Efficient Pairing Computation on Supersingular Abelian Varieties

Paulo S. L. M. Barreto[1], Steven Galbraith[2],
Colm O hEigeartaigh[3], and Michael Scott[3]

[1] Universidade de São Paulo, Escola Politécnica.
Av. Prof. Luciano Gualberto, tr. 3, n. 158, s. C1-46.
BR 05508-900, São Paulo(SP), Brazil.
pbarreto@larc.usp.br
[2] Mathematics Department, Royal Holloway University of London.
Egham, Surrey TW20 0EX, UK.
steven.galbraith@rhul.ac.uk
[3] School of Computing, Dublin City University.
Ballymun, Dublin 9, Ireland.
{coheigeartaigh,mike}@computing.dcu.ie

Abstract. We present a general technique for the efficient computation of pairings on supersingular Abelian varieties. As particular cases, we describe efficient pairing algorithms for elliptic and hyperelliptic curves in characteristic 2. The latter is faster than all previously known pairing algorithms, and as a bonus also gives rise to faster conventional Jacobian arithmetic.

Keywords: Tate pairing, supersingular curves, pairing-based cryptosystem, efficient algorithms.

1 Introduction

Efficient computation of pairings is essential to the large and ever growing area of pairing-based cryptosystems (see e.g. [4] for a comprehensive overview).

There has been a lot of work on efficient implementation of pairings on elliptic curves. Supersingular curves lead to more efficient implementations in terms of processing speed [2, 8, 5] and bandwidth requirements [15, 9] than the best available algorithms for ordinary curves [3]. Pairings on hyperelliptic curves have received considerably less attention than their elliptic counterparts. The best results are by Duursma and Lee [5] for a very special family of supersingular hyperelliptic curves. These results suggest that supersingular hyperelliptic curves may provide similar efficiency to elliptic curves, but these issues have not been at all clear until now.

We tackle this problem by providing criteria under which pairings on supersingular hyperelliptic curves are efficiently computable. Our method is fairly general and includes that of [5] as a particular case. We illustrate the method

by describing efficient pairing algorithms for supersingular genus 1 and genus 2 curves in characteristic 2. As a bonus, we derive explicit formulae for faster conventional arithmetic on the Jacobian of supersingular genus 2 curves in characteristic 2.

This paper is organised as follows. Section 2 gives a brief summary on standard techniques for the efficient computation of the Tate pairing. Section 3 discusses the contributions of Duursma and Lee for certain supersingular curves, and section 4 generalises those contributions using the simpler, unified approach of *eta pairings*. Section 5 explores the consequences of the eta pairing approach for curves in characteristic 2. We compare the eta pairing approach with the work of Rubin and Silverberg in section 6, and present some experimental results in section 7. Finally, we draw our conclusions in section 8.

Parts of this work were presented by one of the authors [1] at the ECC'2004 conference on September 20–22, 2004. Subsequently and independently, on November 14 2004, a paper [10] containing some results related to those in this paper was posted on the ePrint archive.

2 The Tate pairing on supersingular curves

Let C be a smooth projective curve over a finite field $K = \mathbb{F}_{q^k}$. We denote the degree zero divisor class group of C over K by $\mathrm{Pic}_0^K(C)$. Let r be an integer such that $r \mid \#\mathrm{Pic}_0^K(C)$. We denote by $\mathrm{Pic}_0^K(C)[r]$ the divisor classes of order dividing r.

Let D_1 be a divisor representing a class in $\mathrm{Pic}_0^K(C)[r]$ and let D_2 be a divisor on C defined over K such that the supports of D_1 and D_2 are disjoint. Since rD_1 is principal there is a function f on C defined over K such that $(f) = rD_1$. The Tate pairing (also called the Tate-Lichtenbaum pairing) is

$$\langle D_1, D_2 \rangle_r = f(D_2).$$

One can show (see Frey and Rück [7]) that the Tate pairing is a well-defined, non-degenerate, bilinear pairing

$$\mathrm{Pic}_0^K(C)[r] \times \mathrm{Pic}_0^K(C)/r\mathrm{Pic}_0^K(C) \to K^*/(K^*)^r.$$

The fact that the Tate pairing is only defined up to r-th powers is often undesirable. To obtain a unique value, one defines the *reduced* pairing

$$e(D_1, D_2) = \langle D_1, D_2 \rangle_r^{(q^k-1)/r}.$$

Throughout the paper we will refer to the extra powering required to compute the reduced pairing as the *final exponentiation*.

One very important property of the reduced pairing is the following [8]. Let $N = hr$ for some h.

$$e(D_1, D_2) = \langle D_1, D_2 \rangle_r^{(q^k-1)/r} = \langle D_1, D_2 \rangle_N^{(q^k-1)/N}. \tag{1}$$

2.1 Miller's algorithm in the elliptic case

We recall how the Tate pairing can be computed in polynomial time using Miller's algorithm [12]. For simplicity we restrict to the case of elliptic curves. The divisor class group of an elliptic curve is isomorphic to the curve itself, so all divisors may be assumed to have the form $D = (P) - (\infty)$.

Let E be an elliptic curve over $\mathbb{F}_q$ and let $r \mid \#E(\mathbb{F}_q)$ be a prime. Suppose the embedding degree is k (i.e., k is the smallest positive integer such that $r \mid (q^k - 1)$). Let $P \in E[r]$ and $Q \in E(\mathbb{F}_{q^k})$, where typically Q is the image of some multiple of P under a non-rational endomorphism called a distortion map. We construct an $\mathbb{F}_{q^k}$-rational divisor D equivalent to $(Q) - (\infty)$ by taking a random point $R \in E(\mathbb{F}_{q^k})$ and defining $D = (Q + R) - (R)$. We aim to compute

$$e(P, Q) = e((P) - (\infty), D).$$

For every integer n and point P there is a function $f_{n,P}$ such that

$$(f_{n,P}) = n(P) - ([n]P) - (n - 1)(\infty).$$

Miller's algorithm builds up these functions $f_{n,P}$ according to the following formula: If l and v are the lines which arise in the addition rule for adding $[n]P$ and $[m]P$ then we have

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

The pairing value $\langle (P) - (\infty), D \rangle_r$ is $f_{r,P}(D)$.

Thus, Miller's algorithm is the following:

Algorithm 1 Miller's algorithm

INPUT: r, P, $Q + R$, R, where the binary representation of r is $\{r_i\}$.
OUTPUT: $\langle P, Q \rangle_r$

$T \leftarrow P$
$f \leftarrow 1$
for $i \leftarrow \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
    ▷ Calculate lines l and v in doubling T
    $T \leftarrow [2]T$
    $f \leftarrow f^2 \cdot l(Q + R)v(R)/(v(Q + R)l(R))$
    if $r_i = 1$ **then**
        ▷ Calculate lines l and v in adding P to T
        $T \leftarrow T + P$
        $f \leftarrow f \cdot l(Q + R)v(R)/((v(Q + R)l(R))$
    end if
end for
return f

Note that the addition in the final iteration is simplified in that l is a vertical line and v disappears.

Miller's algorithm can be generalised to general divisor class groups. The basic algorithm is the same, but the functions are more complicated

2.2 Improvements to Miller's algorithm

Several improved implementation techniques to compute the reduced Tate pairing on supersingular elliptic curves have been proposed [2, 8]. These include:

Exploiting properties of the field of definition: It is typical in pairing applications to pair a point defined over $\mathbb{F}_q$ with a point defined over $\mathbb{F}_{q^k}$. Hence it makes sense to represent $\mathbb{F}_{q^k}$ as an extension of $\mathbb{F}_q$ and to try to simplify the operations in $\mathbb{F}_{q^k}$ as much as possible.

The final exponentiation eliminates terms defined over subfields. Hence, terms defined over subfields can be omitted from the calculations. For example, if $k > 1$ then the point R can be chosen to be defined over a subfield, in which case all terms $l(R)$ and $v(R)$ may be ignored.

Changing the base in Miller's algorithm: Miller's algorithm is usually presented as a loop through the binary expansion of the group order. It is sometimes more efficient to use other bases, for example to write the group order in base three when implementing pairings in characteristic three.

Replacing divisors by points: As explained above, the point R can be ignored. In fact, one can choose $R = \infty$ by [2, Theorem 1]. Hence, the reduced pairing can be computed as

$$e(P, Q) = f_r(Q)^{(q^k-1)/r},$$

where the function f_r is now evaluated on a *point* rather than on a divisor.

Exploiting the form of the distortion maps and denominator elimination: If the distortion map is chosen so that the x-coordinates always lies in a subfield, then all terms $v(Q)$ may be eliminated. As a result there are no longer any divisions in Miller's algorithm.

Hamming weight/group order issues: Miller's algorithm to compute f_r involves a number of arithmetic operations proportional to the Hamming weight of r, and for this reason it is advantageous to choose r with low Hamming weight (with respect to the base being used) whenever possible. In many cases it is worth using a small multiple of r which has low Hamming weight and exploiting formula (1).

Speeding up the final exponentiation: The naive way to compute the final powering to $(q^k-1)/N$ (for some multiple N of r) has cubic complexity. However, this exponent has a rather simple structure for supersingular curves when one chooses N to be the full curve order rather than a factor r thereof. By carefully exploiting that structure, one can replace the powering by a few applications of the Frobenius, a couple of multiplications, and one inversion. Details can be found in [2, Appendix A.2].

Pairing value compression: It is possible to save the bandwidth requirements of pairing values by storing and manipulating traces [15], or by working on a torus [9]. These methods compress pairing values to half their usual size (or to a third thereof, in the specific case of pairing on supersingular elliptic curves in characteristic 3).

The techniques mentioned above give impressive results for pairing implementation. For the remainder of the paper we focus on further improvements. We consider only supersingular curves over $\mathbb{F}_q$ with embedding degree $k > 1$ and with suitable distortion maps ψ. We will always be computing a modified pairing

$$\hat{e}_r(P, Q) = \langle P, \psi(Q) \rangle_r$$

where P and Q are defined over $\mathbb{F}_q$.

3 The Duursma-Lee techniques

Duursma and Lee [5] had several ideas which give a significant improvement to the computation of pairings on curves of the form $y^2 = x^p - x + d$ over $\mathbb{F}_{p^m}$ (these curves have embedding degree $2p$). In particular, they give new results in the case of the embedding degree 6 curve in characteristic three.

The presentation in [5] is *ad hoc* and quite involved, but a careful examination shows that it contains four independent contributions:

1. A nice choice of function for computing pD in the divisor class group;
2. The definition of a pairing on *points* (in $g > 1$) rather than divisors;
3. A shorter loop than would be expected for the given group order;
4. Incorporating the large powering and other Frobenius operations directly into the formulae.

One of the aims of this paper is to clarify the presentation in [5] so that it can easily be generalised to other cases. Points 1, 2 and 4 are relatively straightforward to adapt, but the third contribution is less obvious. We give a general explanation of this idea in the next section.

4 The eta pairing approach

One crucial aspect of [5] is that they work with a group order of the form $p^{mk}+1$ where the embedding degree is $2k$ (so $k = p$ for the family of curves considered in [5]) and where $\gcd(m, 2k) = 1$. This group order has Hamming weight 2 in base p, which is minimal. The final powering by $p^{mk} - 1$ is simply computing a Frobenius conjugation and a division.

However, instead of requiring a loop with mk iterations, they have a loop with only m iterations. In this section we explain how this can be achieved in a very general way.

4.1 Generic view of the method

Let P be a point on a hyperelliptic curve C of genus g over $\mathbb{F}_{q^k}$ (where $q = p^m$) and let ∞ be a (fixed) point on C. Assume we have a distortion map ψ which gives rise to denominator elimination in the usual way. Let D be the divisor $D = (P) - (\infty)$ and suppose that D has order dividing $q^k + 1$. We will develop the functions required for a Miller algorithm in base p.

Define effective divisors E_i of degree $d_i \leq g$ such that $p^i D \equiv E_i - d_i(\infty)$ in the divisor class group. Define f_{p^j,p^iP} to be functions such that

$$(f_{p^j,p^iP}) = p^j E_i - E_{i+j} - (p^j d_i - d_{i+j})(\infty).$$

Then one can choose the f_{p^j,p^iP} such that

$$f_{p^{i+1},P} = f_{p^i,P}^p f_{p,p^iP}. \tag{2}$$

Miller's algorithm requires computing $f_{q^k+1,P}$, but as we will now show, since C is hyperelliptic it suffices to compute $f_{q^k,P}$, which can be done using equation (2). This follows because $q^k((P) - (\infty)) \equiv ((\overline{P}) - (\infty))$ where $\overline{P}$ is the image of P under the hyperelliptic involution. Therefore, the final step in Miller's algorithm involves a function which depends only on x. Hence, if the point $Q \in E(\mathbb{F}_{q^{2k}})$ has x-coordinate in $\mathbb{F}_{q^k}$ (i.e., if the distortion map is of a suitable form) then this operation may be omitted. In other words, there are no longer any addition steps in Miller's algorithm in base p.

By expanding out the definitions we obtain

$$\langle P, \psi(Q)\rangle_{q^k+1} = f_{q^k,P}(\psi(Q)) = \prod_{i=0}^{km-1} f_{p,p^iP}(\psi(Q))^{p^{km-1-i}}. \tag{3}$$

where we should complete the calculation by exponentiating to the power (q^k-1).

What Duursma and Lee noticed is that this loop can be shortened from a product of km terms to a product of k terms. We now give a general explanation of this fact.

Definition 1. *Let notation be as above. We define the η_T pairing as*

$$\eta_T(P, Q) = \prod_{i=0}^{m-1} f_{p,p^iP}(\psi(Q))^{p^{m-1-i}}.$$

It is clear that equation (3) can be written as a product of $\eta_T(P, Q)$ terms as follows.

$$\langle P, \psi(Q)\rangle_{q^k+1} = \eta_T(P, Q)^{q^{k-1}} \eta_T(qP, Q)^{q^{k-2}} \cdots \eta_T(q^{k-1}P, Q). \tag{4}$$

We will prove that in certain special cases $\eta_T(P, Q)$ is bilinear and hence that

$$\langle P, \psi(Q)\rangle_{q^k+1} = \eta_T(P, Q)^{pq^{k-1}}.$$

An important observation is that, for many supersingular curves, multiplication by p has an extremely special form. This has already been exploited by many authors. Essential to the case of curves of genus > 1 is the fact that multiplication by p (or a power of p) of a special divisor of the form $(P) - (\infty)$ may be another divisor of this form. Hence, from now on we assume that $q((P) - (\infty)) = (\gamma(P)) - (\infty)$ for some automorphism γ on the curve.

The key to our results is the following condition on the distortion map ψ which relates multiplication by q with the q-power Frobenius. The condition is

$$\gamma\psi^q(Q) = \psi(Q). \tag{5}$$

We can now give our main result.

Theorem 1. *If ψ satisfies condition (5) then*

$$\eta_T(qP, Q) = \eta_T(P, Q)^q.$$

Note that this is a rather subtle case of bilinearity as we consider the q on the left hand side as point exponentiation while the q on the right hand side is a Galois action on field elements.

Proof. Define the function

$$g_P = \prod_{i=0}^{m-1} f_{p,p^iP}^{p^{km-1-i}}$$

so that $g_P = f_{q,P}$ and $\eta_T(P, Q) = g_P(\psi(Q))$. The first step is to compare g_P and g_{qP}.

It is known (see Silverman [17] Chapter II page 33) that

$$\gamma^*\left(\sum n_P(P)\right) = \sum_{S \in \gamma^{-1}(P)} n_P e_\gamma(S)(S).$$

and so

$$\begin{aligned}\gamma^*(f_{p,p^{m+i},P}) &= \gamma^*(p(p^ip^mP) - (p^{i+1}p^mP) - (p-1)(\infty)\\ &= p(p^iP) - (p^{i+1}P) - (p-1)(\infty)\\ &= (f_{p,p^iP}).\end{aligned}$$

Also, (Silverman [17] pages 33-34)

$$\gamma^*(f_{p,p^{m+i}P}) = (\gamma^* f_{p,p^{m+i}P}) = (f_{p,p^{m+i}P} \circ \gamma).$$

Hence, we can take

$$f_{p,p^{m+i}P} = f_{p,p^iP}\gamma^{-1}$$

which implies that

$$g_{qP} = g_P\gamma^{-1}.$$

We want to deduce that

$$\eta_T(P,Q)^q = \eta_T(qP,Q).$$

The left hand side is

$$\eta_T(P,Q)^q = g_P(\psi(Q))^q = g_P(\psi^q(Q))$$

since both P and Q are defined over $\mathbb{F}_q$ and so g_P and Q are preserved by the Frobenius action. The right hand side is

$$\eta_T(qP,Q) = g_{qP}(\psi(Q)) = g_P\gamma^{-1}\psi(Q).$$

The result follows from condition (5). □

Note: The proof has been presented in a way which emphasises the key role of condition (5). It seems unlikely that the result would be true if condition (5) is not satisfied, but we can't prove this.

We now show that these results explain the loop shortening achieved by Duursma and Lee. They have the curve $E : y^2 = x^3 - x + b$ over $\mathbb{F}_{3^m}$ where $\gcd(m,6) = 1$. The tripling formula is $[3](x,y) = \phi\pi^2(x,y)$ where π is the 3-power Frobenius and $\phi(x,y) = (x-b,-y)$. Note that $\phi^2(x,y) = (x-2b,y), \phi^3 = -1$ etc. The distortion map is $\psi(x,y) = (\rho - x, iy)$ where $i^2 = -1$ and $\rho^3 = \rho + b$ (and thus $\rho^{3^2} = \rho + 2b$, $\rho^{3^3} = \rho$). We will show that $\gamma\psi^q = \psi$.

Let $q = 3^m$ with $m \equiv 1 \pmod 6$. Suppose $(x,y) \in E(\mathbb{F}_q)$. Then $\gamma(x,y) = [q](x,y) = \phi(x,y)$ and $\psi^q = \psi^3$. Then

$$\gamma\psi^q(x,y) = \phi(\rho + b - x, -iy) = (\rho + b - x - b, iy) = (\rho - x, iy) = \psi(x,y).$$

Similarly, when $m \equiv 5 \pmod 6$ we have $\gamma = -\phi^2$ and $\psi^q = -\psi^{3^2}$ and so

$$\gamma\psi^q(x,y) = \phi^2(\rho + 2b - x, iy) = (\rho + 2b - x - 2b, iy) = \psi(x,y).$$

Hence condition (5) is satisfied and so the η_T pairing is bilinear. The method of Duursma and Lee computes the *eta* pairing in a way which combines the other three techniques mentioned at the start of this section.

5 Supersingular pairings in characteristic 2

5.1 The elliptic case

We now consider the case of the supersingular curve $E : y^2 + y = x^3 + x + b$ over $\mathbb{F}_{2^m}$ where m is odd. We will use the ideas presented in section 4. The order of E is given in table 1.

We write $x^{(i)}$ for x^{2^i}. Depending on the field of definition we will usually have either $x^{(m)} = x$ or $x^{(4m)} = x$. Hence we can consider the values (i) as being modulo m or $4m$. This allows us to extend to negative values such as $x^{(-i)}$, which can be interpreted as the 2^i-th root of x.

Table 1. Order of the curve $E : y^2 + y = x^3 + x + b$ over $\mathbb{F}_{2^m}$, $b \in \mathbb{F}_2$.

$\#E(\mathbb{F}_{2^m})$	condition
$2^m + 1 + (-1)^b 2^{(m+1)/2}$	$m \equiv 1, 7 \pmod 8$
$2^m + 1 - (-1)^b 2^{(m+1)/2}$	$m \equiv 3, 5 \pmod 8$

The field $\mathbb{F}_{2^{4m}}$ has elements s, t such that $s^2 = s + 1$ and $t^2 = t + s$; we will represent $\mathbb{F}_{2^{4m}}$ in basis $\{1, s, t, st\}$. Element s satisfies $s^{(1)} = s^2$, $s^{(2)} = s$, and thus $s^{(i)} = s + i$ and $s^{(-i)} = s^{(m-i)} = s + (m - i)$; the latter simplifies to $s^{(-i)} = s + i + 1$ because m is odd. Similarly for t, $t^{(1)} = t + s$, $t^{(2)} = t + 1$, $t^{(3)} = t + s + 1$, $t^{(4)} = t$, and thus $t^{(i)} = t + is + \tau(i)$ and $t^{(-i)} = t^{(m-i)} = t + (m - i)s + \tau(m - i) = t + (i + 1)s + \tau(m) + \tau(i)$, where $\tau(i) = 0$ for $i \equiv 0, 1 \pmod 4$ and $\tau(i) = 1$ for $i \equiv 2, 3 \pmod 4$.

We adopt the following notation:

$$\phi(x, y) \equiv (x + 1, y + x),$$
$$\psi(x, y) \equiv (x + s^2, y + sx + t).$$

One can verify that $\phi^2(x, y) = (x, y + 1)$, $\phi^3(x, y) = (x + 1, y + x + 1)$, and $\phi^4(x, y) = (x, y)$.

Let $P = (x_P, y_P), Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$. We have $x_P^{(m)} = x_P$ (and similarly for the other point coordinates). One can show by induction that

$$2^i P = \phi^i(x_P^{(2i)}, y_P^{(2i)}).$$

We now show that the eta pairing can be applied in this case.

Lemma 1. *With notation as above, condition (5) is satisfied.*

Proof. Write $q = 2^m$ where $m \equiv 1, 3 \pmod 4$. For $P = (x_P, y_P)$ we have

$$\gamma(P) = [q]P = \phi^m(x_P, y_P) = \phi^m(P)$$

We must show that $\gamma\psi^q = \psi$.

Consider first the case $m \equiv 1 \pmod 4$. We have

$$\begin{aligned}\gamma\psi^{2^m}(x, y) &= \phi\psi^2(x, y) = \phi(x + s, y + s^2 x + (t + s)) \\ &= (x + s + 1, y + s^2 x + t + s + x + s) = (x + s^2, y + sx + t) = \psi(x, y).\end{aligned}$$

Similarly, when $m \equiv 3 \pmod 4$ we find that

$$\begin{aligned}\gamma\psi^{2^m}(x, y) &= \phi^3\psi^{2^3}(x, y) = \phi^3(x + s, y + s^2 x + (t + s + 1)) \\ &= (x + s + 1, y + s^2 x + t + s + 1 + x + s + 1) = \psi(x, y).\end{aligned}$$

This completes the proof. □

Given a point V, the function

$$g_V(x,y) \equiv (x_V^2+1)(x_V+x)+y_V+y$$

has divisor $2(V)-(2V)-(\infty)$. We desire to compute the function

$$\langle P,\psi(Q)\rangle_{q^2+1} = \eta_T(P,Q)^{2q} \equiv \prod_{i=0}^{m-1} g_i,$$

where $g_i \equiv (g_{2^iP}(\psi(Q))^{(-i)})^q$. A straightforward but tedious calculation gives

$$g_i = (x_P^{(i+1)}+1)(x_P^{(i)}+x_Q^{(-i)})+x_P^{(i+1)}+y_P^{(i)}+y_Q^{(-i)}+(x_P^{(i+1)}+x_Q^{(-i)}+1)s+t.$$

Algorithm 2 computes the above product.

Algorithm 2 Computation of $\hat{e}(P,Q)$ on $E(\mathbb{F}_{2^m}) : y^2+y=x^3+x+b$

INPUT: P, Q
OUTPUT: $\hat{e}(P,Q)$

1: **let** $P=(x_P,y_P)$, $Q=(x_Q,y_Q)$
2: $f \leftarrow 1$
3: **for** $i \leftarrow 1$ **to** m **do**
4: $\quad u \leftarrow x_P^2$
5: $\quad g \leftarrow (u+1)\cdot(x_P+x_Q)+u+y_P+y_Q+(u+x_Q+1)s+t$
6: $\quad f \leftarrow f\cdot g$
7: $\quad x_P \leftarrow u,\ \ y_P \leftarrow y_P^2,\ x_Q \leftarrow \sqrt{x_Q},\ \ y_Q \leftarrow \sqrt{y_Q}$
8: **end for**
9: **return** f^{q^2-1}

Each step costs 7 $\mathbb{F}_q$ multiplications (1 to compute g, 6 to accumulate it into f by making use of the sparse structure of g), except the first step which costs only 1 $\mathbb{F}_q$ multiplication (because $f=1$). The total loop cost is therefore $7m-6$ multiplications.

A further speed-up may be possible by precomputing all squares x_P and y_P, and then referencing them backwards to get rid of the square roots. However, using the technique described in [6] calculating square roots in the field $\mathbb{F}_{2^m}$ is just as fast as squaring. In fact it may be a little faster as large precomputed tables can lead to memory cache misses which are detrimental to performance.

5.2 The genus 2 case

Here we consider the curve $C_d : y^2+y=x^5+x^3+d$ with $d=0$ or 1 over $\mathbb{F}_{2^m}$, where m is coprime to 6. This curve is supersingular and has embedding degree 12. We give an octupling formula which enables fast point exponentiation and give a corresponding function for Miller's algorithm. The group order is given in Table 2; some examples are listed in Table 3.

Table 2. Order of $\mathrm{Jac}(C_d)$ for the curve $C_d : y^2 + y = x^5 + x^3 + d$ over $\mathbb{F}_{2^m}$, $d \in \mathbb{F}_2$.

$\#\mathrm{Jac}(C_d)(\mathbb{F}_{2^m})$	condition
$2^{2m} + (-1)^d 2^{(3m+1)/2} + 2^m + (-1)^d 2^{(m+1)/2} + 1$	$m \equiv 1,\ 7, 17, 23 \pmod{24}$
$2^{2m} - (-1)^d 2^{(3m+1)/2} + 2^m - (-1)^d 2^{(m+1)/2} + 1$	$m \equiv 5, 11, 13, 19 \pmod{24}$

Table 3. Some examples.

field	curve	cofactor
$\mathbb{F}_{2^{79}}$	$y^2 + y = x^5 + x^3 + 1$	151681
$\mathbb{F}_{2^{103}}$	$y^2 + y = x^5 + x^3$	$13 \cdot 1237$
$\mathbb{F}_{2^{127}}$	$y^2 + y = x^5 + x^3 + 1$	198168459411337
$\mathbb{F}_{2^{199}}$	$y^2 + y = x^5 + x^3 + 1$	$2389 \cdot 121789$
$\mathbb{F}_{2^{239}}$	$y^2 + y = x^5 + x^3 + 1$	1
$\mathbb{F}_{2^{313}}$	$y^2 + y = x^5 + x^3 + 1$	1

Appendix A gives detailed formulae for arithmetic in $\mathrm{Jac}(C)$ over $\mathbb{F}_{2^m}$. These derive from, and improve upon, the explicit formulae given in [11] (1 inversion, 20 multiplications and 4 squarings in the most expensive addition case; 1 inversion, 8 multiplications, and 4 squarings in the most expensive doubling case; and 24 squarings for octupling, without inversions nor multiplications).

5.3 The distortion map

Choose $w \in \mathbb{F}_{2^6}$ to be a root of the polynomial

$$x^6 + x^5 + x^3 + x^2 + 1.$$

Note that $w^8 = w + 1$. Define $s_1 = w^2 + w^4$, $s_2 = w^4 + 1$, and let $s_0 \in \mathbb{F}_{2^{12}}$ be a solution of $s_0^2 + s_0 = w^5 + w^3$.

We will represent elements of the field $\mathbb{F}_{2^{12m}}$ as 12-tuples with respect to the basis

$$\{1, w, w^2, w^3, w^4, w^5, s_0, ws_0, w^2 s_0, w^3 s_0, w^4 s_0, w^5 s_0\}.$$

The distortion map is

$$\psi(x, y) = (x + w, y + s_2 x^2 + s_1 x + s_0).$$

5.4 Octupling

Consider a divisor of form $D = (P) - (\infty)$. In general, jD is not equivalent to a divisor of the form $(Q) - (\infty)$, but as shown in Appendix B in this case we have the octupling formula $8D = (P') - (\infty)$ where $P' = \phi\pi^6(P)$, π is the 2-power Frobenius map, and $\phi(x, y) = (x + 1, y + x^2 + 1)$ so that $\phi^2 = -1$. As a suggestive (but non-standard) notation, we write $[8]P = \phi\pi^6(P)$ and so

$8D = ([8]P) - (\infty)$. Similar results for other supersingular curves were obtained by Duursma and Lee [5].

Since our basic operation is octupling, we are forced to consider the η_T pairing in the case where we have a power of 2^3. Hence we will work with $q = 2^{3m}$ and we will only be reducing the base 8 loop from $2m$ iterations to m iterations. It would be interesting to know if the loop can be shortened further in this case.

We now show that condition (5) is satisfied for our distortion map.

Lemma 2. *Let the notation be as above with $q = 2^{3m}$. Then condition (5) is satisfied.*

Proof. We have $q = 2^{3m}$ where $m \equiv 1, 5, 7$ or $11 \pmod{12}$. Hence we have $3m \equiv 3, 9 \pmod{12}$ and $\gamma = \phi^{3m}$.

We will repeatedly use the formulae $w^8 = w + 1$ and $s_0^8 = s_0 + w^2$ (from which we deduce $s_0^{8^2} = s_0 + 1$ and $s_0^{8^3} = s_0 + w^2 + 1$).

The cases $m \equiv 7, 11 \pmod{12}$ (i.e., $3m \equiv 1 \pmod 4$) yield

$$\begin{aligned}\gamma\psi^q(x,y) = \phi\psi^8(x,y) &= \phi(x + w + 1, y + w^4x^2 + s_1x + s_0 + w^2) \\ &= (x + w, y + w^4x^2 + s_1x + s_0 + w^2 + x^2 + w^2 + 1 + 1) = \psi(x,y).\end{aligned}$$

Similarly, when $3m \equiv 3 \pmod 4$ we have

$$\begin{aligned}\gamma\psi^q(x,y) = \phi^3\psi^{8^3}(x,y) &= -\phi(x + w + 1, y + w^4x^2 + s_1x + s_0 + w^2 + 1) \\ &= -(x + w, y + w^4x^2 + s_1x + s_0 + w^2 + 1 + x^2 + w^2 + 1 + 1) = \psi(x,y).\end{aligned}$$

This proves the lemma. □

5.5 Functions

Let $f_{8,P}$ be a function such that $(f_{8,P}) = 8(P) - ([8]P) - 7(\infty)$. We show in appendix B that, for $\psi(Q) = (x, y)$:

$$f_{8,P}(\psi(Q)) = \frac{(y + b_4(x))^2(y + b_8''(x))}{a_4'(x)^2 a_8'(x)}$$

where

$$b_4(x) = x^3 + (x_P^8 + x_P^4)x^2 + (x_P^4)x + y_P^4$$

and

$$b_8''(x) = (x_P^{32} + 1)x^2 + (x_P^{32} + x_P^{16})x + (y_P^{16} + x_P^{16} + x_P^{48} + 1).$$

The denominator $a_4'(x)^2 a_8'(x)$ can be ignored for the usual reasons.

We can write $f_{8,P} = \alpha\beta$ where $\alpha = (y + b_4(x))^2 \circ \psi$ and $\beta = (y + b_8'') \circ \psi$.

5.6 Miller's algorithm

Miller's algorithm (using the Octupling formulae) computes (cf. eq. 3):

$$\langle P, \psi(Q)\rangle_{2^{6m}+1} = \prod_{i=0}^{2m-1} f_{8,[8^i]P}(\psi(Q))^{8^{2m-1-i}}.$$

Following our general presentation of the η_T pairing, we will instead compute

$$\eta_T(P,Q) = \prod_{i=0}^{m-1} f_{8,[8^i]P}(\psi(Q))^{8^{m-1-i}}.$$

By Lemma 2 the pairing is bilinear, so that (for $q = 2^{3m}$):

$$\langle P, \psi(Q)\rangle_{2^{6m}+1} = \eta_T(P,Q)^q \cdot \eta_T([q]P,Q) = \eta_T(P,Q)^{2q}.$$

In the Appendix we show how this product may be efficiently computed, leading to a very fast pairing implementation on genus 2 curves in characteristic two. Details of timings are given in Section 7.

Note that to compute a pairing between general elements $D_1 = (P_1)+(P_2)-2(\infty)$ and $D_2 = (Q_1)+(Q_2)-2(\infty)$ in the divisor class group of C it is necessary to compute

$$e(P_1,Q_1)e(P_1,Q_2)e(P_2,Q_1)e(P_2,Q_2).$$

Obviously there are some optimisations available (for example, the final exponentiation only needs to be performed once and the calculation of the functions can be shared for both $e(P_1,Q_1)$ and $e(P_1,Q_2)$), but nevertheless, a general pairing will be much slower than a pairing of two special divisors. This can be exploited in protocol design. For example, one can set-up the Boneh-Franklin IBE system so that encryption is fast (by arranging that identities map to special divisors) at some cost to decryption.

Appendix C discusses implementation issues in detail.

6 The Rubin-Silverberg approach

Rubin and Silverberg [13] (also see [14, 16]) have proposed an alternative way to view pairings on Abelian varieties. Their method can be thought of as a method for computing pairings on trace zero subvarieties of Weil restrictions of elliptic curves. A simpler way to think about their method is as a form of point compression for pairings on elliptic curves. We recall their approach in the case of characteristic 2.

Let $E_b : y^2 + y = x^3 + x + b$ with $b = 0, 1$ over $\mathbb{F}_2$ be the supersingular elliptic curves with embedding degree $k = 4$. The idea of Rubin and Silverberg is to compute pairings with points defined over $\mathbb{F}_{2^{3m}}$ where m is coprime to 12. This means that pairing values lie in $\mathbb{F}_{2^{12m}}$.

To transmit group elements, Rubin and Silverberg propose a compression method so that the element is represented using only 2 elements in $\mathbb{F}_{2^m}$. Hence,

the bandwidth is about $2m$ bits but the finite field security is 2^{12m}, which corresponds to 'security multiplier 6'.

The group $E(\mathbb{F}_{2^{3m}})$ clearly has $E(\mathbb{F}_{2^m})$ as a subgroup. Indeed, we can write

$$E(\mathbb{F}_{2^{3m}}) \cong E(\mathbb{F}_{2^m}) \times A$$

where A is a finite group and one can check that the order of A is $2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$. Note that this agrees with the group orders in Table 2.

The following result is an important classification of A.

Lemma 3. *Let m be coprime to 12. Let* Tr *be the trace map with respect to $\mathbb{F}_{2^{3m}}/\mathbb{F}_{2^m}$. Then $A = \{P \in E(\mathbb{F}_{2^{3m}}) : \mathrm{Tr}(P) = 0\}$.*

The method is to perform pairing computations with the curve E over $\mathbb{F}_{2^{3m}}$ so that the pairing values lie in $\mathbb{F}_{2^{12m}}$. Suitable group orders are the same as in the genus 2 case (e.g. $m = 103$ with a 192-bit subgroup).

Write $\mathbb{F}_{2^{3m}}$ as $\mathbb{F}_{2^m}(\theta)$ where $\theta^3 = \theta + 1$. So points in $E(\mathbb{F}_{2^{3m}})$ are represented as (x, y) where x is represented as a triple (x_0, x_1, x_2) over $\mathbb{F}_{2^m}$ with respect to the basis $\{1, \theta, \theta^2\}$.

To transmit a point we first apply point compression so that we need send only the x-coordinate and a single bit determining the sign (sometimes even this bit can be removed). Then to transmit the x-coordinate just send x_0 and x_1 (and possibly another bit).

To recover (decompress) we must do the following: Given x_0 and x_1 compute an element $x_2 \in \mathbb{F}_{2^m}$ such that there is a point P with x-coordinate (x_0, x_1, x_2) which satisfies $\mathrm{Tr}(P) = 0$.

In this case the trace is $\mathrm{Tr}(P) = P + \pi(P) + \pi^2(P)$ where π is the 2^m-power Frobenius map. So the condition is that $P, \pi(P)$ and $\pi^2(P)$ sum to zero, or in other words, lie on a straight line. The decompression procedure is to deduce which x_2 ensures that there is a line $l(x, y) = 0$ through the three points.

Let $P = (x_P, y_P)$ and write $l(x, y) = y + u_0 x + u_1$ where $u_0, u_1 \in \mathbb{F}_{2^{3m}}$. Define $\bar{l}(x, y) = l(x, y) + 1$. Then

$$l(x,y)\bar{l}(x,y) = y^2+y+(u_0x+u_1)+(u_0x+u_1)^2 = x^3+u_0^2x^2+(u_0+1)x+(u_1^2+u_1+b).$$

Also,

$$l(x,y)\bar{l}(x,y) = (x - x_P)(x - x_{\pi(P)})(x - x_{\pi^2(P)}) = x^3 + Tx^2 + Sx + N$$

from which we deduce that $T = S^2 + 1$. One can check that $T = x_0$ and $S = x_0^2 + x_1^2 + x_1x_2 + x_2^2$. Hence x_2 is a solution to the equation

$$y^4 + x_1y^2 + (x_0^4 + x_0 + 1 + x_1^2) = 0.$$

Solving this equation involves solving a quadratic and then taking square roots. A single bit is needed to distinguish the two roots of the quadratic and to ensure a unique solution to the decompression process.

The total cost is solving a quadratic and then taking a square root, plus solving another quadratic to recover the y-coordinate of the point.

From a performance point of view it is essential to compare the running time of the pairing computation on the genus 2 curve with the Rubin-Silverberg method. In a general implementation, where we may be required to compute the pairing of general divisors on the genus 2 curve, then the Rubin-Silverberg approach may be superior.

7 Experimental results

7.1 The η_T pairing

The Jacobian of the supersingular genus 2 curve $y^2+y = x^5+x^3+b$, defined over $\mathbb{F}_2$, is a 2-dimensional Abelian variety over $\mathbb{F}_{2^m}$ with embedding degree $k = 12$. Another way to get this Abelian variety is with the Rubin-Silverberg approach, by taking the $k = 4$ elliptic case and using the Rubin-Silverberg construction with $r = 3$, to obtain a 2-dimensional Abelian variety with embedding degree $3 \times 4 = 12$. We can use this to compare the performance of the η_T pairing in the elliptic and genus 2 case, by comparing the running time of the pairing using the Jacobian of the curve over $\mathbb{F}_{2^m}$ with using $E(\mathbb{F}_{2^{3m}})$.

Note: this does not take into account the cost of conversion between the Rubin-Silverberg Abelian variety representation and the elliptic curve over the larger field. The base-field size is chosen as $\mathbb{F}_{2^{103}}$ because the Jacobian has order 13 times a 192-bit prime.

Table 4. Running times for pairing computation.

case	curve	optimisation	pairing time (ms)
1	$E(\mathbb{F}_{2^{239}})$	algorithm 2	3.16
2	$E(\mathbb{F}_{2^{3\cdot 103}})$	algorithm 2	5.83
3	$C(\mathbb{F}_{2^{103}})$	single precomp	4.53
4	$C(\mathbb{F}_{2^{103}})$	double precomp	3.47
5	$C(\mathbb{F}_{2^{103}})$	unroll	3.14
6	$C(\mathbb{F}_{2^{103}})$	integrate $2q$-power	3.20
7	$E(\mathbb{F}_{3^{97}})$	see [9]	4.05

Table 4 gives some running times for calculating the η_T pairing for two points. Cases 1 and 2 illustrate the elliptic curve algorithm over the field $\mathbb{F}_{2^{3\cdot 103}}$. Case 3 uses the genus 2 algorithm over the field $\mathbb{F}_{2^{103}}$ with the formulae given in Appendix C, and precomputes the values $x_P^{(i)}$ and $y_P^{(i)}$. The precomputation time is included in the running time. Case 4 does this precomputation for both input points. Case 5 does likewise, except that it unrolls the pairing function computation, also described in Appendix C. Case 6 integrates the $2q$ exponentiation into

the formulae, and also applies all previous optimisations. For comparison purposes, we finally provide the timings of an optimised pairing in characteristic 3 as case 7.

As can be seen from the table, the fastest hyperelliptic implementation is about 46% faster than the elliptic curve case in characteristic 2, and about 16% faster than the elliptic curve characteristic 3 case. Note that bringing the final $2q$ exponentiation into the formulae does not seem to provide any speedup.

7.2 Scalar Multiplication

Appendix A gives explicit formulae for calculating the group law for supersingular hyperelliptic curves of genus 2 of the form $y^2 + y = x^5 + x^3 + d$ defined over $\mathbb{F}_2$. They are based on explicit formulae given in [11], and contain additional optimisations.

Scalar multiplication for a general divisor over the above curve is significantly faster using the octupling formulae given in Appendix A and 8-ary windowing, than using a plain sliding-window method (with window width 4). Table 5 gives the observed running times for these two methods. As can be seen, the octupling method is more than twice as fast as the sliding-window method. For comparison, we also include the running time for scalar multiplication over $\mathbb{F}_{3^{97}}$.

Table 5. Running times over $\mathbb{F}_{2^{103}}$.

Method	time (ms)
Sliding window	1.603
Octupling	0.631
9-ary over $\mathbb{F}_{3^{97}}$	1.028

One of the potential advantages of using hyperelliptic curves is that the base field can be much smaller than that required for an elliptic curve, for the same level of security. Great potential savings can be realised if an element of the base field can be represented in a single machine word, rather than using a multi-precision representation, and for comparison with elliptic curves we regard it as quite "fair" to try to exploit this feature.

So in implementing arithmetic in the field $\mathbb{F}_{2^{103}}$ we take advantage of the 128-bit registers available to those processors, like the Pentium IV, which support the SSE2 instruction set, and have written a special function to carry out field multiplication using SSE2 instructions. This is twice as fast as a standard multi-precision implementation, and improves the overall timings by about 50%.

All timings were done on a Pentium IV running at 3 GHz.

8 Conclusions

We have presented the eta pairing approach to compute pairings on supersingular curves. This approach generalises and clarifies the Duursma-Lee algorithm. We have provided full examples of the method in characteristic 2 for genus 1 and 2, which turn out to be very efficiently implementable; in the former case we also described more efficient formulae for conventional arithmetic.

References

1. P. S. L. M. Barreto. The well-tempered pairing. Bochum, Germany, 2004. 8th Workshop on Elliptic Curve Cryptography – ECC'2004. Invited talk.
2. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.
3. P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *Journal of Cryptology*, 17(4):321–334, 2004.
4. R. Dutta, R. Barua, and P. Sarkar. Pairing-based cryptography: A survey. Cryptology ePrint Archive, Report 2004/064, 2004. `http://eprint.iacr.org/2004/064`.
5. I. Duursma and H.-S. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *Advances in Cryptology – Asiacrypt'2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag, 2003.
6. K. Fong, D. Hankerson, J. López, and A. Menezes. Field inversion and point halving revisited. Technical report CORR 2003-18, University of Waterloo, 2002.
7. G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Math. Comp.*, 52:865–874, 1994.
8. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.
9. R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. Cryptology ePrint Archive, Report 2004/132, 2004.
10. S. Kwon. Efficient Tate pairing computation for supersingular elliptic curves over binary fields. Cryptology ePrint Archive, Report 2004/303, 2004. `http://eprint.iacr.org/2004/303`.
11. T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. In *Applicable Algebra in Engineering, Communication and Computing*, Online publication. Springer-Verlag, 2004. `http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s0%0200-004-0154-8`.
12. V. S. Miller. Short programs for functions on curves. Unpublished manuscript, 1986. `http://crypto.stanford.edu/miller/miller.pdf`.
13. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 336–353. Springer-Verlag, 2002.
14. K. Rubin and A. Silverberg. Using primitive subgroups to do more with fewer bits. In *Algorithmic Number Theory –ANTS VI*, volume 3076 of *Lecture Notes in Computer Science*, pages 18–41. Springer-Verlag, 2004.

15. M. Scott and Paulo S. L. M. Barreto. Compressed pairings. In *Proceedings*, Lecture Notes in Computer Science, Santa Barbara, USA, 2004. Advances in Cryptology – Crypto'2004, Springer-Verlag. to appear.
16. A. Silverberg. Compression for trace zero subgroups of elliptic curves. Preprint, 2004. Available from `http://www.math.uci.edu/~asilverb/bibliography/compress.pdf`.
17. J. H. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer-Verlag, Berlin, Germany, 1986.

A Supersingular hyperelliptic arithmetic

Tanja Lange describes general explicit formulae for arithmetic on the Jacobian of the hyperelliptic curve

$$y^2 + (h_2x^2 + h_1x + h_0)y = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$$

over any finite field. Here we consider the special case over $\mathbb{F}_{2^m}$ with odd m where $h_2 = h_1 = 0$, $h_0 = 1$, $f_4 = f_2 = f_1 = 0$, $f_3 = 1$, $f_0 = d \in \mathbb{F}_2$, i.e. the supersingular hyperelliptic curve

$$y^2 + y = x^3 + x + d.$$

We assume divisors in Mumford representation, hence as pairs $[u, v]$ for polynomials u and v where u is monic and $\deg(u) > \deg(v)$; the zero element is denoted $[1, 0]$. Algorithm 3 describes divisor doubling, algorithms 4 and 5 describe divisor addition, and algorithm 6 describes divisor octupling.

B The hyperelliptic function $f_{8,P}$

We now derive an explicit expression for the function $f_{8,P}$ needed for Miller's algorithm on the supersingular hyperelliptic curve $C_b : y^2 + y = x^5 + x^3 + b$.

Let $P = (x_P, y_P)$. We will consider divisors $D_n = n(P) - n(\infty)$. To achieve this we will consider the **reduced** divisor (via Cantor's algorithm) D'_n which is equivalent to D_n. We will consider functions such that $D_n = D'_n + (f_n)$.

The divisor $D_1 = (P) - (\infty)$ has Mumford representation

$$(a_1(x), b_1(x)) = (x + x_P, y_P).$$

We take the function $f_1 = 1$.

Now consider $D_2 = 2(P) - 2(\infty)$. One can show that this divisor has Mumford representation $(a_2(x), b_2(x)) = (x^2 + x_P^2, (x_P^4 + x_P^2)x + y_P^2)$. This divisor is reduced (so no reduction step in Cantor's algorithm is performed). Hence $D'_2 = D_2$ and so the function f_2 may be chosen to be 1.

Now consider $D_4 = 4(P) - 4(\infty)$. The Mumford representation (after performing the composition step of Cantor's algorithm) is

$$(a_4(x), b_4(x)) = (x^4 + x_P^4, x^3 + (x_P^8 + x_P^4)x^2 + (x_P^4)x + y_P^4).$$

Algorithm 3 Doubling of a divisor $[u, v]$

INPUT: divisor $[u, v]$.
OUTPUT: $[u', v'] = 2[u, v]$.

if $\deg(u) = 0$ **then** ▷ $[u, v] = [1, 0]$
  $[u', v'] \leftarrow [1, 0]$
else if $\deg(u) = 1$ **then** ▷ $[u, v] = [x + u_0, v_0]$
  $u'_0 \leftarrow u_0^2$, $v'_1 \leftarrow {u'_0}^2 + u'_0$, $v'_0 \leftarrow v_0^2 + d$
  $[u', v'] \leftarrow [x^2 + u'_0, v'_1 x + v'_0]$ ▷ (3S)
else ▷ $[u, v] = [x^2 + u_1 x + u_0, v_1 x + v_0]$
  if $u_1 = 0$ **then**
    $s'_0 \leftarrow v_1^2$, $l'_0 \leftarrow u_0 s'_0$
    $u'_0 \leftarrow {s'_0}^2$
    $w_1 \leftarrow s'_0 + 1, w_0 \leftarrow w_1 + u'_0 + u_0, w_1 \leftarrow u'_0 w_1 + l'_0$,
    $v'_1 \leftarrow w_0 + v_1, v'_0 \leftarrow w_1 + v_0 + 1$
    $[u', v'] \leftarrow [x^2 + x + u'_0, v'_1 x + v'_0]$ ▷ (2S, 3M)
  else if $u_1 = 1$ **then**
    $s'_0 \leftarrow v_1^2, u'_0 \leftarrow {s'_0}^2$
    $v'_0 \leftarrow s'_0(u'_0 + {u'_0}^2 + u_0) + u'_0 v_1 + v_0 + 1$
    $[u', v'] \leftarrow [x + u'_0, v'_0]$ ▷ (3S, 3M)
  else ▷ $u_1 \notin \{0, 1\}$
    $s'_1 \leftarrow 1 + u_1^2$, $s'_0 \leftarrow u_1 s'_1 + v_1^2$
    $w_1 \leftarrow 1/s'_1$, $w_0 \leftarrow s'_0 w_1$
    $l'_2 \leftarrow u_1 + w_0$, $l'_1 \leftarrow u_1 w_0 + u_0$, $l'_0 \leftarrow u_0 w_0$
    $u'_1 \leftarrow w_1^2$, $u'_0 \leftarrow w_0^2$
    $w_1 \leftarrow l'_2 + u'_1$, $w_0 \leftarrow u'_1 w_1 + u'_0 + l'_1$, $w_1 \leftarrow u'_0 w_1 + l'_0$,
    $v'_1 \leftarrow w_0 s'_1 + v_1$, $v'_0 \leftarrow w_1 s'_1 + v_0 + 1$
    $[u', v'] \leftarrow [x^2 + u'_1 x + u'_0, v'_1 x + v'_0]$ ▷ (I, 4S, 8M)
  end if
end if

Algorithm 4 Addition of divisors $[u, v_1]$ and $[u, v_2]$, $v_1 \neq v_2$

INPUT: divisors $[u, v_1]$ and $[u, v_2]$ with $v_1 \neq v_2$.
OUTPUT: $[u', v'] = [u, v_1] + [u, v_2]$.

if $v_2 = v_1 + 1$ **then** ▷ $[u, v_1] = -[u, v_2]$
  $[u', v'] \leftarrow [1, 0]$
else ▷ $u = x^2 + u_1 x + u_0$, $v_i = v_{i1} x + v_{i0}$, $[u, v_i] = (P_1) + ((-1)^i P_2) - 2(\infty)$
  $w_0 \leftarrow (v_{10} + v_{20}) u_1$, $z_0 \leftarrow v_{11} w_0 + v_{10}$
  $u'_0 \leftarrow w_0^2$, $v'_1 \leftarrow {u'_0}^2 + u'_0$, $v'_0 \leftarrow z_0^2 + d$
  $[u', v'] \leftarrow [x^2 + u'_0, v'_1 x + v'_0]$ ▷ (3S, 2M)
end if

Algorithm 5 Addition of divisors $[u_1, v_1]$ and $[u_2, v_2]$, $u_1 \neq u_2$

INPUT: divisors $[u_1, v_1]$ and $[u_2, v_2]$ with $\deg(u_1) \leqslant \deg(u_2)$ and $u_1 \neq u_2$.
OUTPUT: $[u', v'] = [u_1, v_1] + [u_2, v_2]$.

1: **if** $\deg(u_1) = 0$ **then** ▷ $[u_1, v_1] = [1, 0]$
2: $[u', v'] \leftarrow [u_2, v_2]$
3: **else if** $\deg(u_1) = 1$ **then** ▷ $u_1 = x + u_{10}$, $v_1 = v_{10}$
4: **if** $\deg(u_2) = 1$ **then** ▷ $u_2 = x + u_{20}$, $v_2 = v_{20}$
5: $w_0 \leftarrow (u_{10} + u_{20})^{-1}$, $z_1 \leftarrow w_0(v_{10} + v_{20})$, $z_0 \leftarrow w_0(v_{20}u_{10} + v_{10}u_{20})$
6: $[u', v'] \leftarrow [x^2 + (u_{10} + u_{20})x + u_{10}u_{20}, z_1x + z_0]$ ▷ (I, 5M)
7: **else** ▷ $u_2 = x^2 + u_{21}x + u_{20}$, $v_2 = v_{21}x + v_{20}$
8: $r \leftarrow u_{10}(u_{10} + u_{21}) + u_{20}$, $t \leftarrow v_{21}u_{10} + v_{20} + v_{10}$
9: **if** $r \neq 0$ **then**
10: $s_0 \leftarrow r^{-1}t$, $w_0 \leftarrow s_0^2$
11: $u'_1 \leftarrow u_{10} + u_{21} + w_0$, $u'_0 \leftarrow 1 + u_{21}^2 + r + w_0(u_{10} + u_{21})$
12: $v'_1 \leftarrow s_0(u'_1 + u_{21}) + v_{21}$, $v'_0 \leftarrow s_0(u'_0 + u_{20}) + v_{20} + 1$
13: $[u', v'] \leftarrow [x^2 + u'_1x + u'_0, v'_1x + v'_0]$ ▷ (I, 2S, 6M)
14: **else if** $t = 1$ **then**
15: $[u', v'] \leftarrow [x + u_{21} + u_{10}, v_{21}(u_{21} + u_{10}) + v_{20}]$ ▷ (3M)
16: **else if** $u_{21} \neq 0$ **then**
17: $[u', v'] \leftarrow [x + u_{21} + u_{10}, v_{21}(u_{21} + u_{10}) + v_{20}] + 2[u_1, v_1]$
18: **else**
19: $[u', v'] \leftarrow [u_1, v_1 + 1] + 2[u_2, v_2]$
20: **end if**
21: **end if**
22: **else**
23: $z_0 \leftarrow u_{10} + u_{20}$, $z_1 \leftarrow u_{11} + u_{21}$, $z_2 \leftarrow u_{11}z_1 + z_0$
24: $w_0 \leftarrow v_{10} + v_{20}$, $w_1 \leftarrow v_{11} + v_{21}$, $w_2 \leftarrow z_1w_1$
25: $r \leftarrow z_0z_2 + z_1^2u_{10}$, $s'_1 \leftarrow z_1w_0 + z_0w_1$, $s'_0 \leftarrow z_2w_0 + u_{10}w_2$
26: **if** $r \neq 0$ **then**
27: **if** $s'_1 \neq 0$ **then**
28: $w_1 \leftarrow (rs'_1)^{-1}$, $w_2 \leftarrow rw_1$, $w_3 \leftarrow {s'_1}^2w_1$,
29: $w_4 \leftarrow rw_2$, $w_5 \leftarrow w_4^2$, $s''_0 \leftarrow s'_0w_2$
30: $l'_2 \leftarrow u_{21} + s''_0$, $l'_1 \leftarrow u_{21}s''_0 + u_{20}$, $l'_0 \leftarrow u_{20}s''_0$
31: $u'_1 \leftarrow z_1 + w_5$, $u'_0 \leftarrow {s''_0}^2 + z_2 + z_1w_5$
32: $w_1 \leftarrow l'_2 + u'_1$, $w_2 \leftarrow u'_1w_1 + u'_0 + l'_1$, $v'_1 \leftarrow w_2w_3 + v_{21}$
33: $w_2 \leftarrow u'_0w_1 + l'_0$, $v'_0 \leftarrow w_2w_3 + v_{20} + 1$
34: $[u', v'] \leftarrow [x^2 + u'_1x + u'_0, v'_1x + v'_0]$ ▷ (I, 4S, 20M)
35: **else**
36: $t \leftarrow r^{-1}$, $s_0 \leftarrow s'_0t$, $w_0 \leftarrow s_0^2$, $w_1 \leftarrow s_0(u_{11} + w_0) + v_{21}$
37: $[u', v'] \leftarrow [x + u_{11} + u_{21} + w_0, u'_0w_1 + s_0 + v_{20} + 1]$ ▷ (I, 2S, 3M)
38: **end if**
39: **else let** $\gcd(u_1, u_2) = x + w_0$
40: $w_1 \leftarrow u_{11} + w_0$, $w_2 \leftarrow u_{21} + w_0$, $z_1 \leftarrow v_{11}w_1 + v_{10}$, $z_2 \leftarrow v_{21}w_2 + v_{20}$
41: **if** $v_{11}w_0 + v_{10} = v_{21}w_0 + v_{20} \equiv z_0$ **then**
42: $D_0 \leftarrow 2[x + w_0, z_0]$, $D_1 \leftarrow [x + w_1, z_1] + D_0$, $D_2 \leftarrow [x + w_2, z_2] + D_1$
43: $[u', v'] \leftarrow D_2$
44: **else**
45: $[u', v'] \leftarrow [x + w_1, z_1] + [x + w_2, z_2]$
46: **end if**
47: **end if**
48: **end if**

Algorithm 6 Octupling of a divisor $[u, v]$

INPUT: divisor $[u, v]$.
OUTPUT: $[u', v'] = 8[u, v]$.
1: **if** $\deg(u) = 2$ **then** $\triangleright$ $[u, v] = [x^2 + u_1 x + u_0, v_1 x + v_0]$
2: $\quad [u', v'] \leftarrow [x^2 + u_1^{64} x + (u_1 + u_0 + 1)^{64}, (v_1 + u_1)^{64} x + (u_1 + u_0 + v_1 + v_0 + 1)^{64}]$
3: **else if** $\deg(u) = 1$ **then** $\triangleright$ $[u, v] = [x + u_0, v_0]$
4: $\quad [u', v'] \leftarrow [x + (u_0 + 1)^{64}, (v_0 + u_0^2 + 1)^{64}]$
5: **else** $\triangleright$ $[u, v] = [1, 0]$
6: $\quad [u', v'] \leftarrow [1, 0]$
7: **end if**

This divisor is not reduced. We have $(b_4^2 + b_4 + x^5 + x^3 + b)/a_4(x) = a_4'(x) = x^2 + x + (x_P^{16} + x_P^8)$ and $b_4'(x) := b_4(x) + 1 \bmod a_4'(x) = (x_P^{16} + 1)x + (y_P^8 + x_P^8 + x_P^{24} + 1)$.

We must consider functions and divisors. The divisor D_4 is equivalent to the divisor $D_4' = E - 2(\infty)$ where E is effective. The divisor D_4' has the Mumford representation $(a_4'(x), b_4'(x))$ given above. Denote by $\overline{E}$ the 'negative' of E. The function $a_4(x)$ has divisor $4(P) + 4(\overline{P}) - 8(\infty)$ while the function $a_4'(x)$ has divisor $E + \overline{E} - 4(\infty)$. The function $y + b_4(x)$ has divisor $4(P) + \overline{E} - 6(\infty)$ while the function $y + b_4(x) + 1$ has divisor $4(\overline{P}) + E - 6(\infty)$. It follows that

$$((y + b_4(x))/a_4'(x)) = 4(P) - E - 2(\infty).$$

Hence we define $f_4 = (y + b_4(x))/a_4'(x)$ and we have $D_4 = D_4' + (f_4)$.

Now for the final step (thankfully!). We double the divisor D_4' using Cantor's composition rule to obtain $D_8'' = 2E - 4(\infty)$. Note that $D_8 = 2D_4 = 2(D_4' + (f_4)) = D_8'' + (f_4^2)$. One computes the Mumford representation of D_8'' to be

$$(a_8''(x), b_8''(x)) = (a_4'(x)^2, (x_P^{32} + 1)x^2 + (x_P^{32} + x_P^{16})x + (y_P^{16} + x_P^{16} + x_P^{48} + 1))$$

and one can check that $((b_8'')^2 + b_8'' + f(x))/a_8''(x) = a_8'(x) = (x + (x_P^{64} + 1))$. Thus, $b_8'(x) := b_8''(x) + 1 \bmod a_8'(x) = y_P^{64} + x_P^{128} + 1$. Define $[8]P = (x_P^{64} + 1, y_P^{64} + x_P^{128} + 1)$. We obtain $D_8' = ([8]P) - (\infty)$ which confirms the octupling formula for the point $[8]P$.

We now consider principal divisors. As before, $y + b_8''(x)$ has divisor $2E + (\overline{[8]P}) - 5(\infty)$ and $(a_8'(x)) = (\overline{[8]P}) + ([8]P) - 2(\infty)$. Hence we have $D_8'' = D_8' + (f_8')$ where $f_8' = (y + b_8''(x))/a_8'(x)$.

Putting it all together, we get

$$(f_8) = 8(P) - ([8]P) - 7(\infty)$$

where

$$f_8 = \left(\frac{y + b_4(x)}{a_4'(x)}\right)^2 \frac{y + b_8''(x)}{a_8'(x)}.$$

C Efficient implementation of pairings in genus 2

C.1 Precomputation

We will precompute a table of powers of x_P and y_P (these are the initial input values for the point P) labelled as

$$x_P^{(i)} = \pi^i(x_P) = x_P^{2^i}, \quad \text{and} \quad y_P^{(i)} = \pi^i(y_P) = y_P^{2^i}$$

for $i = 0, 1, \ldots, m-1$.

We focus on computing the term $f_{8,P}(\psi(Q))$ (i.e. we do not bring Frobenius actions into this computation).

Note that, at loop iteration i, the current value of the x-coordinate of $[2^{3i}]P$ can be written in terms of the precomputed initial values as

$$x_P^{(6i)} + \gamma_1(i)$$

where $\gamma_1(i)$ is 1 when i is odd and 0 otherwise. Similarly, the current value of the y-coordinate of $[2^{3i}]P$ is

$$y_P^{(6i)} + \gamma_1(i)x_P^{(6i+1)} + \gamma_3(i)$$

where $\gamma_3(i) = 1$ when $i \equiv 1, 2 \pmod 4$ and 0 otherwise.

Obviously, in the above the exponents $6i$ in $x_P^{(6i)}$ are taken modulo m. One sees that they wrap around rapidly.

C.2 The α factor

Write $\alpha = (y + b_4(x))^2 \circ \psi$ as a function of (x_Q, y_Q). We have $(y + b_4(x)) \circ \psi =$

$$y + s_2x^2 + s_1x + s_0 + (x+w)^3 + (x_P^8 + x_P^4)(x+w)^2 + (x_P^4)(x+w) + y_P^4$$

and squaring gives

$$y^2+s_2^2x^4+s_1^2x^2+s_0^2+x^6+x^4w^2+x^2w^4+w^6+(x_P^{16}+x_P^8)(x^4+w^4)+(x_P^8)(x^2+w^2)+y_P^8.$$

Now, $s_2^2 = (w^4+1)^2 = w$ and $s_1^2 = (w^2+w^4)^2 = w^4+w+1$. Also, $s_0^2 = s_0 + w^5 + w^3$.

Expressing as a 12-tuple we get α as follows: The first component is

$$y^2 + x^2 + x^6 + 1 + (x_P^{16} + x_P^8)x^4 + x_P^8x^2 + y_P^8$$

and the remaining components are

$$(x^4 + x^2, x^4 + 1 + x_P^8, 1 + 1, x^2 + x^2 + x_P^{16} + x_P^8, 1 + 1, 1, 0, 0, 0, 0, 0)$$

which can be slightly simplified.

Finally, we want to evaluate this on (x_Q, y_Q) and to replace the current value for x_P with the precomputed values. We obtain the 12-tuple with first component

$$y_Q^2+x_Q^6+(x_P^{(6i+4)}+x_P^{(6i+3)})x_Q^4+(x_P^{(6i+3)}+1+\gamma_1(i))x_Q^2+y_P^{(6i+3)}+\gamma_1(i)x_P^{(6i+4)}+\gamma_3(i)+1$$

and remaining components

$$(x_Q^4 + x_Q^2, x_Q^4 + x_P^{(6i+3)} + \gamma_1(i) + 1, 0, x_P^{(6i+4)} + x_P^{(6i+3)}, 0, 1, 0, 0, 0, 0, 0).$$

C.3 The β factor

We then do a similar thing for $\beta = (y + b_8'') \circ \psi$. We have

$$\beta = y+s_2x^2+s_1x+s_0+(x_P^{32}+1)(x+w)^2+(x_P^{32}+x_P^{16})(x+w)+(y_P^{16}+x_P^{16}+x_P^{48}+1).$$

We expand $s_2 = 1 + w^4$ etc and write $x_P^{16} + x_P^{48} = x_P^{16}(1 + x_P^{32})$. Hence, β can be expressed as a 12-tuple with first component

$$y + (x_P^{32})x^2 + (x_P^{32} + x_P^{16})x + y_P^{16} + x_P^{16}(1 + x_P^{32}) + 1$$

and remaining components

$$(x_P^{32} + x_P^{16}, x + x_P^{32} + 1, 0, x^2 + x, 0, 1, 0, 0, 0, 0, 0).$$

Finally, we substitute $(x, y) = (x_Q, y_Q)$ and insert the precomputed values $x_P = x_P^{(6i)} + \gamma_1(i)$ and $y_P = y_P^{(6i)} + \gamma_1(i)x_P^{(6i+1)} + \gamma_3(i)$. Using the formula $\gamma_1(i)(1 + \gamma_1(i)) = 0$ gives the 12-tuple with first component

$$\begin{aligned} &y_Q + (x_P^{(6i+5)} + \gamma_1(i))x_Q^2 + (x_P^{(6i+5)} + x_P^{(6i+4)})x_Q \\ &+y_P^{(6i+4)} + x_P^{(6i+4)} \left(x_P^{(6i+5)} + \gamma_1(i) + 1\right) + \gamma_3(i) + 1. \end{aligned}$$

and remaining components

$$(x_P^{(6i+5)} + x_P^{(6i+4)}, x_Q + x_P^{(6i+5)} + \gamma_1(i) + 1, 0, x_Q^2 + x_Q, 0, 1, 0, 0, 0, 0, 0).$$

It remains to multiply the α and β together efficiently. But first we consider how to absorb the powers of 8 into the equations.

C.4 Absorbing powers of 8

The eta pairing $\eta_T(P, Q)$ can be expressed as the product

$$\prod_{i=0}^{m-1} f_{8,2^{3i}P}(\psi(Q))^{2^{3(m-1-i)}}$$

where $f_{8,2^{3i}P} = \alpha\beta$ as described previously. The goal of this section is to write this as

$$\prod_{i=0}^{m-1} f_i$$

where each f_i is an equation which has the 2-power Frobenius action already brought into the equation. Using the formulae for α and β above we will compute $\alpha^{2^{3(m-1-i)}}$ and $\beta^{2^{3(m-1-i)}}$.

To achieve this efficiently requires precomputation of the 2-power Frobenius orbit of the point Q, so define for $i = 0, 1, \ldots, m-1$

$$x_Q^{(i)} = x_Q^{2^i} \qquad \text{and} \qquad y_Q^{(i)} = y_Q^{2^i}.$$

The most delicate part of the argument is handling how w and s_0 behave under powering by $2^{3(m-1-i)}$. Recall that $w^8 = w + 1$ from which we deduce

$$\begin{aligned} s_0^8 &= s_0 + w^2 \\ s_0^{8^2} &= s_0 + 1 \\ s_0^{8^3} &= s_0 + w^2 + 1 \end{aligned}$$

Note that m is coprime to 12 and so is odd. We have $w^8 = w + 1$ and so, since $m - 1 - i \equiv i \pmod 2$ we have $w^{2^{3(m-1-i)}} = w + \gamma_1(i)$. The same formula holds when w is replaced by w^2 or w^4. For s_0 note that if $m \equiv 1 \pmod 4$ then

$$s_0^{2^{3(m-1-i)}} = s_0 + \gamma_1(i)w^2 + \gamma_3(i)$$

while if $m \equiv 3 \pmod 4$ then

$$s_0^{2^{3(m-1-i)}} = s_0 + \gamma_1(i)w^2 + \gamma_3(i) + 1.$$

We denote by $\gamma_4(m, i)$ the value $\gamma_3(i)$ when $m \equiv 1 \pmod 4$ and $\gamma_3(i) + 1$ otherwise.

C.5 The α factor

The basic shape of the term α will be similar to previously, except a few extra terms due to equation (6). The process is simple, just bring the 2-power operation into the formula and simplify the 'exponents'.

The "constant" term will be

$$\begin{aligned} &y_Q^{(3m-2-3i)} + (x_Q^{(3m-2-3i)})^3 + (x_P^{(3i+1)} + x_P^{(3i)})x_Q^{(3m-1-3i)} \\ &+(x_P^{(3i)} + \gamma_1(i) + 1)x_Q^{(3m-2-3i)} + y_P^{(3i)} + \gamma_1(i)x_P^{(3i+1)} + \gamma_3(i) + 1 \end{aligned}$$

plus when $m - 1 - i$ is odd (i.e., when i is odd) another term must be added (coming from the fact that $(w^{2j})^8 = w^{2j} + 1$ and the s_0 term). We write this other term as

$$\gamma_1(i)\left(x_Q^{(3m-2-3i)} + 1 + \gamma_1(i) + x_P^{(3i+1)}\right) + \gamma_4(m, i).$$

We can apply $\gamma_1(i)(1 + \gamma_1(i)) = 0$, cancel various terms and simplify the cubing of $x_Q^{(3m-2-3i)}$. The expression simplifies to

$$y_Q^{(3m-2-3i)} + (x_P^{(3i+1)} + x_P^{(3i)})x_Q^{(3m-1-3i)} + (x_P^{(3i)} + 1 + x_Q^{(3m-1-3i)})x_Q^{(3m-2-3i)} + y_P^{(3i)} + \gamma_5(m)$$

where $\gamma_5(m) = 1$ if $m \equiv 1 \pmod 4$ and 0 otherwise.

The remaining terms are (note that there is an additional $\gamma_1(i)w^2$ term due to the s_0 term):

$$(x_Q^{(3m-1-3i)} + x_Q^{(3m-2-3i)})w + (x_Q^{(3m-1-3i)} + x_P^{(3i)} + 1)w^2 + (x_P^{(3i+1)} + x_P^{(3i)})w^4 + s_0.$$

As usual, the indices inside round brackets should be reduced modulo m to the range $\{0, 1, \ldots, m-1\}$.

In terms of checking, this can be easily done in an implementation: in loop iteration i then this value should be equal to $\alpha^{2^{3(m-1-i)}}$.

C.6 The β factor

We now consider the β factor. One sees that the "constant term" of $\beta^{2^{3(m-1-i)}}$ is

$$\begin{aligned} y_Q^{(3m-3-3i)} + (x_p^{(3i+2)} + \gamma_1(i))x_Q^{(3m-2-3i)} + (x_P^{(3i+2)} + x_P^{(3i+1)})x_Q^{(3m-3-3i)} \\ + y_P^{(3i+1)} + x_P^{(3i+1)}(1 + \gamma_1(i) + x_P^{(3i+2)}) + \gamma_3(i) + 1 \end{aligned}$$

plus

$$\gamma_1(i)\left(x_P^{(3i+1)} + x_Q^{(3m-2-3i)} + \gamma_1(i) + 1\right) + \gamma_4(m, i).$$

This simplifies to

$$\begin{aligned} y_Q^{(3m-3-3i)} + x_p^{(3i+2)}x_Q^{(3m-2-3i)} + (x_P^{(3i+2)} + x_P^{(3i+1)})x_Q^{(3m-3-3i)} \\ + y_P^{(3i+1)} + x_P^{(3i+1)}(1 + x_P^{(3i+2)}) + \gamma_5(m). \end{aligned}$$

The remaining terms are (again, including a $\gamma_1(i)w^2$ term)

$$(x_P^{(3i+2)} + x_P^{(3i+1)})w + (x_Q^{(3m-3-3i)} + x_P^{(3i+2)} + 1)w^2 + (x_Q^{(3m-2-3i)} + x_Q^{(3m-3-3i)})w^4 + s_0.$$

C.7 Unrolling the $\alpha\beta$ multiplication

Let $\alpha = a + bw + cw^2 + dw^4 + s_0$ and $\beta = e + fw + gw^2 + hw^4 + s_0$, where α and β are the terms given in the sections C.5 and C.6. Then:

$$\begin{aligned} \alpha\beta &= (a + bw + cw^2 + dw^4 + s_0)(e + fw + gw^2 + hw^4 + s_0) \\ &= ae + afw + agw^2 + ahw^4 + as_0 + \\ &\quad bew + bfw^2 + bgw^3 + bhw^5 + bws_0 + \\ &\quad cew^2 + cfw^3 + cgw^4 + ch(w^5 + w^3 + w^2 + 1) + cw^2s_0 + \\ &\quad dew^4 + dfw^5 + dg(w^5 + w^3 + w^2 + 1) + dh(w + 1) + dw^4s_0 + \\ &\quad es_0 + fws_0 + gw^2s_0 + hw^4s_0 + (s_0 + w^5 + w^3) \end{aligned}$$

(remember that $w^6 = w^5 + w^3 + w^2 + 1$ and $s_0^2 = s_0 + w^5 + w^3$). Using the basis described earlier, this gives us;

$$\begin{aligned}\alpha\beta = &(ae + ch + dg + dh) + \\ &(af + be + dh)w + \\ &(ag + bf + ce + ch + dg)w^2 + \\ &(bg + cf + ch + dg + 1)w^3 + \\ &(ah + cg + de)w^4 + \\ &(bh + ch + df + dg + 1)w^5 + \\ &(a + e + 1)s_0 + \\ &(b + f)ws_0 + \\ &(c + g)w^2 s_0 + \\ &(d + h)w^4 s_0\end{aligned}$$

We can eliminate some multiplications using Karatsuba in the following way. Let $dh = d \cdot h, dg = d \cdot g, ch = c \cdot h, cg = c \cdot g, ae = a \cdot e, bf = b \cdot f$. Then we can write:

$$\begin{aligned}\alpha\beta = &(ae + ch + dg + dh) \\ &((a + b) \cdot (f + e) + ae + bf + dh)w + \\ &((a + c) \cdot (g + e) + ae + cg + bf + ch + dg)w^2 + \\ &((b + c) \cdot (g + f) + bf + cg + ch + dg + 1)w^3 + \\ &((a + d) \cdot (h + e) + ae + dh + cg)w^4 + \\ &((b + d) \cdot (h + f) + bf + dh + ch + dg + 1)w^5 + \\ &(a + e + 1)s_0 + \\ &(b + f)ws_0 + \\ &(c + g)w^2 s_0 + \\ &(d + h)w^4 s_0\end{aligned}$$

□