2 = x^3 + Ax + B$ over $\mathbb{F}_{p^d}$, where $p$ is an odd prime. Let $n$ be a divisor of $\#E(\mathbb{F}_{p^d})$ and let the embedding degree of $n$ be $k$, which we assume to be even. Define $q = p^{dk/2}$ and write $\mathbb{F}_{q^2} = \mathbb{F}_q(\theta)$ where $\theta^2 = w$ for some $w \in \mathbb{F}_q$. Let $P = (x_P, y_P), Q = (x_Q, \theta y_Q) \in E(\mathbb{F}_{q^2})$ be points of order $n$ with $x_P, y_P, x_Q, y_Q \in \mathbb{F}_q$.

We assume for simplicity and efficiency that $q \equiv 3 \pmod 4$, so that we may take $\theta$ such that $\theta^2 = -1$. The method also works for the general case $\theta^2 = w$. However, it is natural to restrict to the case $q \equiv 3 \pmod 4$ when using point compression since it makes computing square roots easier. Since our methods are faster that the convential approach in the case $q \equiv 3 \pmod 4$ then they should be even better in the more general case.

To compute pairings we must compute the Miller functions $f_{i,P}(Q)$ and $f_{i,Q}(P)$ for various $i$ corresponding to initial segments of the binary expansion of $n$. The following result is the key observation.

**Lemma 3.** *Let notation be as above and assume that $n$ is odd. and let $f_i$ be either $f_{i,P}(Q)$ or $f_{i,Q}(P)$. If $i$ is even or $i = n$ then*

$$f_{i,P}(Q) = y_P \alpha_{i,P} + \theta y_Q \beta_{i,P} \qquad f_{i,Q}(P) = y_P \alpha_{i,Q} + \theta y_Q \beta_{i,Q}$$

*and if $i$ is odd then*

$$f_{i,P}(Q) = \alpha_{i,P} + \theta y_P y_Q \beta_{i,P} \qquad f_{i,Q}(P) = \alpha_{i,Q} + \theta y_P y_Q \beta_{i,Q}$$

*where $\alpha_{i,P}, \beta_{i,P}, \alpha_{i,Q}, \beta_{i,Q}$ all depend on the $x$-coordinates of $P$ and $Q$ and the equation of the curve.*

*Remark 1.* If $n$ is even, say $n = 2^s m$, then the values $f_i$ are as above when $i$ is an initial segment of $m$. However, the final $s$ squarings are different. There are three cases: (1) $[i]P = \mathcal{O}$; (2) $[i]P \neq \mathcal{O}$ but $[2i]P = \mathcal{O}$; (3) $[2i]P \neq \mathcal{O}$. In cases (1) and (2) the value for $f_i$ is of the form $\alpha_i + \theta y_P y_Q \beta_i$ while in case (3) the value is of the form $\alpha_i y_P + \theta y_Q \beta_i$.

*Proof.* To prove this we will give explicit recurrence formulae for the $\alpha_i$ and $\beta_i$.

First, consider the case of $f_{i,P}(Q)$. Let $(x_{i,P}, u_i y_P) = [i](x_P, y_P)$. We start with $i = 1$ (which is odd) and $\alpha_{1,P} = 1$, $\beta_{1,P} = 0$. For a doubling, we start with $f_i$ in either of the two forms and compute

$$f_{2i,P} = f_{i,P}^2 l(Q)/v(Q).$$

We have $f_{i,P}^2 = \alpha'_{2i,P} + \theta y_P y_Q \beta'_{2i,P}$ where $\alpha'_{2i,P}$ is either $y_P^2\alpha_{i,P}^2 - y_Q^2\beta_{i,P}^2$ or $\alpha_{i,P}^2 - y_P^2 y_Q^2 \beta_{i,P}^2$ depending on whether $i$ is even or odd. In both cases $\beta'_{2i,P} = 2\alpha_{i,P}\beta_{i,P}$.

The line $l$ has equation $y - y_{i,P} - \lambda(x - x_{i,P})$. Using equation (1), we have $\lambda = y_P u_{i,P}(3x_{i,P}^2 + A)/(2(x_{i,P}^3 + Ax_{i,P} + B))$, therefore,

$$l(Q) = \theta y_Q - y_P u_{i,P}\left(\frac{3x_{i,P}^2 + A}{2(x_{i,P}^3 + Ax_{i,P} + B)}(x_Q - x_{i,P}) + 1\right).$$

We write this as $\theta y_Q + y_P t_1$. The vertical line contributes simply $v(Q) = x_Q - x_{2i,P}$.

Multiplying all these together gives $f_{2i,P} = y_P \alpha_{2i,P} + \theta y_Q \beta_{2i,P}$ where

$$\alpha_{2i,P} = (\alpha'_{2i,P} t_1 - (x_Q^3 + Ax_Q + B)\beta'_{2i,P})/(x_Q - x_{2i,P})$$

and

$$\beta_{2i,P} = ((x_P^3 + Ax_P + B)t_1\beta'_{2i,P} + \alpha'_{2i,P})/(x_Q - x_{2i,P}).$$

This completes proof of first part of the first claim.

Now suppose a further addition is performed in Miller's algorithm. It is known that the final addition does not affect the form of the value. In general case, from Lemma 2 we deduce that the line $l$ is

$$y - y_P\left(\frac{u_{2i} - 1}{x_{2i} - x_P}(x - x_P) + 1\right).$$

and so

$$l(Q) = \theta y_Q - y_P\left(\frac{u_{2,i} - 1}{x_{2i} - x_P}(x_Q - x_P) + 1\right).$$

Writing this as $\theta y_Q + y_P t_2$, we have $f_{2i+1,P} = \alpha_{2i+1,P} + \theta y_P y_Q \beta_{2i+1,P}$ where

$$\alpha_{2i+1} = ((x_P^3 + Ax_p + B)\alpha_{2i,P} t_2 - (x_Q^3 + Ax_Q + B)\beta_{2i,P})/(x_Q - x_{2i+1,P}),$$

and

$$\beta_{2i+1,P} = (\alpha_{2i,P} + \beta_{2i,P} t_2)/(x_Q - x_{2i+1,P}).$$

This completes the proof in the case of $f_{i,P}(Q)$.

We now consider the values $f_{i,Q} = f_{i,Q}(P)$. Let $(x_{i,Q}, u_i y_Q \theta) = [i](x_Q, y_Q\theta)$. Lemma 2 handles this case as well (just absorb the $\theta$ into the value of $y$).

As before, we start with $\alpha_{1,Q} = 1$ and $\beta_{1,Q} = 0$ and first consider the case of doubling. The result about squaring the previous value is the same as before, so we have $f_{i,Q}^2 = \alpha'_{2i,Q} + \theta y_P y_Q \beta'_{2i,Q}$. The line in a doubling has slope

$$\lambda = (3x_{i,Q}^2 + A)/(2\theta y_{i,Q}) = \theta y_{i,Q}(3x_{i,Q}^2 + A)/(2(\theta y_{i,Q})^2) = \theta y_Q u_i(3x_{i,Q}^2 + A)/(2(x_{i,Q}^3 + Ax_{i,Q} + B)).$$

hence, we have $l(P)$ equal to

$$y_P - \theta y_Q u_i \left( \frac{3x_{i,Q}^2 + A}{2(x_{i,Q}^3 + Ax_{i,Q} + B)}(x_P - x_{i,Q}) + 1 \right).$$

Writing this as $y_P + \theta y_Q t_3$ and noting that $v(P) = (x_P - x_{2i,P})$, we have $f_{2i,Q} = f_{i,Q}^2 l(P)/v(P) = y_P\alpha_{2i,Q} + \theta y_Q \beta_{2i,Q}$ where

$$\alpha_{2i,Q} = (\alpha'_{2i,Q} - (x_Q^3 + Ax_Q + B)\beta'_{2i,Q}t_3)/(x_P - x_{2i,Q})$$

and

$$\beta_{2i,Q} = \alpha'_{2i,Q}t_3 + (x_P^3 + Ax_P + B)\beta'_{2i,Q}/(x_P - x_{2i,Q}).$$

Finally, we consider a further addition (i.e.,obtaining $f_{2i+1,Q}$ from $f_{2i,Q}$. As before, the last iteration does not need to be computed. In the general case,

$$\lambda = \frac{\theta y_{2i,Q} - \theta y_Q}{x_{2i,Q} - x_Q} = \theta y_Q \frac{u_{2i} - 1}{x_{2i,Q} - x_Q}.$$

and so $l(P)$ equals

$$y_P - \theta y_Q \left( \frac{u_{2i} - 1}{x_{2i,Q} - x_Q}(x_P - x_Q) + 1 \right)$$

Writing this as $y_P + \theta y_Q t_4$, we have $f_{2i+1,Q} = \alpha_{2i+1,Q} + \theta y_P y_Q \beta_{2i+1,Q}$, where

$$\alpha_{2i+1,Q} = ((x_P^3 + Ax_P + B)\alpha_{2i,Q} - (x_Q^3 + Ax_Q + B)\beta_{2i,Q}t_4)/(x_P - x_{2i+1,Q})$$

and

$$\beta_{2i+1,Q} = (\alpha_{2i,Q}t_4 + \beta_{2i,Q})/(x_P - x_{2i+1,Q}).$$

This completes the proof. □

To summarize, given $x_P$ and $x_Q$ being the $x$-coordinates of points of order $n$ on $E(\mathbb{F}_q)$, one can compute the values $\alpha_{n,P}$, $\beta_{n,P}$, $\alpha_{n,Q}$ and $\beta_{n,Q}$ for $f_{n,P}(Q)$ and $f_{n,Q}(P)$ using knowledge of only $x_P$, $x_Q$ and the coefficients of the curve.

## 4 Computing Parings Using $x$-Coordinates Only

We now use the results of the previous section to compute various pairings using $x$-coordinates only. We then propose some optimizations of the computation.

### 4.1 The Weil Pairing

For interest we present the case of the Weil pairing even though this is not usually used in practical applications.

By Lemma 3 we have

$$e_n(P,Q) = (-1)^n \frac{f_{n,P}(Q)}{f_{n,Q}(P)} = (-1)^n \frac{y_P\alpha_{n,P} + \theta y_Q \beta_{n,P}}{y_P\alpha_{n,Q} + \theta y_Q \beta_{n,Q}}.$$

Multiplying numerator and denominator by the conjugate of the denominator gives

$$\begin{aligned} e_n(P,Q) &= (-1)^n \frac{(y_P\alpha_{n,P} + \theta y_Q\beta_{n,P})(y_P\alpha_{n,Q} - \theta y_Q\beta_{n,Q})}{y_P^2\alpha_{n,Q}^2 - y_Q^2\beta_{n,Q}^2} \\ &= (-1)^n \frac{(x_P^3 + Ax_P + B)\alpha_{n,P}\alpha_{n,Q} + (x_Q^3 + Ax_Q + B)\beta_{n,P}\beta_{n,Q} + \theta(\cdots)}{(x_P^3 + Ax_P + B)\alpha_{n,Q}^2 - (x_Q^3 + Ax_Q + B)\beta_{n,Q}^2} \end{aligned}$$

Hence, the trace of $e_n(P,Q)$ with respect to $\mathbb{F}_{q^2}/\mathbb{F}_q$ can be computed using $x$-coordinates only.

## 4.2 Tate and ate Pairings

For efficient pairing computation it is more common to use the ate or Tate pairings. As before suppose $q \equiv 3 \pmod 4$, $\mathbb{F}_{q^2} = \mathbb{F}_q(\theta)$ where $\theta^2 = -1$ and $P = (x_P, y_P), Q = (x_Q, \theta y_Q) \in E(\mathbb{F}_{q^2})$ have order $n$.

The Tate pairing is

$$t_n(P,Q) = f_{n,P}(Q)^{(q-1)(q+1)/n}$$

and the ate pairing is

$$a_n(P,Q) = f_{T,Q}(P)^{(q-1)(q+1)/n}$$

where $T$ is $t-1$ where $t$ is the trace of Frobenius.

We will show that one can compute the trace of $f_{n,P}(Q)^{q-1}$ and $f_{n,Q}(P)^{q-1}$ using $x$-coordinates only. Note that due to the exponentiation to the power $q-1$ one can apply denominator elimination during this computation. The remaining part of the final exponentiation (raising to the power $(q+1)/n$) is performed using Lucas sequences.

**Lemma 4.** *One can compute* $\mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(f_{n,P}(Q)^{q-1})$ *and* $\mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(f_{T,Q}(P)^{q-1})$ *using $x$-coordinates only.*

*Proof.* Following Lemma 3, we have $f_{n,P}(Q) = y_P\alpha_{n,P} + \theta y_Q\beta_{n,P}$ where $\alpha_{n,P}$ and $\beta_{n,P}$ lie in $\mathbb{F}_q$ and can be computed using $x$-coordinates only. Then

$$f_{n,P}(Q)^{q-1} = \frac{y_P\alpha_{n,P} - \theta y_Q\beta_{n,P}}{y_P\alpha_{n,P} + \theta y_Q\beta_{n,P}}.$$

Multiplying by the Galois conjugate of the denominator gives

$$f_{n,P}(Q)^{q-1} = \frac{(x_P^3 + Ax_P + B)\alpha_{n,P}^2 - (x_Q^3 + Ax_Q + B)\beta_{n,P}^2 + \theta(\cdots)}{(x_P^3 + Ax_P + B)\alpha_{n,P}^2 + (x_Q^3 + Ax_Q + B)\beta_{n,P}^2}$$

which shows that one can compute $\mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(f_{n,P}(Q)^{q-1})$.

The case of the ate pairing is exactly the same when $T$ is even (just swapping $P$ and $Q$ and replacing $n$ by $T$). For the case when $T$ is odd we have

$$f_{T,Q}(P)^{q-1} = \frac{\alpha_{T,Q} + \theta y_P y_Q\beta_{T,Q}}{\alpha_{T,Q} - \theta y_P y_Q\beta_{T,Q}}.$$

As before this simplifies to

$$\frac{\alpha_{T,Q}^2 - (x_P^3 + Ax_P + B)(x_Q^3 + Ax_Q + B)\beta_{T,Q}^2 + \theta(\cdots)}{\alpha_{T,Q}^2 + (x_P^3 + Ax_P + B)(x_Q^3 + Ax_Q + B)\beta_{T,Q}^2}$$

and clearly the trace can be computed using only $x$-coordinates.

As mentioned above, any step of the algorithm which results in multiplying both $\alpha_i$ and $\beta_i$ by a quantity in $\mathbb{F}_q^*$ can be omitted. This is the denominator elimination trick which has great benefit in pairing computations with even embedding degree.

*Remark 2.* 1. Clearly the above methods apply to any algorithm which involves computing a Miller function in $\mathbb{F}_{q^2}$ and then raising to a multiple of $q-1$. In particular, it immediately applies to Scott's fast pairing computation using an efficient endomorphism.
2. When using the Tate or ate pairing there is an exponentiation to the power $q-1$, and so denominator elimination can be used in the Miller algorithm as usual.
3. One can use any variant of projective coordinates for the pairing computation (in our case this is essentially just using $(x, z)$-coordinates). We sketch some details later in the paper.

## 5 Computing Pairings with Only One $y$-Coordinate

In some applications one may be pairing points $P$ and $Q$ where, say, only the $x$-coordinate of $Q$ is known but both coordinates of $P$ are known. The above methods can of course be applied by forgetting $y_P$, but in practice it is more efficient to use knowledge of $y_P$ to speed up the computation of the quantities $\alpha_i$ and $\beta_i$ in Lemma 3.

We assume in this section that $y_P$ is known, where the pairing computation involves the Miller function $f_{n,P}(Q)$. If instead $y_Q$ is known then it might be preferable to swap the arguments to the pairing function in the cryptographic protocol so that the full benefit of the method in this section is available.

The following gives the formulae of computing $\alpha_{n,P}$ and $\beta_{n,P}$ without $y$-coordinate of the second element.

**Lemma 5.** *Let notation be as in Lemma 3. Denote equivalence in $\mathbb{F}_{q^2}/\mathbb{F}_q$ by $\equiv$. Then*

$$f_{i,P}(Q) \equiv \alpha_{i,P} + \theta y_Q \beta_{i,P},$$

*where $\alpha_{i,P}$, $\beta_{i,P}$ depend only on $x_P$, $y_P$, and $x_Q$.*

*Proof.* The proof is essentially the same with Lemma 3, except here we can calculate the slope of the lines explicitly. We write down the formulae here for the sake of efficiency analysis in the following section.

As before let $\alpha_{1,P} = 1$ and $\beta_{1,P} = 0$. The line equations are the standard ones, which are of the form $\theta y_Q + t_1$ for double and of the form $\theta y_Q + t_2$ for addition. So we have

$$\begin{aligned}
f_{2i,P}(Q) &\equiv (\alpha_{i,P} + \theta y_Q \beta_{i,P})^2(\theta y_Q + t_1)\\
&\equiv ((\alpha_{i,P}^2 - y_Q^2\beta_{i,P}^2)t_1 - 2\alpha_{i,P}\beta_{i,P}y_Q^2) + \theta y_Q(\alpha_{i,P}^2 - y_Q^2\beta_{i,P}^2 + 2\alpha_{i,P}\beta_{i,P}t_1)\\
&\equiv \alpha_{2i,P} + \theta y_Q \beta_{2i,P};\\
f_{2i+1,P}(Q) &\equiv (\alpha_{2i,P} + \theta y_Q \beta_{2i,P})(\theta y_Q + t_2)\\
&\equiv (\alpha_{2i,P}t_2 - y_Q^2\beta_{2i,P}) + \theta y_Q(\alpha_{2i,P} + \beta_{2i,P}t_2)\\
&\equiv \alpha_{2i+1,P} + \theta y_Q \beta_{2i+1,P}.
\end{aligned}$$

This completes the proof. □

*Remark 3.* The above Lemma can be applied to any pairing algorithm based on Miller's algorithm. Specifically, it easily applies to the pairing computation using Jacobian coordinates [12] and Tate pairing using an efficient endomorphism [25]. The extension of our algorithm to these two methods is discussed in Appendix B and Appendix C respectively and timings are given in Tables 3, 4 and 5.

## 6 Implementation Results and Efficiency Comparision

In this section, efficiency analysis and implementation results of our proposed algorithms are given.

We consider the case of elliptic curves over $\mathbb{F}_p$ with embedding degree 2 at the 80-bit security level, which means $p$ has 512 bits and $n$ has 160 bits. The merits of using elliptic curves of embedding degree 2 is discussed by Scott in [26]. We assume that the point order $n$ has low Hamming weight. Therefore, only doubling steps in the Miller loop are considered.

Two concrete elliptic curves are used. Let $E1$ be the supersingular elliptic curve given in Appendix A, and $E2$ be the non-supersingular elliptic curve given in Appendix C. Note that both of these curves are given in Miracl [23] and that $p \equiv 3 \mod 4$ in both cases. The reason why we consider two types of elliptic curves is that the Tate pairing using endomorphism requires to work on certain type of non-supersingular curves. Since we are working with $k = 2$ curves over $\mathbb{F}_p$ we do not consider the ate pairing.

To obtain a consistent comparison of running times we implemented all the algorithms in Magma [9]. The purpose of the running times is only to show the relative costs of the methods. Of course faster running times could be obtained on our platform by using lower-level software libraries.

## 6.1 Tate Pairing Using $x$-Coordinates Only

In this scenario, only $x$-coordinates of $P$ and $Q$ are given.

To compute the $\mathrm{Tr}(t_n(P, Q))$, one can first extract two square roots and then perform the conventional Miller algorithm. We refer this as conventional method. Using curve $E1$ the computation cost of the overall computation is approximately estimated as follows.

A binary method for computing the square root would cost about $792M$ (as $(p + 1)/4$ is of 510 bits and hamming weight 282). The length of Miller loop is 159(n is of 160 bit). The hard part of the final exponentiation using Lucas sequences costs about $2 * 353M$ ($(p + 1)/r$ is of 353 bit). Assuming $1I = 10M$, the overall cost is about $792 * 2 + 20 * 159 + 2 * 353 = 5470M$.

For our new algorithm, each doubling in the Miller loop costs about $14M+1I$ (compared with $10M+I$ for the conventional Miller algorithm using affine coordinates). Hence, the Miller algorithm costs about $4*159 = 636M$ more, but we save $1584M$ by not needing to compute two square roots. The final exponentiation is the same. The following table summarizes the analysis and the implementation result.

**Table 2.** Efficiency Comparision 1

| Method | Operation | Time(ms) |
|---|---|---|
| Conventional($A$) | 5470M | 12.7 |
| New($A$) | 4522M | 11.0 |

This suggests that our approach is about 15% faster.

## 6.2 Tate Pairing Using $x_P$, $y_P$ and $x_Q$

In this scenario, $x_P$, $y_P$ and $x_Q$ are given.

To compute $\mathrm{Tr}(t_n(P, Q))$ using conventional methods, one first needs to extract one square root and then perform the conventional Miller algorithm. We divide the conventional methods into three cases as below, thus the cost can be approximately estimated separately.

**Affine Coordinates** We use the elliptic curve $E1$. One square root extraction costs about $792M$. Every doubling step in the Miller loop costs $10M + 1I$. The final exponentiation costs about $2 * 353M$. Assuming $1I = 10M$, the overall cost is about $792 + 20 * 159 + 2 * 353 = 4678M$.

For our new algorithm, each doubling step costs about $12M + 1I$, thus it costs about $2 * 159M$ more in the Miller loop. But our algorithm save $792M$ for taking square root. The final exponentiation is the same. The following table summarizes the estimation and the implementation result.

This suggests that our approach is about 11% faster.

**Table 3.** Efficiency Comparision 2

| Method | Operation | Time(ms) |
|---|---|---|
| Conventional($A$) | 4678M | 11.2 |
| New($A$) | 4204M | 9.7 |

**Jacobian Coordinates** We consider the elliptic curve $E1$. The costs of computing square roots and performing the final exponentiation are the same as the affine coordinate case. Every doubling step in the Miller loop costs about $19M$ using the method in [12]. So the overall cost is about $4519M$

For our new algorithm, each double step costs about $21M$, thus it costs about more $2 * 159M$ in the Miller loop. And the saving of taking square root is $792M$. The final exponentiation is the same.

The following table summarizes the estimation and the implementation result.

**Table 4.** Efficiency Comparision 2

| Method | Operation | Time(ms) |
|---|---|---|
| Conventional($J$)[12] | 4519M | 11.1 |
| New($J$) | 4045M | 9.8 |

This suggests that our approach is about 11% faster.

**Tate Pairing Using an Efficient Endomorphism** We consider the ordinary elliptic curve $E2$. The square root extraction costs $510M + 242M = 752M$. The operation count of Miller's algorithm and final exponentiation is $3329M$, which was given in [25]. So this method costs about $3329M + 752M = 4081M$.

It's not hard to find that, our new algorithm would cost $20M + I$ at each doubling in the Miller algorithm, which is $4M$ more then the method above. There are about 80 doubling step in the concrete curve considered. So the new algorithm cost more $320M$ in the Miller algorithm, but save $752M$ in the square root extraction. The final exponentiation is the same.

The following table summarizes the estimation and the implementation result.

**Table 5.** Efficiency Comparision 2

| Method | Operation | Time(ms) |
|---|---|---|
| Conventional($J$)[12] | 4081M | 11.1 |
| New($J$) | 3649M | 10.4 |

This suggests that our approach is about 10% faster.

## 6.3 Summary of Efficiency Analysis

Our approach can be applied to all known pairing algorithms when the embedding degree is even and point compression is being used. Applying our new algorithm to the conventional methods for pairings on elliptic curves over $\mathbb{F}_p$ with embedding degree 2, the overall pairing computation is about $10 - 15\%$ faster than their analogous methods. This leads to the following recommendation.

- In the case only $x$-coordinates of $P$ and $Q$ are given, such as the Joux's three party key agreement scheme [19], it is better to use the algorithm in Lemma 4 when the ratio between inversion and multiplication does not exceed 16. Otherwise, it is better to use the algorithm in Lemma 5.
- In the case $x_P$, $y_P$ and $x_Q$ are given, such as the ID-KEM schemes of [1] and [8, 10] it is better to use the algorithm in Lemma 5. Again, the choice between the coordinate systems depends on the ratio of inversion and multiplication.

## 7 Conclusions

We have shown how to efficiently compute the trace of the Weil and Tate pairings using only the $x$-coordinates of points. We have discussed the application of these ideas to pairing computation when elliptic curve point compression is used. The new methods to compute the compressed Tate pairing are about 10%-15% faster then their their counterparts.

We stress that this issue is relevant only when working with small embedding degrees, such as embedding degree 2. In particular, our results apply to the scenario studied by Scott [26]. For example, if using BN curves with $k = 12$ then recovering the $y$-coordinate of a point only requires square roots in $\mathbb{F}_p$ for the point on the small field and square roots in $\mathbb{F}_{p^2}$ for the point on the sextic twist curve. Hence, one would not use the methods of our paper in cases like this.

## References

1. M. Barbosa, L. Chen, Z. Cheng, M. Chimley, A. Dent, P. Farshim, K. Harrison, J. Malone-Lee, N. P. Smart, F. Vercauteren, SK-KEM: An Identity-Based KEM, submission to IEEE P1363.3 ID-based PKC. Available from `http://grouper.ieee.org/groups/1363/IBC/submissions/index.html`
2. P.S.L.M. Barreto, S. Galbraith, C. ÓhÉigeartaigh, and M. Scott. *Efficient pairing computation on supersingular abelian varieties.* Designs, Codes and Cryptography. Volume 42, Number 3, pp.239-271, Springer-Verlag, 2007.
3. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. *Efficient algorithms for pairing-based cryptosystems.* Crypto'2002, LNCS 2442, pp.354-368, Springer-Verlag, 2002.
4. P.S.L.M. Barreto and M. Naehrig. *Pairing-Friendly Elliptic Curves of Prime Order.* SAC'2005, LNCS 3897, pp.319-331, Springer-Verlag, 2006.
5. I. F. Blake, G. Seroussi and N. P. Smart, *Elliptic curves in cryptography*, Cambridge 1999.
6. D. Boneh and M. Franklin. *Identity based encryption from the Weil Pairing.* Crypto 2001, LNCS 2139, pp213-299, Springer Verlag, 2002.
7. D. Boneh, H. Shacham, and B. Lynn. *Short signatures from the Weil pairing.* Journal of Cryptology, Vol. 17, No. 4, pp. 297-319, 2004
8. D.Boneh and X. Boyen. *Efficient selective-ID secure identity based encryption without random oracles.* EUROCRYPT 2004, LNCS 3027, pp.223-238, Springer-Verlag, 2004.
9. W. Bosma, J. Cannon and C. Playoust, The Magma Algebra System I: The User Language, *J. Symbolic Computation*, **24** (1997) 235-265.
10. X. Boyen. *The BB1 Identity-based cryptosystem: A standard for Encryption and Key Encapsulation.* Submissions for IEEE P1363.3: Identity-Based Public Key Cryptography
11. J. W. S. Cassels, *Lectures on elliptic curves*, Cambridge (1991)
12. S. Chatterjee, P. Sarkar, and R. Barua *Efficient Computation of Tate Pairing in Projective Coordinate Over General Characteristic Fields.* ICISC 2004, LNCS 3506, pp. 168-181, 2005.
13. R. Crandall and C. Pomerance, Prime numbers A Computational Perspective, Springer (2nd ed.), 2005.
14. D. Freeman, M. Scott, and E. Teske. *A Taxonomy of pairing–friendly elliptic curves.* Cryptology ePrint Archive, Report 2006/372 .
15. S. Galbraith, H. Hopkins and I. Shparlinski, *Secure Bilinear Diffie-Hellman Bits*, in H. Wang, J. Pieprzyk and V. Varadharajan (eds.), ACISP 2004, Springer LNCS 3108 (2004) 370–378. Earlier version in eprint archive 2002/155.
16. R. Granger, D. Page, M. Stam, *On Small Characteristic Algebraic Tori in Pairing Based Cryptography*, LMS JCM 9, pp. 64-85, 2006.
17. F. Hess, N.P. Smart, and F. Vercauteren. *The Eta Pairing Revisited.* IEEE Transactions on Information Theory, vol 52, pp. 4595-4602, Oct. 2006. Also available from http://eprint.iacr.org/2006/110.

18. T. Izu and T. Takagi. *Efficient computation of the Tate pairing for the Large MOV degree.* ICISC 2002, LNCS 2587, pp. 283-297, Springer-Verlag, (2003).
19. A. Joux, A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17 (4) : 263–276, 2004.
20. M. Joye and J. J. Quisquater. *Efficient computation of full Lucas sequences.* Electronics Letters, 32(6):537C538, 1996.
21. N. Kobilitz and A. Menezes. *Pairing-Based Cryptography at High Security Levels.*
22. Peter. L. Montgomery. *Speeding the Pollard and Elliptic Curve Methods of Factorization.* Mathematics of Computation. 1987, 48: 243-264.
23. *Multiprecision Integer and Rational Arithmetic C/C++ Library.* http://www.shamus.ie/index.php.
24. M. Naehrig and P.S.L.M. Barreto, On compressible pairings and their computation, eprint 2007/429.
25. M. Scott, *Faster pairings using an elliptic curve with an efficient endomorphism.* Indocrypt 2005, LNCS 3797, pp.258-269, Springer Verlag, 2005
26. M. Scott, *Computing the Tate Pairing.* CT-RSA 2005, LNCS 3376, pp 293-304, Springer-Verlag, 2005.
27. M. Scott and P. S. L. M. Barreto, *Compressed Pairings.* Crypto'2004, LNCS 3152, pp140-156, Springer-Verlag, 2004.

## A The Supersingular Elliptic Curve from Miracl

The supersingular elliptic curve with $k = 2$ provided by the Miracl is of the form $y^2 = x^3 + ax + b$, where $ab \neq 0$. Here, using the fact that $p \equiv 3 \bmod 4$, we easily obtain the supersingular elliptic curve $E1$: $y^2 = x^3 + x$ over $\mathbb{F}_p$, where

- $p =$ 8BA2A5229BD9C57CFC8ACEC76DFDBF3E3E1952C6B3193
  ECF5C571FB502FC5DF410F9267E9F2A605BB0F76F52A7
  9E8043BF4AF0EF2E9FA78B0F1E2CDFC4E8549B
- $n =$ 80000000000000000000000000000000000020001

## B The Compressed Tate Pairing using Jacobian Coordinates

We use the method of S. Chatterjee et al. in [12] as building block. Here, we briefly recall their method. For consistence, we use their description for the notation and algorithm. The elliptic curve considered is $y^2 = x^3 + ax$.

### B.1 Encapsulated Point Doubling and Line Computation

Let $P = (X_1, Y_1, Z_1)$ correspond to $(X_1/Z_1^2, Y_1/Z_1^3)$ in affine coordinate. Let $((X_3, Y_3, Z_3)) = 2P$. Then we have

$$t_1 = Y_1^2; \quad t_2 = 4X_1t_1; \quad t_3 = 8t_1^2; \quad t_4 = Z_1^2; \quad t_5 = 3X_1^2 + aZ_1^4;$$
$$X_3 = t_5^2 - 2t_2; \quad Y_3 = t_5(t_2 - X_3) - t_3; \quad Z_3 = 2Y_1Z_1;$$

The line valuation at $Q' = (-x_Q, \theta y_Q)$ is

$$g_{P,P}(-x_Q, \theta y_Q) = Z_3 t_4 y_Q \theta - (2t_1 - t_5(t_4 x_Q + X_1))$$

.

In the Miller algorithm, $f_1$ is updated by

$$f_1 = f_1^2 * g_{P,P}(-x_Q, \theta y_Q)$$

.

### B.2 Encapsulated(Mixed) Point Addition and Line Computation

Given $R = (X_1, Y_1, Z_1)$ and $P = (X, Y, Z)$ we compute $R + P = (X_3, Y_3, Z_3)$ as follows,

$$\begin{gathered} t_1 = Z_1^2; \quad t_2 = Z_1 t_1; \quad t_3 = X t_1; \\ t_4 = Y t_2; \quad t_5 = t_3 - X_1; \quad t_6 = t_4 - Y_1; \\ t_7 = t_5^2; \quad t_8 = t_5 t_7; \quad t_9 = X_1 t_7; \\ X_3 = t_6^2 - (t_8 + 2t_9); \quad Y_3 = t_6(t_9 - X_3) - Y_1 t_8; \quad Z_3 = Z_1 t_5; \end{gathered}$$

The line valuation at $Q' = (-x_Q, \theta y_Q)$ is

$$g_{R,P}(-x_Q, \theta y_Q) = Z_3 y_Q \theta - (Z_3 Y - t_6(x_Q + X))$$

Then $f_1$ is updated by

$$f_1 = f_1 * g_{R,P}(-x_Q, \theta y_Q)$$

.

### B.3 New Algorithm in the Jacobian Coordinate Case

Equipped with the above formulas for line valuations, we can rewrite Lemma 5 in the case of Jacobian coordinate. The formulas for $\alpha_{i,P}$ and $\beta_{i,P}$ is given below.

$$\begin{aligned} f_{2i,P}(Q) &= (\alpha_{i,P} + \theta y_Q \beta_{i,P})^2 (s_1 + s_2 \theta y_Q) \\ &= (m_1 + m_2 y_Q \theta)(s_1 + s_2 \theta y_Q) \\ &= \alpha_{2i,P} + \theta y_Q \beta_{2i,P}; \end{aligned}$$

where $m_1 = \alpha_{i,P}^2 - \beta_{i,P}^2 y_Q^2$, $m_2 = 2\alpha_{i,P}\beta_{i,P}$, hence $\alpha_{2i,P} = m_1 s_1 - m_2 s_2 y_Q^2$, and $\beta_{2i,P} = m_1 s_2 + m_2 s_1$. Similarly,

$$\begin{aligned} f_{2i+1,P}(Q) &= (\alpha_{2i,P} + \theta y_Q \beta_{2i,P})(s_3 + s_4 \theta y_Q) \\ &= (m_1 + m_2 \theta y_Q)(s_3 + s_4 \theta y_Q) \\ &= \alpha_{2i+1,P} + \theta y_Q \beta_{2i+1,P} \end{aligned}$$

where $m_1 = \alpha_{2i,P}$, $m_2 = \beta_{2i,P}$, hence $\alpha_{2i+1,P} = m_1 s_3 - m_2 s_4 y_Q^2$, and $\beta_{2i+1,P} = m_1 s_4 + m_2 s_3$.

Note that, during the implementation, we can use the Karatsuba style formula to compute $\alpha_{i,P}$ and $\beta_{i,P}$. The trick can save one base field multiplication in each Miller step in the above computation.

## C Faster Tate Pairing Using an Efficient Endomorphism

In [25], Scott proposed a new method to compute the Tate pairing using efficient endomorphisms. The method is recalled as follows.

The example curve $E2$ considered is defined by $y^2 = x^3 + 5$ over $\mathbb{F}_p$, where,

- $p =$ DAC2F97CDD22AC93CCC12106F6541B748C9D8F71C806B
  1023B95D69281EB5F739F9A3EFC931882113CA321A0AC
  348D825249B44E45C180726EC6E896E6DE568B
- $\lambda = 2^{80} + 2^{16} =$ 100000000000000010000(hex)
- $n = \lambda^2 + \lambda + 1 =$ 10000000000000002000100000000000100010001(hex)

– $\beta$=704027DC704986DC115B5DBEE212B29F21C650042202F
B5790BB520E4970BCE603FF01F4F0B509A0DB16332424
6EE01848A1A47FDB43C9D7BC041237FFE2CE2C

Here, $\beta$ is the non-trivial cubic root of unity in $\mathbb{F}_p$ which is needed for efficient endomorphism.

**Algorithm 3: Faster Tate Algorithm**

```
Input: $P, Q$, $E$, $\lambda = 2^a + 2^b$, $a > b$
Output: $t_n(P, Q)$
1 $A \leftarrow P$, $f_1 \leftarrow 1$, $f_2 \leftarrow 1$, $j \leftarrow 1$
2 for $i \leftarrow 1$ to $a - b$ do
3 $f_1 \leftarrow f_1^2 \cdot g(A, A, Q, 0)$
4 $f_2 \leftarrow f_2^2 \cdot s[0]$
5 end for
6 $f_1 \leftarrow f_1 \cdot g(A, P, Q, 0)$
7 $f_2 \leftarrow f_2 \cdot s[0]$
8 for $i \leftarrow 1$ to $b$ do
9 $f_1 \leftarrow f_1^2 \cdot g(A, A, Q, 0)$
10 $f_2 \leftarrow f_2^2 \cdot s[0]$
11 end for
12 $f_1 \leftarrow f_1 \cdot g(A, P, Q, -)$
13 $f_1 \leftarrow f_1^\lambda \cdot f_2$
14 return $f^{(p-1)(p+1)/r}$
```

**Algorithm 4: Function $g(.)$**

```
Input: $A, B, Q, i$
1 $x_i, y_i \leftarrow A$
2 $x_Q, y_Q \leftarrow Q$
3 $m_i = A.add(B)$ (Add $B$ to $A$ and return slope)
4 store $-y_Q - y_i - m_i(\beta x_Q - x_i)$ in an array element $s[i]$
5 return $y_Q - y_i - m_i(x_Q - x_i)$
```

Note that this is for the computation of the whole Tate pairing. A slight change of the final steps can easily lead to the compressed pairing.

Through algorithm 3 and 4, it is easy to see that the update of $f_1$ and $f_2$ using line valuations can be computed in the similar fashion of Lemma 5 without the value of $y_Q$. If we only care about the compressed Tate pairing, than we simply replace line 13 and 14 with the Lucas sequence.

The computation in each step of loop costs about $16M + I$. Applying our algorithm to this method, each step would cost about $20M + I$.

This clarifies that our new algorithm applies to the Tate pairing using efficient endomorphism.