Fast arithmetic and pairing evaluation on genus 2 curves

David Freeman
University of California, Berkeley
dfreeman@math.berkeley.edu

November 6, 2005

Abstract

We present two algorithms for fast arithmetic in Jacobians of genus 2 curves. The first speeds up the process of "double-and-add" by an estimated 5.5% over the standard algorithm. The second modifies the construction of the functions used to compute the Weil and Tate pairings in a way that saves 6 field multiplications per evaluation.

1 Introduction

Arithmetic on elliptic and hyperelliptic curves has been proposed as a basis for numerous cryptographic protocols. Basic group operations can be used to construct Diffie-Hellman key exchange and ElGamal signatures, while bilinear pairings such as the Weil and Tate pairings have been proposed for three-way key establishment, identity-based encryption, short signatures, and other applications. When analyzing an implementation of one of these protocols, it is important to know how long each step in the algorithm takes to compute, as the security of the system will depend on the ratio of the security parameters (e.g. key length) to computational time and power.

Eisenträger, Lauter, and Montgomery [ELM1] have given an algorithm that speeds up scalar multiplication on an elliptic curve by streamlining the algorithm for "double-and-add," a fundamental process used in computing scalar multiples in any group. We adapt their algorithm to curves of genus 2. Specifically, for two points P and Q on the Jacobian of a genus 2 curve, we eliminate an intermediate step in the computation of $2P + Q$ that costs three field multiplications. Our algorithm achieves an improvement of 5.5% over the standard algorithm.

As an application of their arithmetic improvement, Eisenträger, et al. give a method for speeding up evaluation of the functions used to compute the Weil and Tate pairings on elliptic curves. We generalize this method to curves of genus 2. Using our algorithm, constructing these functions requires 13 more multiplications than the standard algorithm, but evaluating them requires approximately 6 fewer. Thus our algorithm will speed up repeated evaluations of the pairing when one of the points is fixed and the other varies.

Acknowledgments

I think Kristin Lauter for suggesting I pursue this line of research and for inviting me to Microsoft Research to discuss these results. I also thank Ed Schaefer for providing helpful feedback on earlier drafts of this paper.

2 Double-and-add algorithm

Let C be a curve of genus 2 defined by the equation $y^2 = f(x)$, where $f(x)$ is a monic polynomial of degree 5. We represent a point P on the Jacobian variety $\mathrm{Jac}(C)$ by a pair of polynomials (a, b), where a is monic of degree at most 2 and b has degree less than the degree of a. (In most cases, a will be quadratic and b linear.) If (x_1, x_2) are the zeroes of a and we let $P_1 = (x_1, b(x_1))$, $P_2 = (x_2, b(x_2))$, then P is the divisor class $[(P_1) + (P_2) - 2(\infty)] \in \mathrm{Pic}^0(C)$. (If a is linear, we take one of these points to be the point at infinity.)

Now let P, Q be elements of $\mathrm{Jac}(C)$. We wish to compute $2P + Q$. We represent the points as follows: $P = (a_1, b_1), Q = (a_2, b_2), P + Q = (a_3, b_3), 2P + Q = (a_4, b_4)$. We assume for simplicity that $\gcd(a_1, a_2) = \gcd(a_1, a_3) = 1$, the a_i are quadratic, and that none of these four points is equal to the identity.[1] We first recall the standard algorithm for double-and-add on the Jacobian variety $\mathrm{Jac}(C)$.

Algorithm 1 ([Cantor]). *Let C be a curve of genus 2 defined by $y^2 = f(x)$ as above, and $P, Q \in \mathrm{Jac}(C)$ be represented by (a_i, b_i) also as above. The following algorithm takes input (f, a_1, b_1, a_2, b_2) and computes (a_4, b_4) representing the point $2P + Q$.*

1. *Use Euclid's algorithm to compute (linear) polynomials k_1, k_2 such that $k_1 a_1 + k_2 a_2 = 1$.*
2. *Let $\hat{a}_3 = a_1 a_2$ and $\hat{b}_3 = b_1 + k_1 a_1 (b_2 - b_1) \pmod{\hat{a}_3}$.*
3. *Let $a_3 = (f^2 - \hat{b}_3^2)/\hat{a}_3$ and $b_3 = -\hat{b}_3 \pmod{a_3}$.*
4. *Use Euclid's algorithm to compute (linear) polynomials h_1, h_3 such that $h_1 a_1 + h_3 a_3 = 1$.*
5. *Let $\hat{a}_4 = a_1 a_3$ and $\hat{b}_4 = b_1 + h_1 a_1 (b_3 - b_1) \pmod{\hat{a}_4}$.*
6. *Let $a_4 = (f^2 - \hat{b}_4^2)/\hat{a}_4$ and $b_4 = -\hat{b}_4 \pmod{a_4}$.*

The key observation in simplifying the algorithm is contained in the following lemma. We need one bit of notation: if a and b are polynomials, we let $a \pmod{b}$ denote the unique polynomial r of degree less than $\deg q$ such that $a = pb + r$ for some polynomial p. (I.e. $a \pmod{b}$ is the remainder when a is divided by b via the Euclidean algorithm.)

Lemma 1. *Let C be a curve of genus 2 defined by $y^2 = f(x)$. Let a_i, b_i, h_i, k_i be defined as in Algorithm 1 above. Then*

$$k_1(b_2 - b_1) \pmod{a_2} \quad = \quad -h_1(b_3 + b_1) \pmod{a_3} \tag{1}$$

[1]I am currently working on the degenerate cases.

Proof. Given our simplifying assumption that the a_i are quadratic and coprime, the polynomial $y - \hat{b}_3$ is the unique cubic polynomial through the four points defining P and Q. Since $f(x)$ is a polynomial of degree 5, this cubic intersects C in exactly two additional points; namely, the points that define $-P-Q$. Thus if we were to add P and $-P-Q$ via Algorithm 1, we would construct the same polynomial in step 2. Since $-P-Q$ is represented by $(a_3, -b_3)$, this implies that

$$\hat{b}_3 = b_1 - h_1 a_1 (b_3 + b_1) \pmod{a_1 a_3}, \tag{2}$$

and the identity (1) follows. □

At this point we take our cue from the elliptic curve case (cf. [ELM1, §3]) and set

$$\begin{aligned} \lambda_1 &= k_1(b_2 - b_1) \pmod{a_2} &&= -h_1(b_3 + b_1) \pmod{a_3}, \\ \lambda_2 &= h_1(b_3 - b_1) \pmod{a_3} &&= -\lambda_1 - 2h_1 b_1 \pmod{a_3}. \end{aligned}$$

(On an elliptic curve, the quantities λ_1 and λ_2 are slopes of the lines used in computing the (geometric) group law; here they can be thought of as a "slope" describing a cubic.) We now have $\hat{b}_3 = b_1 + \lambda_1 a_1$ and $\hat{b}_4 = b_1 + \lambda_2 a_1$, so there is no need to reference b_3 when computing b_4. Our new algorithm can be summarized as follows:

Algorithm 2 ([Cantor]). *Let C be a curve of genus 2 defined by $y^2 = f(x)$ as above, and $P, Q \in \mathrm{Jac}(C)$ be represented by (a_i, b_i) also as above. The following algorithm takes input (f, a_1, b_1, a_2, b_2) and computes (a_4, b_4) representing the point $2P + Q$.*

1. *Use Euclid's algorithm to compute (linear) polynomials k_1, k_2, h_1, h_3 such that $k_1 a_1 + k_2 a_2 = 1$ and $h_1 a_1 + h_3 a_3 = 1$.*
2. *Let $\lambda_1 = k_1(b_2 - b_1) \pmod{a_2}$ and $\lambda_2 = -\lambda_1 - 2h_1 b_1 \pmod{a_3}$.*
3. *Let $\hat{a}_3 = a_1 a_2$ and $\hat{b}_3 = b_1 + \lambda_1 a_1$.*
4. *Let $a_3 = (f^2 - \hat{b}_3^2)/\hat{a}_3$.*
5. *Let $\hat{a}_4 = a_1 a_3$ and $\hat{b}_4 = b_1 + \lambda_2 a_1$.*
6. *Let $a_4 = (f^2 - \hat{b}_4^2)/\hat{a}_4$ and $b_4 = -\hat{b}_4 \pmod{a_4}$.*

The correctness of Algorithm 2 follows from Lemma 1 and the correctness of Algorithm 1. Note that we still need to compute $\hat{b}_3$ since it is used in the computation of a_3. The total savings is thus one polynomial division; specifically, the computation of $b_3 = -\hat{b}_3 \pmod{a_3}$.

Proposition 2. *Given an implementation of Algorithm 1, let t be the maximum number of field multiplications executed in performing Algorithm 1, and let s be the number of field multiplications executed in computing b_3 from $\hat{b}_3$ and a_3. Then Algorithm 2 can be performed with at most $t - s$ field multiplications.*

Proof. It suffices to show that every multiplication performed in Algorithm 2 is also performed in Algorithm 1, with the exception of the computation of b_3. The only step where this is not obvious is step 5. In this step, Algorithm 2 computes $a_1(-\lambda_1 - 2h_1 b_1) \pmod{a_1 a_3}$ in the place where Algorithm 1 computes $a_1 h_1 (b_3 - b_1) \pmod{a_1 a_3}$. Thus if we store λ_1 when it is computed in step 2, these operations take the same number of field multiplications. □

In [ELM2, §4.5], the authors state that adding two distinct elements of $\mathrm{Jac}(C)$ costs 26 multiplications and 2 inversions, so forming $2P+Q$ via $(P+Q)+P$ as in Algorithm 1 costs 52 multiplications and 4 inversions. The computation of b_3, which our simplification omits, involves computing the remainder of the cubic polynomial $\hat{b}_3$ divided by the monic quadratic a_3. This computation requires four multiplications. If, as in [ELM1], we estimate a division as 5.18 multiplications, our total savings is 4 multiplications out of 72.7, or 5.5%.

3 Pairing algorithm

The algorithms for computing the Weil and Tate pairings make use of rational functions with prescribed poles and zeroes. For any nonzero point $T \in \mathrm{Jac}(C)$ we denote by (T) the unique effective divisor of degree 2 such that T is the class of $(T) - 2(\infty)$ in $\mathrm{Pic}^0(C)$ [CF, Ch. 1]. Now let $m > 0$ be an integer, and fix an nonzero m-torsion point R of $\mathrm{Jac}(C)$. For an integer $c > 0$, let f_c be a function such that

$$\operatorname{div} f_c = c(R) - (cR) - (2c-2)(\infty).$$

(The existence of such a function follows from the Riemann-Roch theorem.) For integers $b, c > 0$, let $g_{b,c}$ be the (unique) cubic of the form $y = p(x)$ passing through the four points defining bR and cR (tangent if there are points in common), and g_{b+c} be the quadratic in x with zeroes at the x-coordinates of the two points defining $(b+c)R$ (i.e. two vertical lines passing through $(b+c)R$ and $-(b+c)R$). Then we have the formula

$$f_{b+c} = f_b \cdot f_c \cdot \frac{g_{b,c}}{g_{b+c}}. \tag{3}$$

Note that to compute f_{b+c}, we must first compute the point $bR + cR$. To take advantage of our improved addition algorithm, we wish to find a way to evaluate the function f_{2b+c} without referencing the intermediate b_3 computed in step 3 of the standard algorithm to compute $bR + bR + cR$ (see above).

We begin by using (3) to write f_{2b+c} as

$$f_{2b+c} = \frac{f_b \cdot f_b \cdot f_c}{g_{2b+c}} \cdot \frac{g_{b,c} \cdot g_{b+c,b}}{g_{b+c}}$$

The term $g_{b,c} \cdot g_{b+c,b}/g_{b+c}$ is the product of two cubics divided by a quadratic. Its divisor is equal to $2(P) + (Q) + (-2P-Q) - 4(\infty)$. We wish to replace this function with a single quartic polynomial that has the same divisor.

Proposition 3. *Let C be a curve of genus 2 defined by $y^2 = f(x)$, R an m-torsion point, and $b, c \in \mathbb{Z}$. Let $P = bR$ be represented by (a_1, b_1), $Q = cR$ be represented by (a_2, b_2). Then there is a polynomial $q(x,y)$ of the form $q(x,y) = r(x) + y \cdot s(x)$, where $\deg r = 4$ and $\deg s = 1$, such that*

$$\operatorname{div} q = 2(P) + (Q) + (-2P-Q) - 4(\infty)$$

Furthermore, q can be computed from the intermediates of Algorithm 1 without performing any divisions.

Remark. For simplicity, we make the same assumptions as in Section 2; namely, that the a_i are quadratic and pairwise coprime, and none of the points is equal to the identity.

Proof. Let $P+Q$ be represented by (a_3, b_3). From equation (2), step 5 of Algorithm 1, and the identity $h_1a_1 + h_3a_3 = 1$, we find that

$$\begin{aligned}\hat{b}_3 &= -b_3 + h_3a_3(b_1+b_3) \pmod{a_1a_3}\\ \hat{b}_4 &= b_3 + h_3a_3(b_1-b_3) \pmod{a_1a_3}.\end{aligned}$$

These are cubic polynomials through the points defining $P, Q, -P-Q$ and $P, P+Q, -2P-Q$, respectively. Let $\mu_1 = h_3(b_1+b_3) \pmod{a_1}$ and $\mu_2 = h_3(b_1-b_3) \pmod{a_1}$. Then the following is a function with divisor equal to $2(P)+(Q)+(-2P-Q)-4(\infty)$:

$$\frac{\hat{b}_3\cdot\hat{b}_4}{a_3} = \frac{(y+b_3-\mu_1a_3)(y-b_3-\mu_2a_3)}{a_3}.$$

Expanding in powers of a_3, this is equal to

$$\frac{y^2-b_3^2}{a_3} - y(\mu_2+\mu_1) + b_3(\mu_1-\mu_2) + \mu_1\mu_2a_3. \tag{4}$$

Since the zeroes x_i of a_3 give two points $(x_i, b_3(x_i))$ on the curve C, the first term of (4) is a cubic polynomial, and it follows that (4) is a polynomial of the form specified in the Proposition.

Now let $a_i = x^2+\alpha_ix+\beta_i$. If we assume that $f(x)$ is a monic quintic with no degree 4 term (which we may do over any field of characteristic not equal to 2 by making a linear change of variables), then we have

$$\frac{y^2-b_3^2}{a_3} = x^3+\alpha_3x^2 + \text{linear term}.$$

(Here and throughout this section, by "linear term" we mean a line c_1x+c_0.) Our quartic is thus

$$q(x,y) = x^3+\alpha_3x^2+b_3(\mu_1-\mu_2)+\mu_1\mu_2a_3 - y(\mu_1+\mu_2) + \text{linear term in } x.$$

To specify the equation for $q(x,y)$ more precisely, write

$$\mu_1\mu_2a_3 = c_4x^4+c_3x^3+c_2x^2+\text{linear term in } x.$$

Let γ be the leading coefficient of $b_1(\mu_1+\mu_2)$ and δ be the leading coefficient of $b_3(\mu_1-\mu_2)$. Then we have

$$\begin{aligned}q'(x,y) &= a_1(x-\alpha_1+\alpha_3-\gamma+\delta+c_4(x^2-\alpha_1x+\alpha_1^2-\beta_1)+c_3(x-\alpha_1)+c_2) \\ &\quad -(y-b_1)(\mu_1+\mu_2) \\ &= (x^3+\alpha_3x^2)+\gamma x^2+(c_4x^4+c_3x^3+c_2x^2)-y(\mu_1+\mu_2)+\text{linear term in } x \\ &= q(x,y)+\text{linear term in } x.\end{aligned} \tag{5}$$

(Recall that $a_1 = x^2+\alpha_1x+\beta_1$.) Since $q(x,y)$ and $q'(x,y)$ are equal up to a linear term in x, and they are both equal to zero at the two points defining P, we conclude they are equal. Thus (5) gives a formula for $q(x,y)$ that can be computed from the intermediates of Algorithm 1 without performing any divisions. □

Analysis of savings

First we consider the cost of evaluating the function $g_{b,c} \cdot g_{b+c,b}/g_{b+c}$ in the standard algorithm for computing the Weil or Tate pairing. As above, we let $P = bR = (a_1, b_1)$ and $Q = cR = (a_2, b_2)$, and follow the notation of Section 2. Then $g_{b,c} = y - \hat{b}_3$ is the cubic passing through P, Q, and $-P-Q$; $g_{2b,c} = y - \hat{b}_4$ is the cubic passing through P, $P+Q$, and $-2P-Q$; and $g_{b+c} = a_3$ is the quadratic with zeroes at the x-coordinates of $P+Q$. We thus have

$$\frac{g_{b,c} \cdot g_{b+c,b}}{g_{b+c}} = \frac{(y - b_1 - \lambda_1 a_1)(y - b_3 - \lambda_2 a_3)}{a_3}. \tag{6}$$

This formula contains three multiplications and one division. Evaluating each of the polynomials a_i (monic quadratic), b_i (linear) and λ_i (linear) takes one multiplication, so evaluating the entire formula requires 10 multiplications and one division.

Next, we wish to use the expression (5) to evaluate $q(x, y)$ at a point (x_0, y_0). This process requires two steps: first constructing the polynomial, and then evaluating it.

Constructing the polynomial requires computation of the linear polynomials μ_i and the constants c_i, γ, and δ. Recall that

$$\hat{b}_3 = b_1 + \lambda_1 a_1 = -b_3 + \mu_1 a_3.$$

In the elliptic curve case λ_1, μ_1, and the b_i are all constants, and since the a_i are monic we can conclude immediately that $\lambda_1 = \mu_1$. In the genus 2 case we find that the leading coefficients of λ_1 and μ_1 are equal, but must do some work to obtain the constant term. Recall that $a_i = x^2 + \alpha_i x + \beta_i$, and let $\lambda_1 = l_1 x + l_0$, and $\mu_1 = m_1 x + m_0$. Then $l_1 = m_1$, and since the x^3 and x^2 terms of $\lambda_1 a_1$ and $\mu_1 a_3$ are equal, we conclude that

$$\mu_1 = \lambda_1 + l_1(\alpha_1 - \alpha_3).$$

Thus we can construct μ_1 from λ_1, a_1, and a_3 with one multiplication. Similarly, since $\lambda_2 a_1$ and $\mu_2 a_3$ have the same cubic and quadratic terms, we can construct μ_2 from λ_2, a_1, and a_3 with one multiplication.

As for the constants, γ and δ are both leading coefficient of products of two linear polynomials, so they require one multiplication each. Finally, c_4, c_3, c_2 are the first three coefficients of the quartic $\mu_1 \mu_2 a_3$, and they require one, three, and five multiplications respectively. We conclude that construction of $q(x, y)$ requires 13 multiplications.

Evaluating $q(x, y)$ at a point requires evaluation of three polynomials: a_3 (monic quadratic), b_1 (linear), and $\mu_1 + \mu_2$ (linear). Evaluation of these polynomials takes one multiplication each. The factor $x^2 - \alpha_1 x + \alpha_1^2 - \beta_1 = x(x - \alpha_1) + \alpha_1^2 - \beta_1$ may be evaluated with two multiplications, and the remainder of the formula requires four multiplications. Thus evaluating the polynomial requires nine multiplications.

Conclusion

Given the a_i, b_i, and λ_i, the standard formula (6) requires no multiplications to construct and 10 multiplications and one division to evaluate. On the other hand, our quartic $q(x, y)$ takes 13 multiplications to construct and 9 multiplications to evaluate. Estimating a division as 5.18

multiplications as in [ELM1], we see that evaluating the polynomial three or more times leads to an overall improvement. In particular, if one were to evaluate (say) the Tate pairing $\phi(P, Q)$ for fixed P and varying Q, using our polynomial $q(x, y)$ would save the equivalent of 6 multiplications per evaluation.

Finally, we note that the constant δ is the only part of the formula which references b_3. (It is defined as the leading coefficient of $b_3(\mu_1 - \mu_2)$.) Thus if a way were found to evaluate δ without using b_3, we would be able to apply the speedup of section 2 to the evaluation of the pairing. Unfortunately, we have not yet figured out a way to circumvent this use of b_3.

References

[Cantor] Cantor, David G., "Computing in the Jacobian of a hyperelliptic curve," *Math. Comp.* **48** (1987), 95-101.

[CF] Cassels, J.W.S., and E.V. Flynn, *Prolegomena to a middlebrow arithmetic of curves of genus 2,* Cambridge University Press, Cambridge 1996.

[ELM1] Eisenträger, Kirsten, Kristin Lauter, and Peter Montgomery, "Fast elliptic curve arithmetic and improved Weil pairing evaluation," in *Topics in Cryptology: CT-RSA 2003,* Ed. Marc Joye, Springer LNCS **2612**, Berlin 2004, 343-354.

[ELM2] Eisenträger, Kirsten, Kristin Lauter, and Peter Montgomery, "Improved Weil and Tate pairings for elliptic and hyperelliptic curves," in *Algorithmic Number Theory: ANTS-VI*, Ed. Duncan Buell, Springer LNCS **3076**, Berlin 2004, 169-183.

[L] Lauter, Kristin, "The equivalence of the geometric and algebraic group laws for Jacobians of genus 2 curves," *Topics in Algebraic and Noncommutative Geometry*, AMS Contemporary Mathematics Series **324** (2003) 165-171.