Refinements of Miller's Algorithm for Computing Weil/Tate Pairing

Ian Blake, Kumar Murty and Guangwu Xu
University of Toronto

October, 2003

Abstract

In this paper we propose three refinements to Miller's algorithm for computing Weil/Tate Pairing. The first one is an overall improvement and achieves its optimal behavior if the binary expansion of the involved integer has more zeros. If more ones are presented in the binary expansion, second improvement is suggested. The third one is especially efficient in the case base three. We also have some performance analysis.

KEYWORDS: ALGORITHM, ELLIPTIC CURVE, CRYPTOGRAPHY, WEIL/TATE PAIRING

1 Introduction

The Weil and Tate pairings are nondegenerate bilinear maps on certain pairs of points on elliptic curves to a multiplicative subgroup of an appropriate order in a certain finite field.

The first notable application of pairings to cryptography was the work of Menezes et al [10] who showed that the Weil pairing on supersingular elliptic curves (whose Frobenius trace is divisible by the characteristic of the field of curve definition) can be used to imbed the discrete logarithm problem on the elliptic curve into a discrete logarithm problem on a certain subgroup of a suitable extension of the finite field of the curve definition. The complexity of the logarithm problem on the curve is often (for non-supersingular curves) assumed to have a complexity on the order of the square root of the group order while that in finite fields is of subexponential complexity. The work showed that discrete logarithm problems on supersingular curves are unsuitable for many cryptographic applications. This represented a dramatic lesson on the caution required in choosing such curves. Frey and Rück [5] also consider this situation using the Tate pairing, which has certain advantages.

Recent work on parings in cryptography has considered their use in the definition and implementation of certain new and potentially very useful protocols. Boneh and Franklin [3] used pairings to develop an efficient identity-based encryption (IBE) system, the first such system since the notion of IBE was first described by Shamir [14]. In such a system a user's public information such as his identity, email address etc.

can be used as their public key with their secret key being derived by a central authority possessing certain additional information on the curve. Since then pairings have been used to define numerous interesting protocols including the identity-based key exchange and signature schemes by Sakai, Ohgishi and Kashahara[13], the one round protocol for tripartite Diffie-Hellman key exchange by Joux[7], the short signature scheme by Boneh, Lynn and Shacham[4], and many others. Interestingly these protocols invariably require the use of supersingular curves. The working of the protocol depends on the properties of the pairing while the security of the protocol depends on the difficulty of the discrete logarithm problem in the multiplicative group of an extension of the finite field which must then be chosen sufficiently large.

Thus the computation of Weil and Tate pairings is an important issue for such protocols which has attracted attention. The original algorithm for computing pairings is due to Miller[11] and most current algorithms are based in some manner on it. It is an efficient probabilistic polynomial-time algorithm for computing the pairings. What the algorithm does is the evaluation of a rational function associated with an n-torsion point of the ellicptic curve. The work of Barreto, Kim, Lynn and Scott[1] and Galbraith, Harrison and Soldera[6] focus in particular on the Tate pairing and they propose methods for its fast computation. They also consider a practically useful case of fields of characteristic three. In [8], Eisenträger, Lauter and Montgomery give an algorithm to speed up point multiplication of an elliptic curve. Using their method, $H(n)$ field multiplications and $H(n)$ field squarings is eliminated when performing scalar multiplication of nP for some point P, where $H(n)$ denotes the Hamming weight of the number n, i.e., the number of one bits in the binary expansion of n. This observation, combined with a parabola substitution, enables them to get an improvement to Miller's algorithm for general elliptic curves. In the framework of this paper, their improvement requires $H(n)$ fewer field multiplications in addition. All of these contributions use ideas very different than those used in this work.

In this paper, we present three versions of improvement to Miller's algorithm. They apply to general elliptic curves. Version 1 is efficient in any case and $log_2 n$ field multiplications are saved. In particular, this improvement includes some practically interesting cases (for example, when n is a Solinas number of the form $2^a \pm 2^b \pm 1$, see [1]) that our version 2 and the algorithm of [8] are not strong enough to deal with. Version 2 gains more saving in the case where n has relatively high Hamming weight, to be more specific, $2H(n)$ field multiplications are removed. It is remarked that the technique of Eisenträger, Lauter and Montgomery does not apply here. However some modification can be still made to improve Miller's algorithm to save $H(n)$(instead of $2H(n)$) field multiplications. Moreover, with this modification, we are able to use the method of Eisenträger, Lauter and Montgomery in point multiplication and save $H(n)$ field multiplications and $H(n)$ field squarings. The third one is especially useful for the field of characteristic three where it saves $log_3 n$ field multiplications compared to the original algorithm in characteristic three. It is noticed that in this case the point tripling can be made very efficient.

The work is organized as follows. After introducing the pairings and Miller's algorithm briefly in section 2, some basic facts and observations on elliptic curves are presented in section 3. In section 4, we use the results from section 3 to simplify some formulas used by Miller's algorithm and get three improved versions of the algorithm. In section 5, some detailed analysis of the three versions is given.

2 Weil Pairing, Tate pairing and Miller's Algorithm

Let E/K be an elliptic curve. Recall that a *divisor* is an element of the free abelian group (denoted by $\mathrm{Div}(E)$) generated by the set of points of $E(\overline{K})$. Given a divisor $D = \sum_{P \in E} n_P(P)$, the *degree* of D is defined by $\deg(D) = \sum_{P \in E} n_P$. We are interested in the subgroup of *divisors of degree* 0, namely $\mathrm{Div}^0(E) = \{D \in \mathrm{Div}(E) : \deg(D) = 0\}$. For a nonzero rational function f over E, we define $\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}_P(f)(P)$. It turns out that $\mathrm{div}(f)$ is an element in $\mathrm{Div}^0(E)$ and is called a *principal divisor*. A characterization of principal divisors is: $D = \sum_{P \in E} n_P(P) \in \mathrm{Div}^0(E)$ is principal iff $\sum_{P \in E} n_P P = \mathcal{O}$ where $\mathcal{O}$ is the point at infinity. The relation $\sim$ on $\mathrm{Div}^0(E)$ is defined to be $D_1 \sim D_2$ iff $D_1 - D_2$ is principal.

The *support* of a divisor $D = \sum_{P \in E} n_P(P)$ is the set of points P with $n_P \neq 0$. If f is a nonzero rational function such that $\mathrm{div}(f)$ and D have disjoint supports, we can extend the evaluation of f at D by defining $f(D) = \Pi_{P \in E} f(P)^{n_P}$.

Let n be an integer which is prime to p =char(K) if $p > 0$, and $E[n] = \{P \in E(\overline{K}) : nP = \mathcal{O}\}$. Take $P, Q \in E[n]$, there exist $D_P, D_Q \in \mathrm{Div}^0(E)$ such that $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. Then there exist functions f_P, f_Q such that $\mathrm{div}(f_P) = nD_P$, $\mathrm{div}(f_Q) = nD_Q$. Suppose that D_P and D_Q have disjoint supports, then the following is meaningful:

$$e(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)},$$

and this is the *Weil pairing*.

The *Tate pairing* can also be defined based on $f_P(D_Q)$. By a suitable field extension if necessary, we may assume that the field K contains nth roots of unity. Let $P \in E(K)[n]$ and $Q \in E(K)$. As before, there exits a function f_P such that $\mathrm{div}(f_P) = n(P) - n(\mathcal{O})$. Take a point $S \in E$ such that $D_Q = (Q + S) - S$ and f_P have disjoint supports. Then we have a map

$$\phi_n : E(K)[n] \times (E(K)/nE(K)) \to K^*/(K^*)^n$$

with

$$\phi_n(P, \overline{Q}) = \overline{f_P(D_Q)},$$

where $\overline{Q}$ is the equivalence class in $E(K)/nE(K)$ containing Q, and $\overline{f_P(D_Q)}$ is the equivalence class in $K^*/(K^*)^n$ containing $\overline{f_P(D_Q)}$. The function ϕ_n is called Tate pairing.

An essential part in computing the Weil/Tate pairing is the evaluation of $f_P(R)$ for each point R in the support of D_Q. In his unpublished manuscript, Miller gave an elegant and efficient algorithm for this calculation. The main idea of Miller's algorithm is as follows. Randomly pick a point R, and let $D_P = (P+R) - (R)$. For each integer k, there is a rational function f_k such that

$$\mathrm{div}(f_k) = k(P+R) - k(R) - (kP) + (\mathcal{O}).$$

In particular, $f_n = f_P$.

For any points S, T, let $h_{S,T}$ and h_S be linear functions such that $h_{S,T} = 0$ and $h_S = 0$ are the line passing through S, T and the vertical line passing through S respectively.

Notice that

$$\mathrm{div}(h_{k_1P,k_2P}) = (k_1P) + (k_2P) + (-(k_1+k_2)P) - 3(\mathcal{O})$$

and

$$\mathrm{div}(h_{(k_1+k_2)P}) = ((k_1+k_2)P) + (-(k_1+k_2)P) - 2(\mathcal{O}),$$

and we have

$$\mathrm{div}(f_{k_1+k_2}) = \mathrm{div}(f_{k_1}) + \mathrm{div}(f_{k_2}) + \mathrm{div}(h_{k_1P,k_2P}) - \mathrm{div}(h_{(k_1+k_2)P}),$$

and hence

$$f_{k_1+k_2} = \frac{f_{k_1} f_{k_2} h_{k_1P,k_2P}}{h_{(k_1+k_2)P}}. \tag{2.1}$$

This is a recursive equation with initial conditions $f_0 = 1$ and $f_1 = \frac{h_{P,R}}{h_{P+R}}$. The latter is because $\mathrm{div}(f_1) = (P+R) - (R) - (P) + (\mathcal{O})$.

Miller's algorithm is given more formally below and its simliarity to the algorithms in [1, 3] is noted:

Algorithm 2.1 *(Miller's algorithm)*
`INPUT:` *Integer* $n = \sum_{i=0}^{t} b_i 2^i$ *with* $b_i \in \{0,1\}$ *and* $b_t = 1$, *and a point* $S \in E$
`OUTPUT:` $f = f_n(S)$.

```
  $f \leftarrow f_1$; $Z \leftarrow P$;
  For  $j \leftarrow t-1, t-2, \dots, 1, 0$ do
    $f \leftarrow f^2 \frac{h_{Z,Z}(S)}{h_{2Z}(S)}$; $Z \leftarrow 2Z$;
    If  $b_j = 1$  then
      $f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$; $Z \leftarrow Z + P$;
    Endif
  Endfor

Return $f$
```

As indicated in [1], when we consider Tate pairing, the function f_k can be choosen so that

$$\mathrm{div}(f_k) = k(P) - (kP) + (k-1)(\mathcal{O}).$$

In this case the above Miller's algorithm remains the same except $f_1 = 1$.

For fields of characteristic three, the following version of Miller's algorithm is more efficient, taking advantage of the fast implementation of point triples in such a case (see [1, 6]):

Algorithm 2.2 *(Miller's algorithm in characteristic three)*
`INPUT:` *Integer* $n = \sum_{i=0}^{r} t_i 3^i$ *with* $t_i \in \{0,1,2\}$ *and* $t_r \neq 0$, *and a point* $S \in E$
`OUTPUT:` $f = f_n(S)$.

```
If  $t_r = 1$  then
  $f \leftarrow f_1$; $Z \leftarrow P$;
Endif
If  $t_r = 2$  then
  $f \leftarrow {f_1}^2 \frac{h_{P,P}(S)}{h_{2P}(S)}$; $Z \leftarrow 2P$;
Endif
For   $j \leftarrow r-1, r-2, \ldots, 1, 0$ do
  $f \leftarrow f^3 \frac{h_{Z,Z}(S)}{h_{2Z}(S)} \frac{h_{2Z,Z}(S)}{h_{3Z}(S)}$; $Z \leftarrow 3Z$;
  If  $t_j = 1$ then
    $f_1 f \leftarrow f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$; $Z \leftarrow Z + P$;
  Endif
  If  $t_j = 2$ then
    $f \leftarrow {f_1}^2 f \frac{h_{Z,2P}(S)}{h_{Z+2P}(S)}$; $Z \leftarrow Z + 2P$;
  Endif
Endfor


Return $f$
```

3 Preliminary Observations and Facts

Some well known facts and observations that can be used to simplify computations in Miller's algorithm are noted in this section.

Consider the elliptic curve E of the form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

For a linear function

$$h(x, y) = k(x - a) + b - y$$

on E, where a, b and k are constants, define $\overline{h}(x, y)$ as the *conjugate* of h as follows:

$$\overline{h}(x, y) = k(x - a) + b + y + a_1x + a_3.$$

Note that for a point $R \in E$, $\overline{h}(R) = h(-R)$. Also note that the product $h(x, y)\overline{h}(x, y)$ is exactly the norm $N_{K(x,y)/K(x)}(h)$.

The following fact will be useful. It is apparently well known although no proof of it in the literature was found.

Lemma 3.1 *If the line $h(x,y)=0$ intersects with E at points $P=(a,b), Q=(c,d)$ and $-(P+Q)$ with $P+Q=(\alpha,\beta)$, then*

$$N_{K(x,y)/K(x)}(h) = -(x-a)(x-c)(x-\alpha).$$

Proof: Notice that $N_{K(x,y)/K(x)}(h)$ can be reduced to a function of the form

$$-x^3+t_2x^2+t_1x+t_0,$$

where $t_0,t_1,t_2\in K$. Moreover, each of $(x-a),(x-c)$ and $(x-\alpha)$ is a factor of $N_{K(x,y)/K(x)}(h)$, and so the desired factorization follows.
[]
For a point $Q\in E$, we write $Q=(x_Q,y_Q)$, i.e., x_Q is the $x-$coordinate of Q and y_Q the $y-$coordinate of Q.
The following observations will play a key role in the refinements of Miller's algorithm.

Lemma 3.2 *Let $Q\in E[n]$ and $S\neq Q,2Q,\cdots,nQ$. Then*

1.
$$\frac{h_{Q,Q}(S)}{h_Q^2(S)h_{2Q}(S)} = -\frac{1}{h_{Q,Q}(-S)}.$$

2. *For any integer k,*
$$\frac{h_{(k+1)Q,kQ}(S)}{h_{(k+1)Q}(S)h_{(2k+1)Q}(S)} = -\frac{h_{kQ}(S)}{h_{(k+1)Q,kQ}(-S)}.$$

3.
$$\frac{h_{Q,Q}(S)h_{2Q,Q}(S)}{h_{2Q}(S)h_{3Q}(S)} = -\frac{h_{Q,Q}(S)h_Q(S)}{h_{2Q,Q}(-S)}.$$

Proof: By Lemma 3.1, we have

1.
$$\begin{aligned}\frac{h_{Q,Q}(S)}{h_Q^2(S)h_{2Q}(S)} &= \frac{h_{Q,Q}(S)h_{Q,Q}(-S)}{h_Q^2(S)h_{2Q}(S)h_{Q,Q}(-S)}\\ &= \frac{N_{K(x,y)/K(x)}(h_{Q,Q})(S)}{(x_S-x_Q)^2(x_S-x_{2Q})h_{Q,Q}(-S)}\\ &= -\frac{1}{h_{Q,Q}(-S)}.\end{aligned}$$

2.

$$\begin{aligned}\frac{h_{(k+1)Q,kQ}(S)}{h_{(k+1)Q}(S)h_{(2k+1)Q}(S)} &= \frac{h_{(k+1)Q,kQ}(S)h_{(k+1)Q,kQ}(-S)}{h_{(k+1)Q}(S)h_{(2k+1)Q}(S)h_{(k+1)Q,kQ}(-S)} \\ &= \frac{N_{K(x,y)/K(x)}(h_{(k+1)Q,kQ})(S)}{(x-x_{(k+1)Q})(x-x_{(2k+1)Q})h_{(k+1)Q,kQ}(-S)} \\ &= \frac{h_{kQ}(S)}{h_{(k+1)Q,kQ}(-S)}.\end{aligned}$$

3.

$$\begin{aligned}\frac{h_{Q,Q}(S)h_{2Q,Q}(S)}{h_{2Q}(S)h_{3Q}(S)} &= \frac{h_{Q,Q}(S)h_{2Q,Q}(S)h_{2Q,Q}(-S)}{h_{2Q}(S)h_{3Q}(S)h_{2Q,Q}(-S)} \\ &= \frac{h_{Q,Q}(S)N_{K(x,y)/K(x)}(h_{2Q,Q})(S)}{(x_S-x_{2Q})(x_S-x_{3Q})h_{2Q,Q}(-S)} \\ &= -\frac{h_{Q,Q}(S)(x_S-x_Q)}{h_{2Q,Q}(-S)} \\ &= -\frac{h_{Q,Q}(S)h_Q(S)}{h_{2Q,Q}(-S)}.\end{aligned}$$

[]

Remark 3.3 1. *Since div(f) = div(cf) for any nonzero constant $c \in K$, the sign does not affect the calculation of either pairing and therefore, minus signs will be omitted in the use of the above lemma.*

2. *In the rest of the discussion, the point $P \in E[n]$ will be fixed and Q is taken to be some multiple of P. In order to satisfy the condition of the lemma, it is sufficient to let $S \neq P, 2P, \ldots, nP$. This is also the requirement of the original Miller algorithm.*

4 The Refinements

Notice that Miller's algorithm 2.1 uses the double-and-add method, and we can display an explicit formula for the function f_n as

$$f_n = f_1^n \Pi_{i=t}^1 \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P}}{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-1}}, \qquad (4.1)$$

where $t = \lfloor \lg_2 n \rfloor$ and $b_{i-1} = \lfloor \frac{n}{2^{i-1}} \rfloor - 2\lfloor \frac{n}{2^i} \rfloor$. In this formula, if $b_{i-1} = 0$, then $h_{2\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P} = h_{2\lfloor \frac{n}{2^i} \rfloor P, \mathcal{O}} = h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}$. Without loss of generality, we also assume that $h_{\mathcal{O}} = 1$. We arrange the product to start from term t down to term 1. This is the way that the algorithm works.

Similarly, in the case of base three, f_n can be expressed as

$$f_n = f_1^n \left(\frac{h_{(3-\lfloor \frac{n}{3^r} \rfloor)P,(\lfloor \frac{n}{3^r} \rfloor -1)P}}{h_{2P}} \right)^{3^r} \Pi_{i=r}^{1} \left(\frac{h_{\lfloor \frac{n}{3^i} \rfloor P, \lfloor \frac{n}{3^i} \rfloor P}}{h_{2\lfloor \frac{n}{3^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{3^i} \rfloor P, \lfloor \frac{n}{3^i} \rfloor P}}{h_{3\lfloor \frac{n}{3^i} \rfloor P}} \frac{h_{3\lfloor \frac{n}{3^i} \rfloor P, t_{i-1}P}}{h_{\lfloor \frac{n}{3^{i-1}} \rfloor P}} \right)^{3^{i-1}}, \tag{4.2}$$

where $r = \lfloor \lg_3 n \rfloor$ and $t_{i-1} = \lfloor \frac{n}{3^{i-1}} \rfloor - 3\lfloor \frac{n}{3^i} \rfloor$.

4.1 Refinement 1

Consider the binary represenation of $n = \sum_{i=0}^{t} b_i 2^i$, and group every two terms in Formula 4.1 together. Then we get the following relation by applying Lemma 3.1.

$$\left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P}}{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}}{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P, b_{i-2}P}}{h_{\lfloor \frac{n}{2^{i-2}} \rfloor P}} \right)^{2^{i-2}}$$

$$= \begin{cases} \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, P}}{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}}{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P, P}}{h_{\lfloor \frac{n}{2^{i-2}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = b_{i-2} = 1 \\ \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, P}}{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}}{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = 1, b_{i-2} = 0 \\ \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, P}}{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}}{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = 0, b_{i-2} = 1 \\ \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}}{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = b_{i-2} = 0 \end{cases}$$

$$= \begin{cases} \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P} h_{2\lfloor \frac{n}{2^i} \rfloor P, P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} \left(\frac{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P, P}}{\overline{\overline{h}}_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P} h_{\lfloor \frac{n}{2^{i-2}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = b_{i-2} = 1 \\ \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P} h_{2\lfloor \frac{n}{2^i} \rfloor P, P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} \left(\frac{1}{\overline{\overline{h}}_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = 1, b_{i-2} = 0 \\ \left(h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P} \right)^{2^{i-1}} \left(\frac{h_{2\lfloor \frac{n}{2^{i-1}} \rfloor P, P}}{\overline{\overline{h}}_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P} h_{\lfloor \frac{n}{2^{i-2}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = 0, b_{i-2} = 1 \\ \left(h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P} \right)^{2^{i-1}} \left(\frac{1}{\overline{\overline{h}}_{\lfloor \frac{n}{2^{i-1}} \rfloor P, \lfloor \frac{n}{2^{i-1}} \rfloor P}} \right)^{2^{i-2}} & \text{if } b_{i-1} = b_{i-2} = 0 \end{cases}$$

This relation provides the correctness of an improved Miller's algorithm which is generally efficient and achieves greater efficiency as the number of zero b_i's increases. As the simplification is achieved by grouping two terms together, it is natural to expand n in terms of base 4 which is given in the next algorithm.

Algorithm 4.1 *(Improved Miller's algorithm (version 1))*
`INPUT:` *Integer* $n = \sum_{i=0}^{r} q_i 4^i$ *with* $q_i \in \{0, 1, 2, 3\}$ *and* $q_r \neq 0$, *and a point* $S \in E$
`OUTPUT:` $f = f_n(S)$.

```
  $f \leftarrow f_1$; $Z \leftarrow P$;
  If  $q_r = 2$ then
    $f \leftarrow f^2 \frac{h_{P,P}(S)}{h_{2P}(S)}$; $Z \leftarrow 2P$;
  Endif
  If  $q_r = 3$ then
    $f \leftarrow f^3 \frac{h^2_{P,P}(S) h_P(S)}{h_{2P,P}(-S)}$; $Z \leftarrow 3P$;
  Endif
  For   $j \leftarrow r-1, r-2, \ldots, 1, 0$ do
    If  $q_j = 0$ then
      $f \leftarrow \frac{f^4 h^2_{Z,Z}(S)}{h_{2Z,2Z}(-S)}$; $Z \leftarrow 4Z$;
    Endif
    If  $q_j = 1$ then
      $f \leftarrow f_1 \frac{f^4 h^2_{Z,Z}(S) h_{4Z,P}(S)}{h_{4Z+P}(S) h_{2Z,2Z}(-S)}$; $Z \leftarrow 4Z + P$;
    Endif
    If  $q_j = 2$ then
      $f \leftarrow {f_1}^2 \frac{f^4 h^2_{Z,Z}(S) h^2_{2Z,P}(S)}{h^2_{2Z}(S) h_{2Z+P,2Z+P}(-S)}$; $Z \leftarrow 4Z + 2P$;
    Endif
    If  $q_j = 3$ then
      $f \leftarrow {f_1}^3 \frac{f^4 h^2_{Z,Z}(S) h^2_{2Z,P}(S) h_{4Z+2P,P}(S)}{h^2_{2Z}(S) h_{2Z+P,2Z+P}(-S) h_{4Z+3P}(S)}$; $Z \leftarrow 4Z + 3P$;
    Endif
  Endfor

Return $f$
```

4.2 Refinement 2

Suppose that most bits in the binary represenation of $n = \sum_{i=0}^{t} b_i 2^i$ are 1, then Miller's algorithm can be modified to save more field operations based on the following observation:

By rearranging Formula 4.1 and then applying Lemma 3.2, the following computations are obtained:

$$\begin{aligned} f_n &= f_1^n \left(\frac{h_{P,P} h_{2P,b_{t-1}P}}{h_{2P}} \right)^{2^{t-1}} \Pi_{i=t-1}^{1} \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}}{(h_{\lfloor \frac{n}{2^i} \rfloor P})^2} \frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P}}{h_{2\lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} \\ &= f_1^n \left(\frac{h_{P,P} h_{2P,b_{t-1}P}}{h_{2P}} \right)^{2^{t-1}} \Pi_{i=t-1}^{1} \left(\frac{h_{2\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P}}{\bar{h}_{\lfloor \frac{n}{2^i} \rfloor P, \lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} \end{aligned}$$

The fact that $h_{nP} = 1$ has been used and the last term inside the product symbol is $\frac{1}{h_{\lfloor \frac{n}{2} \rfloor P, \lfloor \frac{n}{2} \rfloor P}}$.

Algorithm 4.2 *(Improved Miller's algorithm (version 2))*
INPUT: *Integer* $n = \sum_{i=0}^{t} b_i 2^i$ *with* $b_i \in \{0,1\}$ *and* $b_t = 1$*, and a point* $S \in E$
OUTPUT: $f = f_n(S)$.

```
  If  $b_{t-1} = 0$ then
    $f \leftarrow {f_1}^2 h_{P,P}(S)$; $Z \leftarrow 2P$;
  Else
    $f \leftarrow {f_1}^3 \frac{h_{P,P}(S) h_{2P,P}(S)}{h_{2P}(S)}$; $Z \leftarrow 3P$;
  Endif
  For  $j \leftarrow t-2, \ldots, 1, 0$ do
    If  $b_j = 0$ then
      $f \leftarrow f^2 \frac{h_{2Z}(S)}{h_{Z,Z}(-S)}$; $Z \leftarrow 2Z$;
    Else
      $f \leftarrow f_1 f^2 \frac{h_{2Z,P}(S)}{h_{Z,Z}(-S)}$; $Z \leftarrow 2Z + P$;
    Endif
  Endfor


Return $f$
```

In [8], Eisenträger, Lauter and Montgomery suggested a method that eliminates a field multiplication and a field squaring in the computation of $2Q+P$. They obtained an improvement of Miller's algorithm by using this observation and a parabola substitution. Their algorithm speeds up the computations in the case that most bits in the binary represenation of n are 1. However, although their method can not be combined with the above algorithm, we may modify and simplify the function f_n as follows so that the method of Eisenträger, Lauter and Montgomery can be used.

$$\begin{aligned} f_n &= f_1^n \Pi_{i=t}^{1} \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P} h_{(\lfloor \frac{n}{2^i} \rfloor + b_{i-1})P, \lfloor \frac{n}{2^i} \rfloor P}}{h_{(\lfloor \frac{n}{2^i} \rfloor + b_{i-1})P} h_{(2\lfloor \frac{n}{2^i} \rfloor + b_{i-1})P}} \right)^{2^{i-1}} \\ &= f_1^n \Pi_{i=t}^{1} \left(\frac{h_{\lfloor \frac{n}{2^i} \rfloor P, b_{i-1}P} h_{\lfloor \frac{n}{2^i} \rfloor P}}{\overline{h}_{(\lfloor \frac{n}{2^i} \rfloor + b_{i-1})P, \lfloor \frac{n}{2^i} \rfloor P}} \right)^{2^{i-1}} . \end{aligned}$$

An algorithm can be formed as before based on the above formula.

4.3 Refinement 3

As stated in [1, 6], point tripling is a relatively cheap operation in the case of characteristic three and the base three version of Miller's algorithm 2.2 should be used in this case.

Let $n = \sum_{i=0}^{r} t_i 3^i$ with $t_r \neq 0$. Applying part 2 of Lemma 3.1 to Formula 4.2, we see that

$$\begin{aligned} f_n &= f_1^n \left(\frac{h_{(3-\lfloor \frac{n}{3^r}\rfloor)P,(\lfloor \frac{n}{3^r}\rfloor-1)P}}{h_{2P}}\right)^{3^r} \Pi_{i=r}^{1} \left(\frac{h_{\lfloor \frac{n}{3^i}\rfloor P,\lfloor \frac{n}{3^i}\rfloor P}}{h_{2\lfloor \frac{n}{3^i}\rfloor P}} \frac{h_{2\lfloor \frac{n}{3^i}\rfloor P,\lfloor \frac{n}{3^i}\rfloor P}}{h_{3\lfloor \frac{n}{3^i}\rfloor P}} \frac{h_{3\lfloor \frac{n}{3^i}\rfloor P,t_{i-1}P}}{h_{\lfloor \frac{n}{3^{i-1}}\rfloor P}}\right)^{3^{i-1}} \\ &= f_1^n \left(\frac{h_{(3-\lfloor \frac{n}{3^r}\rfloor)P,(\lfloor \frac{n}{3^r}\rfloor-1)P}}{h_{2P}}\right)^{3^r} \Pi_{i=r}^{1} \left(\frac{h_{\lfloor \frac{n}{3^i}\rfloor P,\lfloor \frac{n}{3^i}\rfloor P} h_{\lfloor \frac{n}{3^i}\rfloor P}}{\overline{h}_{2\lfloor \frac{n}{3^i}\rfloor P,\lfloor \frac{n}{3^i}\rfloor P}} \frac{h_{3\lfloor \frac{n}{3^i}\rfloor P,t_{i-1}P}}{h_{\lfloor \frac{n}{3^{i-1}}\rfloor P}}\right)^{3^{i-1}} . \end{aligned}$$

This formula is realised by the algorithm 4.3 which improves algorithm 2.2.

Algorithm 4.3 *(Improved Miller's algorithm (version 3))*
`INPUT:` *Integer* $n = \sum_{i=0}^{r} t_i 3^i$ *with* $t_i \in \{0, 1, 2\}$ *and* $t_r \neq 0$*, and a point* $S \in E$
`OUTPUT:` $f = f_n(S)$.

```
  $f_2 \leftarrow {f_1}^2 \frac{h_{P,P}(S)}{h_{2P}S)}$;
  $f \leftarrow f_1$; $Z \leftarrow P$;
  If  $t_r = 2$ then
    $f \leftarrow f_2$; $Z \leftarrow 2P$;
  Endif
  For  $j \leftarrow r-1, r-2, \ldots, 1, 0$ do
    $f = f^3 \frac{h_{Z,Z}(S) h_Z(S)}{h_{2Z,Z}(-S)}$; $Z \leftarrow 3Z$;
    If  $t_j = 1$ then
      $f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$; $Z \leftarrow Z + P$;
    Endif
    If  $t_j = 2$ then
      $f \leftarrow f_2 f \frac{h_{Z,2P}(S)}{h_{Z+2P}(S)}$; $Z \leftarrow Z + 2P$;
    Endif
  Endfor


Return $f$
```

5 Analysis

In this section, some detailed analysis of the refinements are given and the number of operations that can be saved discussed.

As indicated in [1, 6, 8], in the actual implementaion of the algorithms, the operations in the numerator and denominator in each step are separated and the single division is used at the end of the procedure.

Observe that the savings come from the elimination of terms like $h_{X,Y}(S)$ and $h_X(S)$. It is easy to see that $h_{X,Y}(S)$ and $h_{X,Y}(-S)$ both cost one field multipliction if the slope has been precalculated. Also note that both algorithm 4.1 and algorithm 4.2 use the same method for doing point operations (doubling and addition) as in Miller's original algorithm 2.1. So we only count the field operations used to evaluate $h_{X,Y}, h_X$ and to multiply (or square) terms like $h_{X,Y}(\pm S), h_X(S)$.

First, the savings made by using our first improvement, (algorithm 4.1) are estimated. Consider a single round of the *for* loop of algorithm 4.1. Two field multiplications will be saved for each case. For example, $q_j = 1$, then the deduction is

$$f \leftarrow f_1 f^4 \frac{h^2_{Z,Z}(S) h_{4Z,P}(S)}{h_{2Z,2Z}(-S) h_{4Z+P}(S)}.$$

If we assume that f_1 and f have already been written as quotients on the right hand side, then it takes 2 squarings and 6 multiplications for the numerator, and 2 squarings and 4 multiplications for the denominator. This should be compared with 2 rounds of the *for* loop of the original Miller's algorithm 2.1 with the following result:

$$f \leftarrow f_1 f^4 \frac{h^2_{Z,Z}(S) h_{2Z,2Z}(S) h_{4Z,P}(S)}{h^2_{2Z}(S) h_{4Z}(S) h_{4Z+P}(S)}.$$

It requires 2 squarings and 8 multiplications for the numerator, and 2 squarings and 4 multiplications for the denominator. So, the number of field multiplication saved in total is $2 \log_4 n = \log_2 n$.

Next, the savings made by using algorithm 4.2 are considered. For each single round in the *for* loop, if $b_j = 1$, our computation of

$$f \leftarrow f_1 f^2 \frac{h_{2Z,P}(S)}{h_{Z,Z}(-S)}$$

needs 1 squaring and 3 multiplications for the numerator, and 1 squaring and 3 multiplications for the denominator. This is two field multiplications fewer then the computation of

$$f \leftarrow f_1 f^2 \frac{h_{Z,Z}(S) h_{2Z,P}(S)}{h_{2Z}(S) h_{2Z+P}(S)}$$

which is from the original algorithm. Thus the overall savings is $2H(n)$, where again, $H(n)$ is the weight of the binary expansion of n.

As indicated in section 4.2, we can rewrite the case of $b_j = 1$ of the *for* loop in the original Miller's algorithm as

$$f \leftarrow f_1 f^2 \frac{h_{Z,P}(S) h_{Z+P,Z}(S)}{h_{Z+P}(S) h_{2Z+P}(S)}$$

and simplify it to

$$f \leftarrow f_1 f^2 \frac{h_{Z,P}(S) h_Z(S)}{h_{Z+P,Z}(-S)}.$$

One field multiplication is saved from the numerator and hence a total of $H(n)$ field multiplications are saved. But since there is no need to reference (the y-coordinate of) $2Z$, the trick of Eisenträger, Lauter and Montgomery can be used, so another $H(n)$ field multiplications and $H(n)$ field squares can be saved.

Similar to the previous discussion, it can be checked that the algorithm 4.3 saves $\log_3 n$ field multiplications compared with its base three counterpart. Note that in this case, $\log_3 n$ of point triplings are performed. Since tripling can be made very efficient, therefore algorithm 4.3 is a good choice here.

Table 1 summarizes the performance of the new algorithms, where the numbers in the saving column indicate the number of field multiplications eliminated in the respective algorithms.

Table 1: *Performance of the improved algorithms*

Algorithm	Saving	Condition for improvement
Algorithm 4.1	$\log_2 n$	All values of n
Algorithm 4.2	$2H(n)$	Higher Hamming weight
Algorithm 4.3	$\log_3 n$	Characteristic three

Finally, two examples are given. We list the calculation formulas for f_{191}, f_{257} using Miller's algorithm from [1, 3] (Algorithm2.1), our improved version 1(Algorithm4.1) and our improved version 2(Algorithm4.2). Notice that the prime numbers 191 and 257 represent two extreme situations since the first has only the one zero in its binary expansion and the second is of weight two, the minimal weight possible for a nontrivial prime number. Here the symbols $h_{kP,mP}, h_{kP}$ are shortened as $h_{k,m}, h_k$ respectively. Also $\overline{h}_{k,m}$ is used to denote $h_{kP,mP}(-S)$.

Example 5.1 *Compute f_{191}:*

Number	$191 = (10111111)_2 = (2333)_4$
Algorithm 2.1	$f = {f_1}^{191} \frac{h_{1,1}^{64}}{h_2^{64}} \frac{h_{2,2}^{32}}{h_4^{32}} \frac{h_{4,1}^{32}}{h_5^{32}} \frac{h_{5,5}^{16}}{h_{10}^{16}} \frac{h_{10,1}^{16}}{h_{11}^{16}} \frac{h_{11,11}^{8}}{h_{22}^{8}} \frac{h_{22,1}^{8}}{h_{23}^{8}} \frac{h_{23,23}^{4}}{h_{46}^{4}} \frac{h_{46,1}^{4}}{h_{47}^{4}} \cdot \frac{h_{47,47}^{2}}{h_{94}^{2}} \frac{h_{94,1}^{2}}{h_{95}^{2}} \frac{h_{95,95}}{h_{190}} \frac{h_{190,1}}{h_{191}}$
Algorithm 4.1	$f = {f_1}^{191} \frac{h_{1,1}^{64}}{h_2^{64}} \frac{h_{2,2}^{32}}{h_4^{32}} \frac{h_{4,1}^{32}}{\overline{h}_{5,5}^{16}} \frac{h_{10,1}^{16}}{h_{11}^{16}} \frac{h_{11,11}^{8}}{h_{22}^{8}} \frac{h_{22,1}^{8}}{\overline{h}_{23,23}^{4}} \frac{h_{46,1}^{4}}{h_{47}^{4}} \frac{h_{47,47}^{2}}{h_{94}^{2}} \frac{h_{94,1}^{2}}{\overline{h}_{95,95}} \frac{h_{190,1}}{h_{191}}$
Algorithm 4.2	$f = {f_1}^{191} h_{1,1}^{64} \frac{h_{4,1}^{32}}{\overline{h}_{2,2}^{32}} \frac{h_{5,5}^{16}}{h_{10}^{16}} \frac{h_{10,1}^{16}}{\overline{h}_{5,5}^{16}} \frac{h_{22,1}^{8}}{\overline{h}_{11,11}^{8}} \frac{h_{46,1}^{4}}{\overline{h}_{23,23}^{4}} \frac{h_{94,1}^{2}}{\overline{h}_{47,47}^{2}} \frac{h_{190,1}}{\overline{h}_{95,95}}$

Compute f_{257}:

Number	$257 = (100000001)_2 = (10001)_4$
Algorithm 2.1	$f = {f_1}^{257} \frac{h_{1,1}^{128}}{h_2^{128}} \frac{h_{2,2}^{64}}{h_4^{64}} \frac{h_{4,4}^{32}}{h_8^{32}} \frac{h_{8,8}^{16}}{h_{16}^{16}} \frac{h_{16,16}^{8}}{h_{32}^{8}} \frac{h_{32,32}^{4}}{h_{64}^{4}} \frac{h_{64,64}^{2}}{h_{128}^{2}} \frac{h_{128,128}}{h_{256}} \frac{h_{256,1}}{h_{257}}$
Algorithm 4.1	$f = {f_1}^{257} \frac{h_{1,1}^{128}}{\overline{h}_{2,2}^{64}} \frac{h_{4,4}^{32}}{\overline{h}_{8,8}^{16}} \frac{h_{16,16}^{8}}{\overline{h}_{32,32}^{4}} \frac{h_{64,64}^{2}}{\overline{h}_{128,128}} \frac{h_{256,1}}{h_{257}}$
Algorithm 4.2	$f = {f_1}^{257} h_{1,1}^{128} \frac{h_4^{64}}{\overline{h}_{2,2}^{64}} \frac{h_8^{32}}{\overline{h}_{4,4}^{32}} \frac{h_{16}^{16}}{\overline{h}_{8,8}^{16}} \frac{h_{32}^{8}}{\overline{h}_{16,16}^{8}} \frac{h_{64}^{4}}{\overline{h}_{32,32}^{4}} \frac{h_{128}^{2}}{\overline{h}_{64,64}^{2}} \frac{h_{256,1}}{\overline{h}_{128,128}}$

6 Comments

Three refinements for the computation of the Tate/Weil pairing have been given and the corresponding performance analyzed. The savings in the number of multiplications noted could prove important for the performance of algorithms in the implementations of many of the new and interesting protocols that have been, and will continue to be, developed using these pairings.

References

[1] P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott,*Efficient algorithms for pairing-based cryptosystems*, Advances in Cryptology-CRYPTO '02,(Santa Barbara, CA, 02) (M. Yung Ed.), Lecture Notes in Comput. Sci., vol. 2442, Springer-Verlag Heidelberg, 2002, pp. 354–368.

[2] I. F. Blake, G. Seroussi and N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, (1999).

[3] D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology, Crypt'01 (J. Kilian ED.), Lecture Notes in Comput. Sci., vol. 2139, Springer-Verlag Heidelberg, 2001, pp. 213–239.

[4] D. Boneh, B. Lynn and H. Shacham, *Short signatures from the Weil pairing*, Advances in Cryptology, Asiacrypt'01 (C. Boyd ED.), Lecture Notes in Comput. Sci., vol. 2248, Springer-Verlag Heidelberg, 2001, pp. 514–532.

[5] G. Frey and H. G. Rück, *A remark concerning m-divisibilty and the discrete logarithm in divisor class group of curves*, Mathematics of Computation, **62** (1994), 865–874.

[6] S. Galbraith, K. Harrison and D. Soldera, *Implementing the Tate Pairing*, Algorithm Number Theory Symposium, ANTS-V (C. Fieker and D. Kohel EDS.), Lecture Notes in Comput. Sci., vol. 2369, Springer-Verlag Heidelberg, 2002, pp. 324–337.

[7] A. Joux, *A one round protocol for tripartite Diffie-Helman*, Algorithm Number Theory Symposium, ANTS-IV (W. Bosma ED.), Lecture Notes in Comput. Sci., vol. 1838, Springer-Verlag Heidelberg, 2000, pp. 385–393.

[8] K. Eisenträger, K. Lauter and P. L. Montgomery, *Fast Elliptic curve arithmetic and improved Weil pairing Evaluation*, Topics in Cryptology, CT-RSA'03, Lecture Notes in Comput. Sci., vol. 2612, Springer-Verlag Heidelberg, 2003, pp. 343–354.

[9] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, **48** (1987), 203-209.

[10] A. J. Menezes, T. Okamoto and S. A. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field,* IEEE Transactions on Information Theory, **39** (1993), 1639-1646.

[11] V. Miller, *Short Programs for functions on curves*, unpublished manuscript, 1986.

[12] V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology, CRYPTO'85, Lecture Notes in Comput. Sci., vol. 218, Springer-Verlag Heidelberg, 1986, pp. 417–462.

[13] R. Sakai, K. Ohgishi and M. Kasahara, *Cryptosystems based on pairing*, SCIS-2000, OKinawa, Japan, 2000.

[14] A. Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology–Crypto'84, Lecture Notes in Comput. Sci., vol. 196, Springer-Verlag Heidelberg, 1984, pp. 47-53.

[15] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, 106, Springer-Verlag, 1986.

[16] E. R. Verheul, *Self-blindable credential certificates from the Weil pairing*, Advances in Cryptology, Asiacrypt'01 (C. Boyd ED.), Lecture Notes in Comput. Sci., vol. 2248, Springer-Verlag Heidelberg, 2001, pp. 533–551.