Explicit formulas for real hyperelliptic curves of genus 2 in affine representation

Stefan Erickson[1], Michael J. Jacobson, Jr.[2], Ning Shang[3], Shuo Shen[3], and Andreas Stein[4]

[1] Department of Mathematics and Computer Science, Colorado College, 14 E. Cache La Poudre, Colorado Spgs., CO. 80903, USA,
`Stefan.Erickson@ColoradoCollege.edu`

[2] Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta, Canada, T2N 1N4
`jacobs@cpsc.ucalgary.ca`

[3] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, USA
`nshang@math.purdue.edu`, `sshen@math.purdue.edu`

[4] Department of Mathematics, University of Wyoming 1000 E. University Avenue, Laramie, WY 82071-3036, USA
`astein@uwyo.edu`

Abstract. In this paper, we present for the first time efficient explicit formulas for arithmetic in the degree 0 divisor class group of a real hyperelliptic curve. Hereby, we consider real hyperelliptic curves of genus 2 given in affine coordinates for which the underlying finite field has characteristic > 3. These formulas are much faster than the optimized generic algorithms for real hyperelliptic curves and the cryptographic protocols in the real setting perform almost as well as those in the imaginary case. We provide the idea for the improvements and the correctness together with a comprehensive analysis of the number of field operations. Finally, we perform a direct comparison of cryptographic protocols using explicit formulas for real hyperelliptic curves with the corresponding protocols presented in the imaginary model.

Keywords: hyperelliptic curve, reduced divisor, infrastructure and distance, Cantor's algorithm, explicit formulas, efficient implementation, cryptographic key exchange

1 Introduction and Motivation

In 1989, Koblitz [9] first proposed the Jacobian of an imaginary hyperelliptic curve for use in public-key cryptographic protocols. Hyperelliptic curves are in a sense generalizations of elliptic curves, which are an attractive option for public-key cryptography because their key-per-bit security is significantly better than RSA. This is due to the fact that the best-known attacks on elliptic curve based cryptosystems have exponential as opposed to subexponential complexity in the bit length of the key. Hyperelliptic curves can be used with the same key-per-bit strength as elliptic curves provided that the genus is very small. In particular, recent attacks [4, 5], imply that only genus 2 and possibly genus 3 hyperelliptic curves offer the same key-per-bit security as elliptic curves.

The Jacobian is a finite abelian group which, like elliptic curve groups, has unique representatives of group elements and efficient arithmetic (divisor addition and reduction). Although the arithmetic appears more complicated than that of elliptic curves [10, 16, 21, 1, 6], there are some indications that it can in some cases be more efficient. Those results are based on optimized explicit formulas and very efficient implementations for genus 2 and 3 imaginary hyperelliptic curves.

Several years later, a key exchange protocol was presented for the real model of a hyperelliptic curve [18]. Its underlying key space was the set of reduced principal ideals in the ring of regular functions of the curve, together with its group-like infrastructure. Although the main operation of divisor class addition, which is composition followed by reduction, is comparable in efficiency to that of the imaginary model [20], the protocol in [18] was significantly slower and more complicated than its imaginary cousin [9], while offering no additional security; the same was true for subsequent modifications presented in [17].

Despite the apparent short-comings of the real model, recent work [7] shows that real hyperelliptic curves may admit protocols that are comparable in efficiency to those based on the imaginary model. The main idea is that, in addition to the divisor class addition operation, the real model has a second operation called a *baby step* that is significantly more efficient. By exploiting this operation and some reasonable heuristics, new public-key protocols for key exchange, digital signatures, and encryption have been devised that are significantly faster than all previous protocols in real hyperelliptic curves and might even be comparable in efficiency with analogous protocols in the imaginary setting. However, the protocols in [7] were based on a generic implementation and did not incorporate explicit formulas. In order to examine the efficiency of these new protocols completely, it is necessary to devise explicit formulas for divisor arithmetic in the real model of cryptographically-relevant low genus curves.

The contribution of this paper is to close this gap and for the first time present efficient explicit formulas for divisor class arithmetic on real hyperelliptic curves. We concentrate on genus 2 real hyperelliptic curves in affine coordinates for which the underlying finite field has characteristic $p > 3$. Formulas for arbitrary characteristic, that also handle all special cases, will be included in the full version of this paper, which will be submitted to a journal. We thus provide explicit formulas for the protocols in [7], thereby enabling a direct comparison with the corresponding protocols presented in the imaginary model.

Notice that although there exist easy transformations from the imaginary model to the real model of a hyperelliptic curve, the converse direction is only possible if the curve defined over $\mathbb{F}_q$ contains an $\mathbb{F}_q$-rational point. If q is odd and one uses an irreducible polynomial for the generation of the real hyperelliptic curve, one has to extend the field of constants to $\mathbb{F}_{q^{2g+2}}$ in order to be able to perform this transformation, which is unrealistic for efficient implementations. Furthermore, complex multiplication methods for generating hyperelliptic curves of small genus often produce real hyperelliptic curves. With an efficient arithmetic, those curves can be readily used in cryptographic protocols. Finally, explicit formulas enable us to provide a real-world comparison of subexponential attacks to hyperelliptic curve cryptosystems in both the real and imaginary setting.

The analysis of the formulas presented here shows that they require a few more finite field multiplications than their imaginary counterparts. However, the baby step operation in its explicit form is significantly more efficient than divisor class addition in either setting, and as a result, the cryptographic protocols in the real setting perform almost as well as those in the imaginary case. In addition, even though the formulas are not as fast as those in the imaginary case, they are certainly more efficient than using generic algorithms. Thus, using our formulas will significantly speed other computations in the divisor class group or infrastructure of a real hyperelliptic curve, for example, computing the regulator or class number.

The paper is organized as follows. We first provide the necessary background on real hyperelliptic curves and introduce the notation. We will also present the essential, generic algorithms for real hyperelliptic curves and explain how to perform arithmetic in the degree 0 divisor class group via ideal arithmetic. In Section 3, we present the explicit formulas for the basic algorithms assuming a finite field of characteristic $p > 3$. We provide the idea for the improvements and the correctness together with a comprehensive analysis of the number of field operations. Some of the calculations can also be found in the Appendix. Section 4 contains numerical data comparing cryptographic protocols based on real hyperelliptic curves with those using imaginary hyperelliptic curves, where divisor class arithmetic is implemented using explicit formulas in both cases.

2 Background and Notation

Throughout this paper, let $\mathbb{F}_q$ be a finite field with $q = p^l$ elements, where p is a prime, and let $\overline{\mathbb{F}}_q = \bigcup_{n \geq 1} \mathbb{F}_{q^n}$ be its algebraic closure. For details on the arithmetic of hyperelliptic curves we refer to [11, 6, 2, 7], and specifically for real hyperelliptic curves we refer to [15, 20, 3, 7, 8].

Definition 1. *A hyperelliptic curve C of genus g defined over $\mathbb{F}_q$ is an absolutely irreducible non-singular curve defined by an equation of the form*

$$C : y^2 + h(x)y = f(x), \tag{2.1}$$

where $f, h \in \mathbb{F}_q[x]$ are such that $y^2 + h(x)y - f(x)$ is absolutely irreducible, i.e. irreducible over $\overline{\mathbb{F}}_q$, and if $b^2 + h(a)b = f(a)$ for $(a, b) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q$, then $2b + h(a) \neq 0$ or $h'(a)b - f'(a) \neq 0$. A hyperelliptic curve C is called

1. *an* imaginary hyperelliptic curve *if the following hold: If q is odd, then f is monic, $\deg(f) = 2g + 1$, and $h = 0$. If q is even, then h and f are monic, $\deg(f) = 2g + 1$, and $\deg(h) \leq g$.*

2. *a* real hyperelliptic curve *if the following hold: If q is odd, then f is monic,* $\deg(f) = 2g+2$*, and* $h = 0$*. If q is even, then h is monic,* $\deg h = g+1$*, and either (a)* $\deg f \leq 2g+1$ *or (b)* $\deg f = 2g+2$ *and the leading coefficient of f is of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q^*$.*

3. *an* unusual hyperelliptic curve *[3] if the following holds: if $\mathbb{F}_q$ has odd characteristic, then* $\deg(f) = 2g+2$ *and* $\operatorname{sgn}(f)$ *is a non-square in $\mathbb{F}_q^*$, whereas if $\mathbb{F}_q$ has characteristic 2, then* $\deg(h) = g+1$, $\deg(f) = 2g+2$ *and the leading coefficient of f is not of the form $e^2 + e$ for any $e \in \mathbb{F}_q^*$.*

The function field $K = \mathbb{F}_q(C)$ of a hyperelliptic curve C is a quadratic, separable extension of $\mathbb{F}_q(x)$ and the integral closure of $\mathbb{F}_q(x)$ in K is given by $\mathbb{F}_q[C] = \mathbb{F}_q[x,y]/(y^2 + h(x)y - f(x))$. Let S denote the set of points at infinity. Then the set $C(\overline{\mathbb{F}}_q) = \{(a,b) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q : b^2 + h(a)b = f(a)\} \cup S$ is called the set of ($\overline{\mathbb{F}}_q$-rational) points on C. For a point $P = (a,b) \in C(\overline{\mathbb{F}}_q)$, the hyperelliptic involution is given by $\iota(a,b) = (a, -b - h(a)) \in C(\overline{\mathbb{F}}_q)$.

Notice that in all three cases we can assume $h = 0$ if q is odd. The imaginary model [5] corresponds to the case where $S = \{\infty_1\}$. In the real model[6], there exist two points at infinity so that $S = \{\infty_1, \infty_2\}$. Let v_1 and v_2 be the normalized valuations of K at ∞_1 and ∞_2, respectively. It is possible to transform an imaginary model of a hyperelliptic curve into a real model. For the converse direction one needs an $\mathbb{F}_q$-rational point (see [15, 6]). From now on, we only consider the real case.

Let C be a real hyperelliptic curve given as in Definition 1. A divisor on C is a finite formal sum $D = \sum_{P \in C} m_P P$ of points $P \in C(\overline{\mathbb{F}}_q)$, where $m_P \in \mathbb{Z}$ and $m_P = 0$ for almost all P. The degree of D is defined by $\deg D = \sum_P m_P$. A divisor D of $\mathbb{F}_q(C)$ is effective if $m_P \geq 0$ for all P, and a divisor D is defined over $\mathbb{F}_q$, if $D^\sigma = \sum_P m_P P^\sigma = D$ for all automorphisms σ of $\overline{\mathbb{F}}_q$ over $\mathbb{F}_q$. The set $Div(K)$ of divisors of C defined over $\mathbb{F}_q$ forms an additive abelian group under formal addition with the set $Div_0(K)$ of all degree zero divisors of C defined over $\mathbb{F}_q$ as a subgroup. For a function $G \in K$, we can associate a principal divisor $\operatorname{div}(G) = \sum_P v_P(G)P$, where $v_P(G)$ is the normalized valuation of G at P. The group of principal divisors $P(K) = \{\operatorname{div}(G) : G \in K\}$ of C forms a subgroup of $Div_0(K)$. The factor group $J(K) = Div_0(K)/P(K)$ is called the *divisor class group* of K. We denote by $\overline{D} \in J(K)$ the class of $D \in Div_0(K)$.

Since C is a real hyperelliptic curve we have $S = \{\infty_1, \infty_2\}$ and we know from [15] that every degree 0 divisor class can be represented by $\overline{D}$ such that $D = \sum_{i=1}^r P_i - r\infty_2$, where $P_i \in C(\overline{\mathbb{F}}_q)$, $P_i \neq \infty_2$, and $P_i \neq \iota P_j$ if $i \neq j$. The representative D of $\overline{D}$ is then called semi-reduced. In addition, there exists a representative D such that $r \leq g$. In this case, the representative D is called reduced. Notice that $P_i = \infty_1$ is allowed for some i. It follows that every degree

[5] In function field terms, the pole divisor ∞ of x in $\mathbb{F}_q(x)$ is totally ramified in K so that $\operatorname{Con}(\infty) = 2\infty_1$.

[6] In function field terms, the pole divisor ∞ of x in $\mathbb{F}_q(x)$ splits completely in K so that $\operatorname{Con}(\infty) = \infty_1 + \infty_2$.

0 divisor class contains a unique representative $\overline{D}$ with

$$D = \sum_{i=1}^{l(D)} Q_i - l(D)\infty_2 + v_1(D)(\infty_1 - \infty_2) \ ,$$

where $Q_i \in C(\overline{\mathbb{F}}_q)$, $Q_i \neq \infty_1, \infty_2$, $Q_i \neq \iota Q_j$ if $i \neq j$, and $0 \leq l(D) + v_1(D) \leq g$. The regulator R of K in $\mathbb{F}_q[C]$ is defined to be the order of the degree 0 divisor class containing $\infty_1 - \infty_2$.

We know that $\mathbb{F}_q[C]$ is a Dedekind domain and the ideal class group $\mathrm{Cl}(K)$ of K in $\mathbb{F}_q[C]$ is the factor group of fractional $\mathbb{F}_q[C]$-ideals modulo principal fractional ideals. A non-zero integral ideal $\mathfrak{a}$ in $\mathbb{F}_q[C]$ is a fractional ideal such that $\mathfrak{a} \subseteq \mathbb{F}_q[C]$. It can be represented as $\mathfrak{a} = k[x]\, d(x)u(x) + k[x]\, d(x)(v(x) + y)$ where , $u, v \in k[x]$ and $u \mid f + hv - v^2$. Note that d and u are unique up to factors in $\mathbb{F}_q^*$ and v is unique modulo u. $\mathfrak{a}$ is primitive if we can take $d(x) = 1$ in which case we simply write $\mathfrak{a} = [u(x), v(x) + y]$. A primitive ideal $\mathfrak{a} = [u(x), v(x) + y]$ is *reduced* if $\deg u \leq g$. A basis $\{u(x), v(x) + y\}$ of a primitive ideal is called *adapted* or *standard* if $\deg(v) < \deg(u)$ and u is monic. For instance, $\mathbb{F}_q[C]$ is represented as $\mathbb{F}_q[C] = [1, y]$. The degree of a primitive ideal is $\deg(\mathfrak{a}) = \deg u$. We call a basis $\{u(x), v(x) + y\}$ of a primitive ideal *reduced* if $-v_1(v - h - y) < -v_1(u) = \deg(u) < -v_1(v + y)$ and u is monic.

For any two ideals $\mathfrak{a}$ and $\mathfrak{b}$ in the same ideal class, there exists $\alpha \in K^*$ with $\mathfrak{b} = (\alpha)\mathfrak{a}$. We then define the distance of $\mathfrak{b}$ with respect to $\mathfrak{a}$ as $\delta(\mathfrak{b}, \mathfrak{a}) = -v_1(\alpha) \pmod{R}$ where R is the regulator. Note that the distance is only well-defined and unique modulo R. In each ideal class, we expect up to R many reduced ideals. If we fix the principal ideal class, then we may assume that $\mathfrak{a} = \mathfrak{a}_1 = \mathbb{F}_q[C] = (1)$. Then, for any principal ideal $\mathfrak{b} = (\alpha)$, we have $\delta(\mathfrak{b}) = \delta(\mathfrak{b}, \mathfrak{a}_1) = -v_1(\alpha) \pmod{R}$. Notice that the distance defines an order on all reduced principal ideals, i.e. the set of reduced principal ideals is $\mathcal{R} = \{\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_m\}$ where $\delta(\mathfrak{a}_1) = 0 < \delta(\mathfrak{a}_2) < \ldots < \delta(\mathfrak{a}_m)$.

The following theorem gives a representation of degree 0 divisor classes in terms of reduced ideals and corresponds to the Mumford representation [13, page 317] in the imaginary model.

Theorem 1. *(Paulus-Rück, 1999) There is a canonical bijection between the divisor class group $J(K)$ and the set of pairs $\{(\mathfrak{a}, n)\}$, where $\mathfrak{a}$ is a reduced ideal of $\mathbb{F}_q[C]$ and n is a non-negative integer with $0 \leq \deg(\mathfrak{a}) + n \leq g$.*

The bijection is such that the unique reduced divisor D in a degree 0 divisor class $\overline{D}$ corresponds to such a pair $\{(\mathfrak{a}, n)\}$. It follows that arithmetic in $J(K)$ can be performed via arithmetic of reduced ideals. An algorithm for computing the group law in $J(K)$ based on this theorem has been presented in [18, 15, 20]. It consists of three steps, namely (a) composition of reduced ideals, (b) reduction of the primitive part of the product, and (c) baby steps, i.e. adjusting the output of the reduction so that the degree condition of the theorem is satisfied. Step (a) and (b) together are called a *giant step*. A giant step is the analogue of the group operation in the imaginary case. We use $[u_1, v_1] + [u_2, v_2]$

to denote the giant step operation. Elements in $J(K)$ can represented as triples $[u, v, n]$ where $[u(x), y + v(x)]$ is a reduced ideal and $0 \leq \deg(\mathfrak{a}) + n \leq g$. It can be easily seen that the arithmetic can be restricted to the special subset $\{(\mathfrak{a}, 0) : \mathfrak{a} \text{ reduced and principal}\} = \mathcal{R}$, which is not a group. We may restrict our arithmetic to the degree 0 divisor classes that correspond to the set $\mathcal{R}$. Those elements can be represented as $[u, v, 0]$ or simply as pairs $[u, v]$. We therefore assume that we only perform operations on elements of $J(K)$ which are given by a pair $\overline{D} = [u, v]$, where $u, v \in \mathbb{F}_q[x]$ such that

1. u is monic,
2. $\deg(u) \leq g$,
3. $u \mid f + hv - v^2$,
4. one of the following degree conditions is satisfied, namely
   (a) for the reduced basis: $-v_1(v - h - y) < -v_1(u) = \deg(u) < -v_1(v + y)$, or
   (b) for the adapted (standard) basis: $\deg(v) < \deg(u)$.

If only 1., 3., and 4. are satisfied, the ideal $[u(x), y + v(x)]$ is only primitive and the corresponding representative $D \in \overline{D}$ is semi-reduced. We also denote this element by $[u, v]$.

In [7], several optimized key-exchange protocols have been presented that use arithmetic in $\mathcal{R}$. In fact, under reasonable assumptions, one can avoid the additional adjusting steps and replace some giant steps by baby steps. Furthermore, in each giant step, it is easy to keep track of the distances of the corresponding reduced ideals. In fact, assuming certain heuristics, one can even avoid computing distances in almost all situations. We will therefore ignore the computation of distances. Even in those cases, where distances are needed, the running time for the computation of the distance is negligible. The protocols for real hyperelliptic curves are analogous to the ones in the imaginary setting, but they also make use of the additional baby step operation.

We now give all three relevant algorithms. For details on how to produce key exchange protocols with these algorithms, we refer to [18, 7]. We will use additive notation in order to express the group operation in $J(K)$ even though ideal arithmetic is usually denoted multiplicatively. Note that, by using these algorithms, arithmetic in $J(K)$ is reduced to polynomial arithmetic in $\mathbb{F}_q[x]$.

Algorithm 1 (Composition).
Input: $\overline{D}_1 = [u_1, v_1]$, $\overline{D}_2 = [u_2, v_2]$, and $h(x), f(x)$ as in (2.1).
Output: $\overline{D} = [u, v]$ such that D is semi-reduced and $\overline{D} = \overline{D}_1 + \overline{D}_2$.

1. Compute $d, x_1, x_2, x_3 \in \mathbb{F}_q[x]$ such that
$$d = \gcd(u_1, u_2, v_1 + v_2 + h) = x_1u_1 + x_2u_2 + x_3(v_1 + v_2 + h) \ .$$
2. Put $u = u_1u_2/d^2$ and $v = (x_1u_1v_2 + x_2u_2v_1 + x_3(v_1v_2 + f))/d \pmod{u}$.

For the group operation, we assume that the representatives of the degree 0 divisor classes D_1 and D_2 are reduced so that the ideals $[u_1(x), y + v_1(x)]$ and $[u_2(x), y + v_2(x)]$ are reduced, i.e. $\deg(u_1), \deg(u_2) \le g$. However, the algorithm also allows semi-reduced representatives D_1 and D_2 as an input. Notice that the output of this algorithm $\overline{D} = [u, v]$ corresponds to a semi-reduced divisor so that $[u(x), v(x) + y]$ is a primitive ideal which is not necessarily reduced.

For the second step, we need to precompute the principal part $H(y) = \lfloor y \rfloor$ of a root y of $y^2 + h(x)y - f(x) = 0$. The other root is $-y - h$. If $y = \sum_{i=-\infty}^{m} y_i x^i \in \mathbb{F}_q\langle x^{-1}\rangle$, then $H(y) = \sum_{i=0}^{m} y_i x^i$.

Algorithm 2 (Reduction).
Input: $\overline{D} = [u, v]$, where D is semi-reduced, and $h(x), f(x)$ as in (2.1).
Output: $\overline{D}' = [u', v']$ such that D' is reduced and $\overline{D}' = \overline{D}$.

```
1. Compute $a = (v + H(y)) \operatorname{div} u$.
2. Let $v' = v - au$, $u' = (f + hv' - v'^2)/u$.
3. If $\deg(u') > g$, put $u = u'$, $v = v'$, and goto 1.
4. Make $u'$ monic and adjust $v'$ to a reduced/adapted basis if necessary.
```

If we allow the input of Algorithm 2 to be reduced and only perform 1,2, and 4, the output will be another reduced divisor D' representing a different degree 0 divisor class $\overline{D}'$. In this case, we call this operation a *baby step* [7] denoted by $[u', v'] = \rho[u, v]$.

3 Explicit Formulas

Let $[u, v]$ be a Mumford representation of a degree 0 divisor class. We present explicit formulas for divisor class addition (ideal multiplication), divisor class doubling (ideal squaring), and a baby step. We will assume that characteristic of the field is a prime $p > 3$. Under this assumption, we can transform the general equation defining the curve to one of the form

$$C : y^2 = f(x)$$

that is isomorphic to the original curve, where $f(x) = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$, i.e., we can assume that $h(x) = 0$, the leading coefficient of $f(x)$ is 1, and the x^5 term of $f(x)$ is 0. The transformation $y \mapsto y - h/2$, valid if the finite field characteristic is not 2, eliminates $h(x)$ and $x \mapsto x - f_5/6$, valid if the characteristic is not 2 or 3, eliminates the x^5 term in $f(x)$. This assumption also implies that $H(y) = x^3 + y_1x + y_0$, with $y_1 = f_4/2$ and $y_0 = f_3/2$.

We also assume that the divisor $[u, v]$ is in reduced basis. Under our assumptions about C, this implies that v is of the form

$$v = x^3 + v_1x + v_0,$$

[7] However, notice that in this case the reduced ideal $\mathfrak{a}$ corresponding to $\overline{D}$ and the reduced ideal $\mathfrak{a}'$ corresponding to $\overline{D}'$ are in the same ideal class.

i.e., the leading two coefficients of v always match that of $H(y)$. We will present formulas for the general description of a genus two real hyperelliptic curve in affine presentation over an arbitrary finite field, using both reduced and adapted basis, in the full version of this paper.

We only count inversions, squarings and multiplications of finite field elements, which consist of the bulk of the computation when compared with additions and subtractions. In the tables below, we let I, S and M denote "inversion," "squaring," and "multiplication," respectively.

In the formulas described below, we assume that the coefficients of $f(x) = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$ and $H(y) = x^3 + y_1x + y_0$ that define the hyperelliptic curve are available. Thus, these are not explicitly listed as input.

3.1 Baby Step

Let $[u, v]$ be the Mumford representation of a degree 0 divisor class. To compute $\rho[u, v] = [u', v']$, we apply the following formulas:

$$v' = H(y) - [(H(y) + v) \bmod u]\,,$$
$$u' = \text{Monic}\left(\frac{f - (v')^2}{u}\right)$$

were, as mentioned above, $H(y)$ is the principal part of a root of a root y of $y^2 + h(x)y - f(x) = 0$. Explicit formulas are derived by simply expanding the operations and using the formula for reducing a degree three polynomial $(H(y)+v)$ modulo a monic polynomial of degree two (u) described in [10]. The resulting formulas are presented in Table 1.

Table 1. Explicit Formulas for a Baby Step

Baby Step, Reduced Basis, $\deg u = 2$		
Input	$u = x^2 + u_1x + u_0, \quad v = x^3 + v_1x + v_0$	
Output	$[u', v'] = \rho[u, v]$	
Step	Expression	Operations
1	$\underline{v' = H(y) - [(H(y) + v) \bmod u]}$ $v'_1 = 2(u_0 - u_1^2) - v_1$ $v'_0 = -2u_0 \cdot u_1 - v_0$	1S, 1M
2	$\underline{u' = \text{Monic}((f - (v')^2)/u)}$ $u_2 = f_4 - 2v'_1$ $I = u_2^{-1}$ $u'_1 = I \cdot (f_3 - 2v'_0 - u_1)$ $u'_0 = I \cdot \left(f_2 - (v'_1)^2\right) - u_0 - u'_1 \cdot u_1$	1I, 1S, 3M
Total		1I, 2S, 4M

3.2 Addition Formulas

Let the Mumford representations of two degree 0 divisor classes be $[u_1, v_1]$ and $[u_2, v_2]$. The main case of addition of degree 0 divisor classes occurs when the two degree 0 divisor classes consist of four points on the curve which are different from each other and their opposites. This situation occurs precisely when $\deg(u_1) = \deg(u_2) = 2$ and u_1, u_2 are relatively prime. In the rare cases when u_1 or u_2 has degree less than 2, or when u_1 and u_2 are not relatively prime, the costs are considerably less than the general case. Here, we present addition for the general case; the special cases will be presented in the full version of the paper.

To optimize the computations, we do not follow Cantor's algorithm literally; we proceed instead as described in [10]. Given two degree 0 divisor classes $[u_1, v_1]$ and $[u_2, v_2]$, the algorithm for divisor addition $[u', v'] = [u_1, v_1] + [u_2, v_2]$ is found by calculating the following subexpressions.

$$\begin{array}{rlrl}
r &= \text{resultant of } u_1,\ u_2 & inv &\equiv r(u_2)^{-1} \pmod{u_1} \\
s' &\equiv (v_1 - v_2) \cdot inv \pmod{u_1} & s &= \frac{1}{r} \cdot s' \\
k &= \frac{f - v_2^2}{u_2} & l &= s \cdot u_2 \\
m &= k - s \cdot (l + 2v_2) & m' &= m/m_4 = m \text{ made monic} \\
u' &= m'/u_1 & v' &= H(y) - [(H(y) + v_2 + l) \pmod{u'}]
\end{array}$$

The explicit formulas are presented in Table 2.

1. Step 1 and Step 2 calculate the coefficients of $s = s_1 x + s_0 = (v_1 - v_2) \cdot (u_2)^{-1} \bmod u_1$. Instead of calculating s_1 and s_0, we calculate $s'_1 = r \cdot s_1$ and $s'_0 = r \cdot s_0$, thereby postponing the inversion until Step 4.
2. If $s'_1 = 0$ in Step 2, one needs to modify Step 4 through Step 7. In this special case, the sum of the two divisors will be a degree one divisor. As this case only occurs very rarely, we do not describe the required modifications here, rather, they will appear in the full version of the paper.
3. The composition of divisors in Step 1 and Step 2 is exactly the same for both real and imaginary cases. These two steps can replace the first three steps of imaginary divisor addition found in [10] for a savings of one squaring. For reference, this improvement makes 1I, 2S, 22M the least known number of field operations needed for divisor addition in the imaginary case.

3.3 Doubling Formulas

Let $[u, v] = [x^2 + u_1 x + u_0, x^3 + v_1 x + v_0]$ be a degree two divisor in reduced basis with both points of the divisor not equal to their opposites. Again following [10], we compute the degree 0 divisor class $[u', v'] := [u, v] + [u, v]$ as follows.

$$\begin{array}{rlrl}
\widetilde{v} &\equiv 2v \pmod{u} & inv &\equiv r(\widetilde{v})^{-1} \pmod{u} \\
k &= \frac{f - v^2}{u} & s' &\equiv k \cdot inv \pmod{u} \\
s &= \frac{1}{r} \cdot s' & \widetilde{u} &= s^2 + \frac{2vs - k}{u} \\
u' &= \widetilde{u} \text{ made monic} & v' &= H(y) - [(H(y) + s \cdot u + v) \pmod{u'}]
\end{array}$$

Table 2. Explicit Formulas for Addition of Divisor Classes

Addition, Reduced Basis, $\deg u_1 = \deg u_2 = 2$, $\gcd(u_1, u_2) = 1$		
Input	$u_1 = x^2 + u_{11}x + u_{10}$, $v_1 = x^3 + v_{11}x + v_{10}$	
	$u_2 = x^2 + u_{21}x + u_{20}$, $v_2 = x^3 + v_{21}x + v_{20}$	
Output	$[u', v'] = [u_1, v_1] + [u_2, v_2]$	
Step	Expression	Operations
	Composition	
1	$\underline{inv = z_1x + z_2}$	4M
	$z_0 = u_{10} - u_{20}$, $z_1 = u_{11} - u_{21}$	
	$z_2 = u_{11} \cdot z_1 - z_0$, $z_3 = u_{10} \cdot z_1$	
	$r = z_1 \cdot z_3 - z_0 \cdot z_2$	
2	$\underline{s' = s'_1x + s'_0}$	4M
	$w_0 = v_{10} - v_{20}$, $w_1 = v_{11} - v_{21}$	
	$s'_1 = w_0 \cdot z_1 - w_1 \cdot z_0$, $s'_0 = w_0 \cdot z_2 - w_1 \cdot z_3$	
	Reduction	
3	$\underline{k = k_2x^2 + k_1x + k_0}$	
	$k_2 = f_4 - 2v_{21}$	
4	$\underline{s = \frac{1}{r}s' = s_1x + s_0}$	1I, 2S, 6M
	$r_2 = r^2$, $\widehat{w}_0 = r_2 - (s'_1 + r)^2 (= r^2m_4)$, $\widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$,	
	$\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r})$, $\widehat{w}_3 = r \cdot r_2 \cdot \widehat{w}_1 (= \frac{1}{m_4})$	
	$s_1 = s'_1 \cdot \widehat{w}_2$, $s_0 = s'_0 \cdot \widehat{w}_2$	
5	$\underline{l = l_3x^3 + l_2x^2 + l_1x + l_0 \text{ (note that } l_3 = s_1)}$	3M
	$\widetilde{w}_0 = s_0 \cdot u_{20}$, $\widetilde{w}_1 = s_1 \cdot u_{21}$, $l_2 = s_0 + \widetilde{w}_1$	
	$l_1 = (s_0 + s_1) \cdot (u_{21} + u_{20}) - \widetilde{w}_1 - \widetilde{w}_0$, $l_0 = \widetilde{w}_0$	
6	$\underline{m' = x^4 + m'_3x^3 + m'_2x^2 + m'_1x + m'_0,\ u' = x^2 + u'_1x + u'_0}$	6M
	$m'_3 = \widehat{w}_3 \cdot (-s_1 \cdot (s_0 + l_2) - 2s_0) (= \frac{m_3}{m_4})$	
	$m'_2 = \widehat{w}_3 \cdot (k_2 - s_1 \cdot (l_1 + 2v_{21}) - s_0 \cdot l_2) (= \frac{m_2}{m_4})$	
	$u'_1 = m'_3 - u_{11}$, $u'_0 = m'_2 - u_{10} - u_{11} \cdot u'_1$	
7	$\underline{v' = x^3 + v'_1x + v'_0}$	3M
	$\underline{w}_1 = u'_1 \cdot (s_1 + 2)$, $\underline{w}_0 = u'_0 \cdot (l_2 - \underline{w}_1)$	
	$v'_1 = (u'_0 + u'_1) \cdot (s_1 - \underline{w}_1 + l_2) - v_{21} - l_1 - \underline{w}_0 - \underline{w}_1$	
	$v'_0 = \underline{w}_0 - v_{20} - l_0$	
Total		1I, 2S, 26M

The resulting explicit formulas are presented in Table 3. The special cases when $s'_1 = 0$ in Step 4 and when $\widetilde{w}_0 = 0$ in Step 5 need to be handled separately. As these occur only rarely, we do not describe the required modifications here, rather, they will appear in the full version of the paper.

Table 3. Explicit Formulas for Doubling Divisor Classes

Doubling, Reduced Basis, $\deg u = 2$		
Input	$[u, v], u = x^2 + u_1x + u_0, v = x^3 + v_1x + v_0$	
Output	$[u', v'] = 2[u, v] := [u, v] + [u, v]$	
Step	Expression	Operations
1	$\underline{\widetilde{v} = \widetilde{v}_1x + \widetilde{v}_0}$ $w_1 = u_1^2,\ \widetilde{v}_1 = 2(v_1 + w_1 - u_0),\ \widetilde{v}_0 = 2(v_0 + u_0 \cdot u_1)$	1S, 1M
2	$\underline{r = res(\widetilde{v}, u),\ inv = inv_1x + inv_0}$ $w_2 = u_0 \cdot \widetilde{v}_1,\ w_3 = u_1 \cdot \widetilde{v}_1$ $inv_1 = \widetilde{v}_1,\ inv_0 = w_3 - \widetilde{v}_0$ $r = \widetilde{v}_0 \cdot inv_0 - w_2 \cdot \widetilde{v}_1$	4M
3	$\underline{k' \equiv (f - v^2)/u \pmod{u} = k'_1x + k'_0:}$ $k'_2 = f_4 - 2v_1,$ $k'_1 = f_3 - 2v_0 - 2k'_2 \cdot u_1,$ $k'_0 = f_2 - v_1^2 - k'_1 \cdot u_1 - k'_2 \cdot (w_1 + 2u_0)$	1S, 3M
4	$\underline{s' = s'_1x + s'_0}$ $s'_1 = inv_1 \cdot k'_0 - \widetilde{v}_0 \cdot k'_1,\ s'_0 = inv_0 \cdot k'_0 - w_2 \cdot k'_1$	4M
5	$\underline{\text{Inversion, } r^{-1},\ s_0,\ s_1,\ \widetilde{u}_2^{-1}}$ $r_2 = r^2,\ \widehat{w}_0 = (s'_1 + r)^2 - r_2 (= r^2\widetilde{u}_2),\ \widehat{w}_1 = (r \cdot \widehat{w}_0)^{-1}$ $\widehat{w}_2 = \widehat{w}_0 \cdot \widehat{w}_1 (= \frac{1}{r}),\ \widehat{w}_3 = r \cdot r_2 \cdot \widehat{w}_1 (= \frac{1}{\widetilde{u}_2})$ $s_1 = \widehat{w}_2 \cdot s'_1,\ s_0 = \widehat{w}_2 \cdot s'_0$	I, 2S, 6M
6	$\underline{u' = x^2 + u'_1x + u'_0}$ $u'_1 = 2\widehat{w}_3 \cdot ((s_0 - u_1) \cdot s_1 + s_0)$ $u'_0 = \widehat{w}_3 \cdot ((s_0 - 2u_1) \cdot s_0 + \widetilde{v}_1 \cdot s_1 - k'_2)$	5M
7	$\underline{v' = x^3 + v'_1x + v'_0}$ $z_0 = u'_0 - u_0,\ z_1 = u'_1 - u_1$ $\underline{w}_0 = z_0 \cdot s_0,\ \underline{w}_1 = z_1 \cdot s_1$ $v'_1 = 2u'_0 - v_1 + (s_0 + s_1) \cdot (z_0 + z_1) - \underline{w}_0 - \underline{w}_1 - u'_1 \cdot (2u'_1 + \underline{w}_1)$ $v'_0 = \underline{w}_0 - v_0 - u'_0 \cdot (2u'_1 + \underline{w}_1)$	5M
Total		1I, 4S, 28M

3.4 Summary of Results

The best known results for the imaginary case are found in [10]. As noted earlier, an improvement of one less squaring has been found which applies to the addition formula in the imaginary case (though not in the doubling case). Compared to the imaginary case, the addition formulas for the real case requires four more multiplications in the main case. The doubling formulas require six more multiplications but one less squaring than the imaginary case. It is worth noting that the baby step operation is the cheapest of all, and that there is no analogue for this operation in the imaginary case. Table 4 summarizes the comparison.

Table 4. Comparison of Operation Counts for Explicit Formulas

	Imaginary	Real
Baby Step	NA	1I, 2S, 4M
Addition	1I, 2S, 22M [10]	1I, 2S, 26M
Doubling	1I, 5S, 22M [10]	1I, 4S, 28M

The main obstruction from getting more competitive formulas in the real case is the extra coefficient interfering with the inversion step. In the imaginary case, the leading coefficient of the new u is simply s_1^2, which allows one to simplify both addition and doubling formulas. In the real case, we found that computing s_0 and s_1 explicitly was the most efficient way to compute addition and doubling of divisors.

4 Numerical Results

As cryptographic applications were one of our motivations for developing explicit formulas for divisor arithmetic on genus 2 real hyperelliptic curves, we have implemented key exchange protocols in the imaginary and real models in order to determine whether the real model can be competitive with the imaginary model in terms of efficiency. In the imaginary case, the main operation is scalar multiplication using a non-adjacent form (NAF) expansion of the multiplier, which we will refer to as SCALAR-MULT. In the real case, there are two variations of scalar multiplication described in [7] that comprise the key exchange protocol. Algorithm VAR-DIST2 is a variation of NAF-based scalar multiplication using only degree 0 divisor class doubling and baby steps, whereas Algorithm FIXED-DIST2 generalizes the usual NAF-based scalar multiplication algorithm. The costs of these each of these algorithms in terms of divisor class additions, doublings, and baby steps, assuming that the NAF representation of the corresponding scalar multiplier has $l+1$ bits, is recalled from [7] in Table 5. All three

Table 5. Operation counts for scalar multiplication in $\mathcal{R}$

	Doubles	Adds	Baby Steps
Imaginary (SCALAR-MULT)	l	$l/3$	-
Real, Variable Distance (VAR-DIST2)	l	$l/3$	d
Real, Fixed Distance (FIXED-DIST2)	l	1	$l/3$

of these algorithms were implemented, using the explicit formulas from [10] for the imaginary case and the formulas in this paper for the real case.

We used the computer algebra library NTL [19] for finite field and polynomial arithmetic and the GNU `C++` compiler version 3.4.3. The computations described below were performed on a Pentium IV 2.4 GHz computer running Linux. Although faster absolute times could be obtained using customized implementations of finite field arithmetic, our goal was to compare the relative performance of algorithms in the imaginary and real settings using exactly the same finite fields as opposed to producing the fastest times possible. Thus, NTL was sufficient for our purposes.

All three algorithms were implemented using curves defined over prime finite fields $\mathbb{F}_p$ where $p > 3$. We ran numerous examples of the three scalar multiplication algorithms using curves with genus 2 where the underlying finite field was chosen so that the size of $J(K)$, and hence the set $\mathcal{R}$, was roughly 2^{160}, 2^{224}, 2^{256}, 2^{384}, and 2^{512}. Note that $|J(K)| \approx p^2$ in this case, and that most likely $|\mathcal{R}| = |J(K)|$ for a randomly-chosen curve. Thus, curves offer 80, 112, 128, 192, and 256 bits of security for cryptographic protocols based on the corresponding DLP. NIST [14] currently recommends these five levels of security for key establishment in U.S. Government applications.

For the finite field, we chose a random prime p of appropriate length such that p^2 had the required bit length. For each finite field, we randomly selected 5000 curves and executed Diffie-Hellman key exchange once for each curve. Thus, we ran 10000 instances of Algorithm SCALAR-MULT (two instances for each participant using each curve) and 5000 instances each of Algorithm FIXED-DIST2 and VAR-DIST2 (one instance of each algorithm per participant using each curve). The random exponents used had $160, 224, 256, 384$, and 512 bits, respectively, ensuring that the number of bits of security provided corresponds to the five levels recommended by NIST (again, considering only generic attacks). In order to provide a fair comparison between the three algorithms, the same sequence of random exponents was used for each run of the key exchange protocol.

Table 6 contains the average CPU time in seconds for each of the three algorithms. The times required to generate domain parameters required for our real hyperelliptic curve protocols (see [7]), are not included in these timings, as domain parameter generation is a one-time computation that is performed when the public keys are generated. The time for Algorithm SCALAR-MULT is denoted by "Imag," the time for Algorithm FIXED-DIST2 by "Fixed" and that for Algorithm VAR-DIST2 by "Var." We also list the times required to execute Diffie-Hellman key exchange using both real and imaginary models. Note that in the imaginary case this amounts to two executions of Algorithm SCALAR-MULT, and in the real case one execution of VAR-DIST2 and one of FIXED-DIST2. The run-times achieved using the real model are slower than those using the imaginary model, but they are certainly close.

Table 6. Scalar multiplication and key exchange timings over $\mathbb{F}_p$ (in seconds).

Security Level (bits)	Imag	Fixed	Var	DH Imag	DH Real
80	0.0048	0.0050	0.0056	0.0097	0.0106
112	0.0083	0.0085	0.0096	0.0166	0.0180
128	0.0103	0.0106	0.0117	0.0206	0.0223
192	0.0220	0.0230	0.0256	0.0442	0.0485
256	0.0403	0.0411	0.0452	0.0806	0.0863

5 Conclusions

The formulas presented in this paper are the first explicit formulas for divisor arithmetic on a real hyperelliptic curve. Although they are a few field multiplications slower than their imaginary counterparts, they will certainly out-perform a generic implementation of Cantor's algorithm and will be useful for any computational tasks in the class group or infrastructure. Unfortunately cryptographic protocols using our formulas in the real model are also slower than those using the imaginary case, even with the improved protocols described in [7] in which many divisor additions are traded for significantly faster baby steps. Nevertheless, we hope the fact that we can achieve run times close to those in the imaginary case will increase interest in cryptographic protocols in this setting.

There is still much work to be done on this topic. As mentioned earlier, formulas for degree 0 divisor class arithmetic that work for the general form of the curve equation and any finite field, including characteristic 2, will be presented in the full version of this paper. As in [10], there are certain special cases that can arise in the formulas, for example, the polynomial s may have degree 1 instead of degree 2. As in the imaginary setting, this can be exploited to simplify the formulas; these cases will also be dealt with in the full version of the paper.

We continue to look for improvements to the formulas presented here. Reducing the number of field multiplications required for addition and doubling by only two or three would likely result in the cryptographic protocols in the real setting being slightly faster than the imaginary case. Another possible improvement that would improve the performance of the protocols in the real setting is compound operations. In particular, compounding the doubling and baby step operations will almost certainly save a few multiplication and require would likely require only one inversion (as opposed to two) as compared to performing them separately. This would improve the speed of the VAR-DIST2 scalar multiplication algorithm (doubling and baby steps) from [7].

Finally, a great deal of work has been done on explicit formulas in the imaginary setting including using projective coordinates to obtain inversion-free formulas, formulas for genus 3 and 4, explicit formulas via theta functions, and explicit formulas via NUCOMP. All of these topics are work in progress.

Acknowledgements This paper is an outcome of a research project proposed at the RMMC Summer School in Computational Number Theory and Cryptography which was held at the University of Wyoming in 2006. We would like to thank the following sponsors for their support: the University of Wyoming, the National Science Foundation (grant DMS-0612103), the Rocky Mountain Mathematics Consortium, The Number Theory Foundation, The Institute for Mathematics and its Applications (IMA), The Fields Institute, The Centre for Information Security and Cryptography (CISaC) and iCORE of Canada.

References

1. R.M. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations. In *Cryptographic Hardware and Embedded Systems—CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 148–162, Springer-Verlag, Berlin, 2004.
2. H. Cohen and G. Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Number 34 in Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2005.
3. A. Enge. How to distinguish hyperelliptic curves in even characteristic. In K. Alster, J. Urbanowicz, and H. C. Williams, editors, *Public-Key Cryptography and Computational Number Theory*, pages 49–58, De Gruyter, Berlin, 2001.
4. P. Gaudry. On breaking the discrete log on hyperelliptic curves. In *Advances in Cryptology - Eurocrypt'2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 19–34. Springer-Verlag, 2000.
5. P. Gaudry, E. Thomé, and N. Thériault and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, 76:475–492, 2007.
6. M.J. Jacobson, Jr., A.J. Menezes, and A Stein. Hyperelliptic curves and cryptography. In *High Primes and Misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams*, volume 41 of *Fields Institute Communications Series*, pages 255–282. American Mathematical Society, 2004.
7. M.J. Jacobson, Jr., R. Scheidler, and A. Stein. Cryptographic protocols on real and imaginary hyperelliptic curves. Conditionally accepted to Advances in Mathematics of Communications pending revisions, 2007.
8. M.J. Jacobson, Jr., R. Scheidler, and A. Stein. Fast Arithmetic on Hyperelliptic Curves Via Continued Fraction Expansions. To appear in Advances in Coding Theory and Cryptology, Series on Coding, Theory and Cryptology, 2, World Scientific Publishing, 2007.
9. N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1988.
10. T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication, and Computing*, 15:295–328, 2005.
11. A.J. Menezes, Y. Wu, and R.J. Zuccherato. An elementary introduction to hyperelliptic curves. Technical Report CORR 96-19, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, 1996. In: Koblitz, N.: Algebraic Aspects of Cryptography. Springer-Verlag, Berlin Heidelberg New York (1998).
12. V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Mathematics of Computation*, 68:807–822, 1999.

13. D. Mumford. *Tata Lectures on Theta I, II.* Birkhäuser, Boston, 1983/84.
14. National Institute of Standards and Technology (NIST). Recommendation on key establishment schemes. NIST Special Publication 800-56, January 2003.
15. S. Paulus and H.-G. Rück. Real and imaginary quadratic representations of hyperelliptic function fields. *Mathematics of Computation*, 68:1233–1241, 1999.
16. J. Pelzl, T. Wollinger, and C. Paar. Low cost security: explicit formulae for genus-4 hyperelliptic curves. In *Selected Areas in Cryptography — SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 1–16, Springer-Verlag, Berlin, 2003.
17. R. Scheidler. Cryptography in quadratic function fields. *Designs, Codes and Cryptography*, 22:239–264, 2001.
18. R. Scheidler, A. Stein, and H.C. Williams. Key-exchange in real quadratic congruence function fields. *Designs, Codes and Cryptography*, 7:153–174, 1996.
19. V. Shoup. NTL: A library for doing number theory. Software, 2001. See http://www.shoup.net/ntl.
20. A. Stein. Sharp upper bounds for arithmetics in hyperelliptic function fields. *Journal of the Ramanujan Mathematical Society*, 9-16(2):1–86, 2001.
21. T. Wollinger, J. Pelzl, and C. Paar. Cantor versus Harley: optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems. *IEEE Transactions on Computers*, 54:861–872, 2005.

A Divisor Addition

To perform divisor addition, we compute the following expressions, then show that these formulas give the desired result.

$$\begin{aligned}
&s \equiv (v_1 - v_2) \cdot (u_2)^{-1} \pmod{u_1} && l = s \cdot u_2 \\
&k = \tfrac{f - v_2^2}{u_2} && \\
&m = k - s \cdot (l + 2v_2) && m' = m/m_4 = \ m \text{ made monic} \\
&u' = m'/u_1 && v' = H(y) - [(H(y) + v_2 + l) \pmod{u'}] \ .
\end{aligned}$$

Let (u_1, v_1) and (u_2, v_2) be two reduced divisors written in the Mumford representation. Assume u_1 and u_2 are both degree 2 and are relatively prime. The composition step of Cantor's Algorithm is given by

$$\begin{aligned}
U_0 &= u_1 u_2 \\
V_0 &\equiv v_2 + s u_2 = v_2 + l \pmod{U_0}
\end{aligned}$$

where $s \equiv u_2^{-1}(v_1 - v_2) \pmod{u_1}$ and $l = s u_2$. The reduction step can be expressed as

$$\begin{aligned}
V_1 &= -V_0 + \left\lfloor \tfrac{V_0 + H(y)}{U_0} \right\rfloor \cdot U_0 \\
U_1 &= \tfrac{f - V_1^2}{U_0}
\end{aligned}$$

where $H(y)$ is the principal part of a root of the equation $y^2 = f(x)$. Since V_0 and d both have degree 3 and U_0 has degree 4, $\left\lfloor \frac{V_0 + H(y)}{U_0} \right\rfloor = 0$, and so

$$V_1 = -V_0 = -(v_2 + l)$$

Plugging this into the formula U_1 yields

$$\begin{aligned} U_1 &= \tfrac{f-(v_2+l)^2}{u_1u_2} \\ &= \tfrac{f-v_2^2-l^2-2v_2l}{u_1u_2} \\ &= \tfrac{1}{u_1}\left(\tfrac{f-v_2}{u_2} - \tfrac{l(l+2v_2)}{u_2}\right) \\ &= \tfrac{k-s(l+2v_2)}{u_1} \end{aligned}$$

where $k = \frac{f-v_2}{u_2}$.

The final output is $[u', v']$ transformed to reduced basis, i.e., $u' = U_1$ made monic, and $v' = H(y) - [(H(y) - V_1) \pmod{u'}]$. In the formulas, we first find $m = k - s(l + 2v_2)$, find the leading coefficient and compute its inverse, then compute $m' = m$ made monic.

B Divisor Doubling

To perform divisor doubling, we compute the following expressions, then show that these formulas give the desired result.

$$\begin{aligned} k &= \tfrac{f-v^2}{u} & s &\equiv k \cdot (2v)^{-1} \pmod{u} \\ \widetilde{u} &= s^2 + \tfrac{(2v)\cdot s-k}{u} & u' &= \widetilde{u} \text{ made monic} \\ v' &= H(y) - [(H(y) + s \cdot u + v) \pmod{u'}] \end{aligned}$$

Let $(u, v) = (x^2+u_1x+u_0, x^3+v_1x+v_0)$ be a degree two reduced basis Mumford representation with both points of the divisor are not equal to their opposites. Then Cantor's Algorithm for doubling the divisor (u, v) must result in (U_1, V_1) such that

$$\begin{aligned} &U_0 = u^2 \\ &V_0 \equiv v \pmod{u} \\ &(V_0 = v + su \text{ for some } s) \\ &V_1 = -V_0 + \left\lfloor \tfrac{V_0+H(y)}{U_0} \right\rfloor U_0 \\ &U_1 = \tfrac{f-V_1^2}{U_0} \end{aligned}$$

Here, s is chosen such that U_0 divides $V_0^2 - f$. Again, $\left\lfloor \frac{V_0+H(y)}{U_0} \right\rfloor$ is zero since U_0 has degree 4 and $V_0 + H(y)$ has degree 3. Hence, $V_1 = -V_0 = -v - su$ and

$$\begin{aligned} U_1 &= \tfrac{f-(-v-su)^2}{u^2} \\ &= \tfrac{f-v^2-2vsu-s^2u^2}{u^2} \\ &= \tfrac{1}{u}\left(\tfrac{f-v^2}{u} - 2vs\right) - s^2 \\ &= \tfrac{1}{u}(k - 2vs) - s^2 \end{aligned}$$

where the division in $k = (f - v^2)/u$ is exact. To ensure that the division of $k - 2vs$ by u is exact, we choose $s \equiv -k \cdot (-2v)^{-1} \pmod{u}$, and obtain

$$k - 2vs \equiv k + 2v \cdot k \cdot (-2v)^{-1} \equiv 0 \pmod{u} \ .$$

Finally, U_1 will be made monic, to arrive at

$$\begin{aligned} u' &= s^2 + \tfrac{2vs-k}{u} \text{ made monic} \\ v' &= H(y) - [(H(y) + v + su) \bmod u'] \end{aligned}$$

where $[u', v']$ is in reduced basis.