

## FAST IDEAL CUBING IN IMAGINARY QUADRATIC NUMBER AND FUNCTION FIELDS

L. IMBERT

CNRS, PIMS  
Department of Mathematics and Statistics  
University of Calgary  
2500 University Drive NW  
Calgary, Alberta, Canada T2N 1N4  
&  
CNRS, LIRMM  
Université Montpellier 2  
161 rue Ada  
F-34095 Montpellier, France

M. J. JACOBSON, JR.

Department of Computer Science  
University of Calgary  
2500 University Drive NW  
Calgary, Alberta, Canada T2N 1N4

A. SCHMIDT

Department of Computer Science  
University of Calgary  
2500 University Drive NW  
Calgary, Alberta, Canada T2N 1N4

(Communicated by the associate editor name)

**ABSTRACT.** We present algorithms for computing the cube of an ideal in an imaginary quadratic number field or function field. In addition to a version that computes a non-reduced output, we present a variation based on Shanks' NUCOMP algorithm that computes a reduced output and keeps the sizes of the intermediate operands small. Extensive numerical results are included demonstrating that in many cases our formulas, when combined with double base chains using binary and ternary exponents, lead to faster exponentiation.

### 1. INTRODUCTION

Exponentiation of ideals in quadratic number fields and function fields (aka hyperelliptic curves) has a number of applications in cryptography and computational number theory. For example, the main step in public-key cryptosystems whose security is based on the presumed intractability of the discrete logarithm problem in the

---

2000 *Mathematics Subject Classification*: Primary: 94A60, 14H45; Secondary: 14Q05.

*Key words and phrases*: quadratic fields, quadratic function fields, hyperelliptic curves, ideal arithmetic, double base number systems.

The second author is supported in part by NSERC of Canada. This work was carried out during Laurent Imbert's leave at the University of Calgary under the CNRS, PIMS-Europe agreement UMI 3069.

class group is to compute a reduced ideal equivalent to  $\mathfrak{g}^n$ , where  $\mathfrak{g}$  is some publicly-available reduced ideal and  $n$  is a randomly-selected positive integer. Sutherland's algorithms for computing the order of an element in a group and the structure of a group [19], when applied to class groups, require the same exponentiation operation for a randomly-selected ideal  $\mathfrak{g}$  and a large, smooth exponent  $n$ . Thus, improvements to the speed of ideal exponentiation translate directly into improvements for both of these applications.

Fast exponentiation is usually implemented using repeated squarings and multiplications based on the standard binary representation of the exponent. Although the number of squarings is fixed and equals the bit-length of the exponent, the number of multiplications depends on the number of non-zero digits in its binary expansion. Recoding techniques such as signed digits, for example non-adjacent form (NAF) or window methods, reduce the number of multiplications required. When a fast cubing operation is available, *i.e.*, the time for a cubing is less than that of a squaring followed by a multiplication, hybrid binary/ternary methods such as double-base chains, for which the average number of multiplications is generally small, can be advantageous [5, 6].

The present work is an attempt to speed up ideal exponentiation in quadratic number fields and function fields by using double-base chains with binary and ternary exponents. To this end, we have developed formulas for cubing an ideal based on the standard ideal multiplication algorithm for quadratic number fields [13] and Cantor's algorithm for quadratic function fields [4]. The cost of ideal multiplication, squaring, and cubing is dominated in both cases by extended GCD computations and ideal reduction. Cubing an ideal by squaring and multiplying requires two extended GCD computations in general and at most three, whereas our new formulas require only one in general and at most two. The number of reduction steps required in both cases is roughly the same, but the drawback of our cubing operation is that the coefficients of the resulting non-reduced ideal are significantly larger than those obtained by squaring and multiplying. This disadvantage could be mitigated by using an improved reduction algorithm due to Jacobson, Sawilla, and Williams [9] in quadratic fields and its generalization to the function field case [1].

Another approach to mitigating the growth in intermediate operands is to use Shanks' NUCOMP algorithm [17, 13]. NUCOMP essentially performs a type of reduction on the intermediate operands in the ideal multiplication algorithm before computing the product. The result is an output that is in many cases completely reduced but at least almost reduced, and although more operations are required than when simply multiplying and reducing, the sizes of the operands are significantly smaller. As the description of NUCOMP for quadratic fields from [13] is the most efficient in practice, we generalize this version to the function field case and use it as the basis for a fast cubing algorithm, which we refer to as NUCUBE.

We have run extensive numerical experiments in order to assess the efficiency of our cubing algorithms. We compare ideal exponentiation using a standard left-to-right binary method, left-to-right NAF, left-to-right double base chains and right-to-left double base chains. For the double base chain methods, we use binary-ternary exponents and our new cubing algorithms. In many cases, except for very large parameters for which the cost of our cubing formula become more expensive than the combined costs of a squaring and a multiplication, the double base methods with our NUCUBE algorithms offer a significant improvement over the binary and NAF methods.

The remainder of this paper is organized as follows. In Section 2 we present the background on ideals in function fields and continued fraction expansions that is required to derive our cubing algorithms. Our cubing formulas for the function field case are derived and presented in Section 3, followed by our function field NUCUBE algorithm in Section 4. The number field versions of cubing and NUCUBE are presented in Section 5. Our numerical results are presented in Section 6, after which we conclude with a summary of open problems and possible directions of further research.

## 2. CONTINUED FRACTIONS AND IDEALS IN QUADRATIC FUNCTION FIELDS

Throughout this paper, let  $\mathbb{K}$  denote either an imaginary quadratic number field or imaginary quadratic (hyperelliptic) function field of genus  $g$  over a finite field  $\mathbb{F}_q$ . In the former case, we write  $\mathbb{K} = \mathbb{Q}(\sqrt{\Delta})$ , where  $\Delta < 0$  with  $\Delta \equiv 0, 1 \pmod{4}$  and  $\Delta$  or  $\Delta/4$  square-free, is a fundamental discriminant. In the latter case, taking  $\mathbb{F}_q$  as the finite field with  $q$  elements, we write  $\mathbb{K} = \mathbb{F}_q(C) = \mathbb{F}_q(x, y)$  to denote the function field of an imaginary hyperelliptic curve  $C$ , *i.e.*, an absolutely irreducible non-singular curve of the form

$$C : y^2 + h(x)y = f(x) ,$$

where  $f, h \in \mathbb{F}_q[x]$ ,  $f$  is monic of degree  $2g + 1$ , and  $h = 0$  if  $q$  is odd and is monic with  $\deg(h) \leq g$  if  $q$  is even.

In this section, we confine our attention to the function field case. The results for number fields are analogous and can be found in the literature, for example [13, Ch. 3 and 4]. The main differences to the function field case are presented as required in Section 5. For more background on hyperelliptic curves and quadratic function fields, see [7] and [11].

The maximal order  $\mathbb{F}_q[x, y]$  of  $\mathbb{F}_q(x, y)$  is an integral domain and a  $\mathbb{F}_q[x]$ -module of rank 2 with  $\mathbb{F}_q[x]$ -basis  $\{1, y\}$ . The non-zero integral ideals in  $\mathbb{F}_q[x, y]$  are exactly the  $\mathbb{F}_q[x]$ -modules of the form  $\mathfrak{a} = \mathbb{F}_q[x]SQ + \mathbb{F}_q[x]S(P + y)$  where  $P, Q, S \in \mathbb{F}_q[x]$  and  $Q$  divides  $f + hP - P^2$ . Here,  $S$  and  $Q$  are unique up to factors in  $\mathbb{F}_q^*$  and  $P$  is unique modulo  $Q$ . For brevity, we write  $\mathfrak{a} = S(Q, P)$ . An ideal  $\mathfrak{a} = S(Q, P)$  is *primitive* if  $S \in \mathbb{F}_q^*$ , in which case we simply take  $S = 1$  and write  $\mathfrak{a} = (Q, P)$ . A primitive ideal  $\mathfrak{a}$  is *reduced* if  $\deg Q \leq g$ .

A *fractional  $\mathbb{F}_q[x]$ -ideal* is a subset  $\mathfrak{f}$  of  $\mathbb{F}_q(x, y)$  such that  $d\mathfrak{f}$  is a  $\mathbb{F}_q[x, y]$ -ideal for some non-zero  $d \in \mathbb{F}_q[x]$ . Let  $\mathcal{I}$  denote the group of non-zero fractional  $\mathbb{F}_q[x, y]$ -ideals,  $\mathcal{P}$  the subgroup of  $\mathcal{I}$  of non-zero principal fractional  $\mathbb{F}_q(x, y)$ -ideals (which we write as  $(\alpha)$  for  $\alpha \in \mathbb{F}_q(x, y)^*$ ),  $\mathcal{Cl} = \mathcal{I}/\mathcal{P}$  the ideal class group of  $\mathbb{F}_q(x, y)$ , and  $h = |\mathcal{Cl}|$  the ideal class number of  $\mathbb{F}_q(x, y)$ . It is well-known that the class number is finite, and that every ideal class has exactly one reduced ideal representative. Thus, in order to perform arithmetic in the class group, we represent equivalence classes using reduced ideals and multiply them by finding a reduced ideal equivalent to the product of the two representatives. The identity element is the principal class, and a representative of the inverse of the ideal class of  $(Q, P)$  is given by  $(Q, -P - h)$ .

The correspondence between ideals of  $\mathbb{F}_q[x, y]$  and divisors of the corresponding hyperelliptic curve is also well-known (see, for example, [11]). The ideal coefficients  $(Q, P)$  are the same coefficients as the Mumford representation of a divisor, and the ideal class group in the case of imaginary quadratic function fields is isomorphic to the divisor class group of the corresponding curve. Thus, all of the algorithms for quadratic function fields presented in this paper also work trivially for divisors.

2.1. CONTINUED FRACTION EXPANSIONS. A *continued fraction expansion* is represented by the symbolic expression

$$q_0 + \frac{1}{q_1 + \frac{1}{\ddots + \frac{1}{q_n + \frac{1}{\phi_{n+1}}}}},$$

which we denote by

$$(1) \quad \langle q_0, q_1, \dots, q_n, \phi_{n+1} \rangle$$

for short. The theory of continued fractions plays an important role in ideal arithmetic, and is required for the development of our NUCUBE algorithm.

In the following, we present the required results on continued fraction expansions of relevance to the quadratic function field case. The corresponding results for quadratic number fields are analogous, and can be found in [13, Ch. 3]; the relevant results are summarized as needed in Section 5. This function field material has been described in more detail elsewhere (see [11]). In our presentation below, we follow the notation of [13] in order to unify as much as possible our treatment of the number field and function field cases.

As shown in [11], elements  $\phi \in \mathbb{F}_q(x, y)$  can be expressed as elements in  $\mathbb{F}_q\langle x^{-1/2} \rangle$ , the field of Puiseux series in  $x^{-1/2}$ . The expression  $\phi = \langle q_0, q_1, \dots, q_n, \phi_{n+1} \rangle$  with  $q_i \in \mathbb{F}_q[x]$  is referred to as the *continued fraction expansion* of  $\phi_0$  with *partial quotients*  $q_0, q_1, \dots, q_n$ . It uniquely defines a Puiseux series  $\phi_{n+1} \in \mathbb{F}_q\langle x^{-1/2} \rangle$  where  $\phi_0 = \phi$  and

$$(2) \quad \phi_{i+1} = (\phi_i - q_i)^{-1}$$

for  $0 \leq i \leq n$ . If we set  $A_{-2} = 0$ ,  $A_{-1} = 1$ ,  $B_{-2} = 1$ ,  $B_{-1} = 0$  and define for  $i = 0, 1, \dots$

$$\begin{aligned} A_i &= q_i A_{i-1} + A_{i-2} \\ B_i &= q_i B_{i-1} + B_{i-2} \end{aligned},$$

then  $A_i/B_i = \langle q_0, q_i, \dots, q_i \rangle$  for  $i = 0 \leq i \leq n-1$ . By induction, it is easy to see that

$$(3) \quad A_i B_{i-1} - B_i A_{i-1} = (-1)^{i-1}.$$

If the  $q_i$ , rather than being arbitrary polynomials in  $\mathbb{F}_q[x]$ , are chosen as  $q_i = \lfloor \phi_i \rfloor$ , then (1) is the well-known regular continued fraction expansion of  $\phi$ . The partial quotients  $q_0, q_1, \dots$  are uniquely determined by  $\phi$  and  $\deg(q_i) \geq 1$  for all  $i \in \mathbb{N}$ . The rational function  $A_i/B_i = \langle q_0, q_1, \dots, q_i \rangle$  is called the  *$i$ th convergent* of  $\phi$ .

If instead we take  $\phi = K/L$  for  $K, L \in \mathbb{F}_q[x]$ , then the regular continued fraction expansion of  $\phi$  as just defined is nothing more than the simple continued fraction expansion of the rational function  $K/L$ . In this case, the Euclidean algorithm can be used to compute it as follows. Set  $R_{-2} = K$ ,  $R_{-1} = L$  and define for  $i = 0, 1, \dots, n-1$

$$R_i = R_{i-2} - q_i R_{i-1}, \quad \text{where } q_i = \lfloor R_{i-2}/R_{i-1} \rfloor.$$

It can be shown by induction that

$$(4) \quad (-1)^{i+1} R_i = L A_i - K B_i$$

and

$$(5) \quad B_i R_{i-1} + B_{i-1} R_i = L .$$

We also require the sequence  $\{C_i\}_{-2 \leq i < n}$  with  $C_i = (-1)^{i+1} B_i$  for our NUCUBE algorithm. It is easy to see that

$$C_i = C_{i-2} - q_i C_{i-1} ,$$

so the  $C_i$  can be computed directly during the computation of the  $R_i$  and  $q_i$ .

Suppose now that  $\mathfrak{a}_0 = (Q_0, P_0)$  denotes a primitive ideal in an imaginary quadratic function field  $\mathbb{F}_q(x, y)$ . Put  $\phi_0 = (P_0 + y)/Q_0$ , and let  $q_0, q_1, \dots$  be any sequence of polynomials in  $\mathbb{F}_q[x]$ . Define

$$(6) \quad P_{i+1} = q_i Q_i - P_i + h, \quad Q_{i+1} = \frac{f + h P_{i+1} - P_{i+1}^2}{Q_i} ,$$

for  $i \geq 0$ . If we set  $\phi_i = (P_i + y)/Q_i$  and  $\phi_{i+1} = (\phi_i - q_i)^{-1}$ , then for all  $i \geq 0$ , we have  $\phi_0 = \langle q_0, q_1, \dots, q_i, \phi_{i+1} \rangle$ . Thus, (6) determines a continued fraction expansion of  $\phi_0$  in  $\mathbb{F}_q\langle x^{-1/2} \rangle$ , and moreover defines a sequence  $\mathfrak{a}_i = (Q_{i-1}, P_{i-1})$  of primitive ideals.

Set  $\theta_1 = 1$  and  $\theta_i = \prod_{j=1}^{i-1} \phi_j^{-1}$  for  $i \geq 2$ . Since  $\phi_i \bar{\phi}_i = -Q_{i-1}/Q_i$ , where  $\overline{a + by} = a + b(-h - y)$  denotes the hyperelliptic involution of  $a + by \in \mathbb{F}_q(x, y)$ , it is easy to see that  $Q_0 \theta_i \bar{\theta}_i = (-1)^{i-1} Q_{i-1}$ . Thus

$$\bar{\theta}_i = \prod_{j=1}^{i-1} \bar{\phi}_j^{-1} = (-1)^{i-1} \frac{Q_{i-1}}{Q_0 \theta_i} = (-1)^{i-1} \frac{Q_{i-1}}{Q_0} \prod_{j=1}^{i-1} \phi_j .$$

Then  $\mathfrak{a}_{i+1} = (\bar{\phi}_i^{-1}) \mathfrak{a}_i$  and hence  $\mathfrak{a}_i = (\bar{\theta}_i) \mathfrak{a}_0$ , for  $i \in \mathbb{N}$ . Therefore, the ideals  $\mathfrak{a}_i$  are all equivalent, so the continued fraction expansion of  $\phi_0$  produces a sequence of equivalent ideals. If we choose the  $q_i$  in (6) to be  $\lfloor \phi_i \rfloor$ , *i.e.* the partial quotients in the regular continued fraction expansion of  $\phi_0$  in  $\mathbb{F}_q\langle x^{-1/2} \rangle$ , then, as shown in [11], the sequence of ideals  $\mathfrak{a}_i$  given by (6) produces a reduced ideal equivalent to  $\mathfrak{a}_0$  after at most  $\lceil (\deg(Q_0) - g)/2 \rceil$  steps.

There are two main ingredients to computing the reduced product of two ideals using NUCOMP as described in [11], and in our NUCUBE algorithm. Suppose that multiplying two ideals yields the non-reduced ideal  $(Q_0, P_0)$ . The first trick is to recover the same partial quotients  $q_i$  that lead to a reduced ideal without having to explicitly compute  $Q_0$  and  $P_0$ . We show in Section 4 that this can be done by computing the simple continued fraction expansion of a certain rational function whose numerator and denominator have degree smaller than that of  $Q_0$ . The second is to derive alternative formulas for  $Q_{i+1}$  and  $P_{i+1}$  using the partial quotients  $q_i$  that do not involve computing all of  $(Q_j, P_j)$  for  $0 \leq j \leq i$ . The following results on continued fraction expansions, adapted from the corresponding results for quadratic number fields described in [13, Ch. 3], are needed for this purpose.

First, note that

$$(7) \quad \phi_0 = \frac{\phi_{i+1} A_i + A_{i-1}}{\phi_{i+1} B_i + B_{i-1}} \quad \text{and} \quad \phi_{i+1} = -\frac{\phi_0 B_{i-1} - A_{i-1}}{\phi_0 B_i - A_i} .$$

The first identity can be shown by repeatedly substituting (2) for  $\phi_{i+1}, \phi_i, \dots, \phi_1$  and simplifying. The second can be obtained by algebraic manipulations to the first.

Next, define

$$(8) \quad G_i = Q_0 A_i - P_0 B_i .$$

It can be shown that

$$(9) \quad (-1)^{i-1} Q_0 = G_i B_{i-1} - B_i G_{i-1}$$

by substituting (8) for  $G_i$  and  $G_{i-1}$ . From (7) it follows that

$$(10) \quad \phi_{i+1} = -\frac{G_{i-1} - yB_{i-1}}{G_i - yB_i} ,$$

and by using (2) and induction, we have

$$(11) \quad \phi_i = \frac{P_i + y}{Q_i} ,$$

where  $Q_i$  and  $P_i$  are obtained from the continued fraction expansion of  $\phi_0$  as per (6).

Finally, by combining (10) and (11) and using (9), we can show that

$$(12) \quad \begin{aligned} (-1)^{i+1} Q_{i+1} &= (G_i^2 + hG_i B_i - fB_i^2)/Q_0 \\ (-1)^{i+1} P_{i+1} &= (fB_i B_{i-1} - hB_i G_{i-1} - G_i G_{i-1})/Q_0 , \end{aligned}$$

giving us the means to compute  $Q_{i+1}$  and  $P_{i+1}$  given only the partial quotients  $q_j$  and without having to compute the  $Q_j$  and  $P_j$  for  $0 \leq j \leq i$ . The equations (12) also imply that

$$(13) \quad G_i = P_{i+1} B_i + Q_{i+1} B_{i-1} - hB_i ,$$

giving us a means to recover one of  $Q_{i+1}$  or  $P_{i+1}$  given the other.

### 3. CUBING FORMULAS

Formulas for ideal multiplication can be found in many sources (see, for example, [11]), and dedicated formulas for ideal squaring can be obtained from these general formulas. See [13, Section 5.4] for a description of how these formulas are derived in the number field case (the function field case is analogous). Special-purpose formulas for ideal cubing can be developed using the same methods.

Let  $\mathfrak{a} = (Q', P')$  be a primitive ideal in some imaginary quadratic function field  $\mathbb{F}_q(x, y)$ , and set  $R' = (f + P'h - (P')^2)/Q'$ . We now describe an algorithm to compute  $\mathfrak{a}^3 = S(Q, P)$  along with  $R = (f + Ph - P^2)/Q$ , as having  $R$  available is computationally useful in the subsequent reduction process (see [11]).

In the general case, we have

$$(14) \quad \begin{aligned} \mathfrak{a}^3 &= (Q' \mathbb{F}_q[x] + (P' + y) \mathbb{F}_q[x])^3 \\ &= (Q')^3 \mathbb{F}_q[x] + (Q')^2 (P' + y) \mathbb{F}_q[x] + Q' (P' + y)^2 \mathbb{F}_q[x] + (P' + y)^3 \mathbb{F}_q[x] \\ &= (Q')^3 \mathbb{F}_q[x] + ((Q')^2 P' + (Q')^2 y) \mathbb{F}_q[x] + (Q'((P')^2 + f) + Q'(2P' - h)y) \mathbb{F}_q[x] \\ &\quad + (((P')^3 + 3P'f - hf) + ((2P' - h)^2 + Q'R')y) \mathbb{F}_q[x] . \end{aligned}$$

Let

$$(15) \quad S' = \gcd(Q', 2P' - h) = u_1 Q' + v_1 (2P' - h)$$

with  $S', u_1, v_1 \in \mathbb{F}_q[x]$  and

$$(16) \quad S = \gcd(Q' S', (2P' - h)^2 + Q' R') = u_2 Q' S' + v_2 ((2P' - h)^2 + Q' R')$$

with  $S, u_2, v_2 \in \mathbb{F}_q[x]$ . Putting (15) and (16) together yields

$$(17) \quad \begin{aligned} S &= \gcd((Q')^2, Q'(2P' - h), (2P' - h)^2 + Q'R') \\ &= u_2u_1(Q')^2 + u_2v_1(Q'(2P' - h)) + v_2((2P' - h)^2 + Q'R') . \end{aligned}$$

It follows that

$$(18) \quad Q = \frac{(Q')^3}{S^2} ,$$

and by combining (14) and (17) we obtain

$$(19) \quad \begin{aligned} P &\equiv \frac{u_2u_1(Q')^2P' + u_2v_1Q'((P')^2 + f) + v_2((P')^3 + 3P'f - hf)}{S} \\ &\equiv P' + \frac{Q'}{S}R'(u_2v_1Q' + v_2(2P' - h)) \pmod{Q} . \end{aligned}$$

Note that the expression  $K = R'(u_2v_1Q' + v_2(2P' - h))$  may be reduced modulo  $(Q')^2/S$  in the computation of  $P$ . Also, note that the  $u_1$  coefficient from (15) is not used, and thus does not need to be computed when applying the extended Euclidean algorithm to  $Q'$  and  $2P' - h$ .

If the coefficient  $R = (f + Ph - P^2)/Q$  is also desired, it can be computed using the formula

$$R = \frac{SR' - K(2P' - h + KQ'/S)}{(Q')^2/S} ,$$

derived by substituting (19) into  $R = (f + Ph - P^2)/Q$ . In evaluating  $R$ ,  $K$  may be reduced modulo  $(Q')^2/S$ , the term  $KQ'/S$  may be reused from the computation of  $P$ , and  $(Q')^2/S$  may be reused from the computation of  $Q$ . As a result, using this formula is more efficient than computing  $R = (f + hP - P^2)/Q$ .

As we have just seen, an ideal cubing requires, in general, two executions of the extended Euclidean algorithm. However, if  $S' = \gcd(Q', 2P' - h) = 1$ , as one would expect to occur fairly frequently, the second execution of the extended Euclidean algorithm can be avoided as follows.

Consider the formula for the product of two ideals  $\mathfrak{a}'\mathfrak{a}'' = \mathfrak{a}$ , with  $\mathfrak{a}' = (Q', P')$ ,  $\mathfrak{a}'' = (Q'', P'')$ , and the result  $\mathfrak{a} = S(Q, P)$  given by

$$S = \gcd(Q', Q'', P' + P'' - h) = UQ' + VQ'' + W(P' + P'' - h)$$

and

$$Q = \frac{Q'Q''}{S^2}, \quad P = P'' + \frac{Q''}{S}(V(P' - P'') + WR'')$$

where  $R'' = (f + P''h - (P'')^2)/Q''$ . Using these formulas under the assumption that  $S = 1$  yields

$$(\mathfrak{a}')^2 = ((Q')^2, P' + v_1Q'R')$$

with  $R' = (f + P'h - (P')^2)/Q'$ . By applying the multiplication formulas to  $\mathfrak{a}^2$  and  $\mathfrak{a}$ , the cube  $\mathfrak{a}^3 = S(Q, P)$  of  $\mathfrak{a} = (Q', P')$  can be derived as follows. First, we have

$$\begin{aligned} S &= \gcd((Q')^2, Q', 2P' + v_1R'Q' - h) \\ &= U(Q')^2 + VQ' + W(2P' + v_1R'Q' - h) \\ &= \gcd(Q', 2P' - h + v_1R'Q') \\ &= \gcd(Q', 2P' - h) \\ &= u_1Q' + v_1(2P' - h) \\ &= 1 , \end{aligned}$$

If we set  $U = 0$ ,  $V = u_1 - v_1^2 R'$  and  $W = v_1$ , then we obtain

$$\begin{aligned} U(Q')^2 + VQ' + W(2P' + v_1 R' Q' - h) &= (u_1 - v_1^2 R')Q' + v_1(2P' + v_1 R' Q' - h) \\ &= u_1 Q' + v_1(2P' - h) \\ &= 1 \end{aligned}$$

and therefore

$$\begin{aligned} (20) \quad Q &= (Q')^3 \\ P &\equiv P' + Q'R'v_1((u_1 - v_1^2 R')Q' + 1) \\ &\equiv P' + Q'R'v_1(2 + v_1(v_1 Q'R' - 2P' + h)) \pmod{Q} . \end{aligned}$$

In this case, we can compute  $K = R'v_1(2 + v_1(v_1 Q'R' - 2P' + h))$  modulo  $(Q')^2$ , and we only need the  $v_1$  coefficient of (15). The same formula for  $R$  described above can be used, taking  $S = 1$ .

The ideal cubing formulas for quadratic function fields are summarized in Algorithm 1. Notice that this algorithm works for an arbitrary quadratic function field  $\mathbb{F}_q(x, y)$  where  $y$  is defined by  $y^2 + hy = f$ . In fact, it works for imaginary, real, and unusual quadratic function fields (see [11] for the definitions of real and unusual function fields), as the derivation of the formulas did not make use of any specific properties of  $f$  and  $h$ . The formulas in Algorithm 1 can be simplified based

---

**Algorithm 1** Ideal Cubing for Quadratic Function Fields

---

**Input:**  $\mathfrak{a} = (Q', P')$ ,  $R' = (f + hP' - (P')^2)/Q'$

**Output:**  $\mathfrak{a}^3 = S(Q, P)$ , (optional)  $R = (f + hP - P^2)/Q$

- 1: Compute  $S' = u_1 Q' + v_1(2P' - h)$  (only compute  $S'$  and  $v_1$ ).
  - 2: **if**  $S' = 1$  **then**
  - 3:   Set  $S = 1$ .
  - 4:   Set  $N = Q'$ ,  $L = N^2$ ,  $K = R'v_1(2 + v_1(v_1 Q'R' - 2P' + h)) \pmod{L}$ .
  - 5: **else**
  - 6:   Compute  $S = u_2 Q'S' + v_2((2P' - h)^2 + Q'R')$ .
  - 7:   Set  $N = Q'/S$ ,  $L = NQ'$ ,  $K = R'(u_2 v_1 Q' + v_2(2P' - h)) \pmod{L}$ .
  - 8: **end if**
  - 9: Set  $T = NK$ .
  - 10: Set  $Q = NL$ .
  - 11: Set  $P = P' + T$ .
  - 12: (optional) Set  $R = (SR' - K(2P' - h + T))/L$ .
- 

on the characteristic of  $\mathbb{F}_q$ . If the characteristic is odd, then  $h$  can be taken to be 0 in Algorithm 1. If the characteristic is even, then  $h$  is non-zero, but all terms multiplied by 2 become zero and subtractions can be written as additions.

Notice that the ideal output by Algorithm 1 is not in general normalized in the sense that  $P$  is not reduced modulo  $Q$ . This is done as part of the subsequent reduction process, so it is not necessary to perform the normalization if reduction will be immediately applied. If the normalization is required, compute  $q$  and  $r$  such that  $P = qQ + r$ , set  $R = R + q(P + r - h)$ , and set  $P = r$ .

#### 4. NUCUBE

In this section, we describe a cubing algorithm which is computationally more efficient than the algorithm presented in the previous section. Instead of computing

the ideal  $\mathfrak{a}^3$  using Algorithm 1 and reducing the output, we use the idea of NUCOMP to reduce the coefficients before cubing. When applied to a reduced ideal, we show that the reduction can be accomplished by computing the simple continued fraction expansion of  $K/L$  where  $K$  and  $L$  have degree at most  $2g$ , as opposed to operands of size  $3g$  when doing reduction after multiplication. Furthermore, we show that the output of our NUCUBE algorithm is always reduced.

We require a generalization of the NUCOMP algorithm as described in [13] for quadratic fields. The following theorem describes the approach.

**Theorem 4.1** (Theorem 5.21 of [13]). *Suppose  $Q, P, N, L, K, P', P'' \in \mathbb{F}_q[x]$  such that  $(Q, P)$  is an ideal in  $\mathbb{F}_q[x, y]$  and*

$$(21) \quad P = P' + NK, \quad Q = NL, \quad P'' \equiv P \pmod{L} .$$

If  $K/L = \langle q_0, q_1, \dots, q_n \rangle$  and we put

$$(22) \quad \frac{P+y}{Q} = \langle q_0, q_1, \dots, q_i, \frac{P_{i+1}+y}{Q_{i+1}} \rangle \quad (0 \leq i < n) ,$$

then

$$\begin{aligned} Q_{i+1} &= (-1)^{i-1}(R_i M_1 - C_i M_2), \\ P_{i+1} &= (NR_i + Q_{i+1} C_{i-1})/C_i - P' + h \end{aligned}$$

for  $-1 \leq i < n$  where

$$\begin{aligned} M_1 &= (NR_i + (P'' - P')C_i)/L \in \mathbb{F}_q[x], \\ M_2 &= (R_i(P' + P'' - h) + TC_i)/L \in \mathbb{F}_q[x], \quad T = (f + hP' - (P')^2)/N. \end{aligned}$$

*Proof.* From (4) and the definition of  $C_i$  we get

$$(23) \quad (-1)^{i+1}R_i = LA_i - KB_i = LA_i - (-1)^{i+1}KC_i .$$

By (8), (21), and (23) we have

$$\begin{aligned} G_i &= QA_i - PB_i \\ &= NLA_i - PB_i \\ (24) \quad &= N(-1)^{i+1}(R_i + KC_i) - (-1)^{i+1}C_i(P' + NK) \\ &= (-1)^{i+1}(NR_i - P'C_i) \end{aligned}$$

and

$$(25) \quad \frac{G_i + B_i P}{Q} = \frac{LA_i}{L} = \frac{(-1)^{i+1}(R_i + KC_i)}{L} .$$

implying that

$$(26) \quad R_i \equiv -KC_i \pmod{L} .$$

By (24) we have

$$\begin{aligned} \frac{fB_i + G_i(P-h)}{Q} &= \frac{fB_i + (-1)^{i+1}(NR_i - P'C_i)(P-h)}{Q} \\ (27) \quad &= (-1)^{i+1} \frac{C_i(f - P'P + hP') + NR_i(P-h)}{NL} \\ &= (-1)^{i+1} \frac{C_i(f + hP' - (P')^2) - C_iNKP' + NR_i(P-h)}{NL} \\ &= (-1)^{i+1}(C_i T - C_i KP' + R_i(P-h))/L , \end{aligned}$$

where  $T = (f + hP' - (P')^2)/N$ . By (8),  $G_i \equiv -PB_i \pmod{Q}$ , from which it follows that

$$fB_i + G_i(P - h) \equiv fB_i - PB_i(P - h) \equiv B_j(f + Ph - P^2) \equiv 0 \pmod{Q}$$

as  $(Q, P)$  is an ideal of  $\mathbb{F}_q[x, y]$ . Combining with (27) we obtain  $C_iT - C_iKP' + R_i(P - h) \equiv 0 \pmod{L}$  and using (21) and (26) we have

$$\begin{aligned} (28) \quad C_iT - C_iKP' + R_i(P - h) &\equiv C_iT + R_iP' + R_i(P - h) \\ &\equiv C_iT + R_i(P' + P - h) \\ &\equiv C_iT + R_i(P' + P'' - h) \\ &\equiv 0 \pmod{L} . \end{aligned}$$

By (12), (24), (25), and (27), we get

$$\begin{aligned} Q_{i+1} &= (-1)^{i+1}(G_i^2 + hG_iB_i - fB_i^2)/Q \\ &= (-1)^{i+1} \left( G_i \frac{G_i + B_iP}{Q} - B_i \frac{fB_i + G_i(P - h)}{Q} \right) \\ &= (-1)^{i+1} \left( (NR_i - P'C_i) \frac{(R_i + KC_i)}{L} - B_i(-1)^{i+1} \frac{C_iT - C_iKP' + R_i(P - h)}{L} \right) \\ &= (-1)^{i+1} \left( NR_i \frac{R_i + KC_i}{L} - P'C_i \frac{R_i + KC_i}{L} - R_i \frac{C_i(P - h)}{L} - C_i^2 \frac{T - KP'}{L} \right) \\ &= (-1)^{i+1} \left( R_i \frac{NR_i + NKC_i - C_iP}{L} - C_i \frac{P'R_i + P'KC_i - R_ih + C_i(T - KP')}{L} \right) \\ &= (-1)^{i+1} \left( R_i \frac{NR_i - C_iP'}{L} - C_i \frac{R_i(P' - h) + C_iT}{L} \right) \\ &= (-1)^{i+1} \left( R_i \frac{NR_i + C_i(P'' - P')}{L} - C_i \frac{R_i(P' + P'' - h) + C_iT}{L} \right) \\ &= (-1)^{i+1}(R_iM_1 - C_iM_2) . \end{aligned}$$

Equations (21) and (26) imply that

$$NR_i + C_i(P'' - P') \equiv -NKC_i + C_i(P'' - P') \equiv C_i(P'' - P') \equiv 0 \pmod{L} ,$$

so  $M_1 \in \mathbb{F}_q[x]$ , and (28) implies that  $M_2 \in \mathbb{F}_q[x]$ . Finally, by (13) and (24) we have

$$\begin{aligned} P_{i+1} &= \frac{G_i - Q_{i+1}B_{i-1} + hB_i}{B_i} \\ &= \frac{(-1)^{i+1}(NR_i - P'C_i) - Q_{i+1}B_{i-1} + hB_i}{B_i} \\ &= \frac{NR_i + Q_{i+1}C_{i-1}}{C_i} - P' + h \end{aligned}$$

as required.  $\square$

The NUCUBE algorithm is constructed by aligning the quantities of Algorithm 1 with those in Theorem 4.1, applying the formulas for  $Q_{i+1}$  and  $P_{i+1}$ , and determining what value of  $i$  results in the ideal  $(Q_{i+1}, P_{i+1})$  being reduced. We assume that we want to compute a reduced ideal equivalent to  $\mathfrak{a}^3 = S(Q, P)$  given a reduced ideal  $\mathfrak{a} = (Q', P')$ . As  $\mathfrak{a}$  is reduced, we can assume that  $\deg(Q') \leq g$  and  $\deg(P') < g$ . We also assume that  $R' = (f + hP' - (P')^2)/Q'$  is given.

First, we set

$$(29) \quad \begin{aligned} N &= Q'/S, \\ L &= (Q')^2/S, \\ K &= \begin{cases} R'v_1(2 + v_1(v_1Q'R' - 2P' + h)) & \text{if } S' = 1 \\ R'(u_2v_1Q' + v_2(2P' - h)) & \text{if } S' > 1 \end{cases} \end{aligned}$$

where  $S'$  and  $S$  are defined in (15) and (16). Algorithm 1 shows that  $N, L$ , and  $K$  satisfy the conditions of Theorem 4.1, so the ideals  $(Q_{i+1}, P_{i+1})$  for  $-1 \leq i < n$  are all equivalent to  $\mathfrak{a}^3$ .

It remains to determine the index  $i$  at which the algorithm should terminate in order to ensure that  $(Q_{i+1}, P_{i+1})$  is as close to being reduced as possible. We now show that selecting  $i$  such that

$$\deg R_i \leq \frac{\deg Q' + g}{2} < \deg R_{i-1}$$

satisfies this property.

**Theorem 4.2.** *Given a reduced ideal  $\mathfrak{a} = (Q', P')$  of an imaginary quadratic function field  $\mathbb{F}_q(x, y)$ , set  $N, L$ , and  $K$  as in (29) and compute the ideal  $(Q_{i+1}, P_{i+1})$  as described in Theorem 4.1. If  $i$  is chosen such that  $R_i \leq (\deg Q' + g)/2 < \deg R_{i-1}$ , then  $(Q_{i+1}, P_{i+1})$  is reduced.*

*Proof.* First, observe that from (5), we have

$$(30) \quad \deg B_i \leq \deg \frac{L}{R_{i-1}} = \deg \frac{(Q')^2}{SR_{i-1}}.$$

We now bound the degree of  $Q_{i+1}$ . From its definition in Theorem 4.1 we have

$$\begin{aligned} \deg Q_{i+1} &= \deg \left( R_i \frac{NR_i + C_i(P'' - P')}{L} - C_i \frac{R_i(P'' + P' - h) + C_i T}{L} \right) \\ &= \deg \left( \frac{NR_i^2}{L} - \frac{R_i C_i(2P' - h)}{L} - \frac{C_i^2(f + hP' - (P')^2)}{LN} \right) \\ &= \deg \left( \frac{R_i^2}{Q'} - \frac{SR_i C_i(2P' - h)}{(Q')^2} - \frac{C_i^2 R' S^2}{(Q')^2} \right) \\ &= \max \left\{ \deg \frac{R_i^2}{Q'}, \deg \frac{SR_i C_i(2P' - h)}{(Q')^2}, \deg \frac{C_i^2 R' S^2}{(Q')^2} \right\}. \end{aligned}$$

The condition on  $\deg R_i$  implies

$$\deg \frac{R_i^2}{Q'} \leq 2 \deg R_i - \deg Q' \leq g.$$

Equation (30) implies

$$\begin{aligned} \deg \frac{SR_i C_i (2P' - h)}{(Q')^2} &\leq \deg B_i \frac{SR_i (2P' - h)}{(Q')^2} \\ &\leq \deg \frac{(Q')^2 SR_i (2P' - h)}{SR_{i-1} (Q')^2} \\ &\leq \deg \frac{R_i (2P' - h)}{R_{i-1}} \\ &\leq \deg(2P' - h) - 1 \\ &\leq g , \end{aligned}$$

as well as

$$\begin{aligned} \deg \frac{C_i^2 R' S^2}{(Q')^2} &\leq \deg \frac{(Q')^4 R' S^2}{S^2 R_{i-1}^2 (Q')^2} \\ &\leq \deg \frac{(Q')^2 R'}{R_{i-1}^2} \\ &= \deg Q' R' + \deg Q' - 2 \deg R_{i-1} \\ &< \deg(f + P'h - (P')^2) + \deg Q' - \deg Q' - g \\ &\leq g + 1 . \end{aligned}$$

It follows that  $\deg Q_{i+1} \leq g$ , and that the ideal  $(Q_{i+1}, P_{i+1})$  is reduced.  $\square$

Another issue to deal with before presenting our NUCUBE algorithm is to eliminate the possibility that  $C_i$  is zero when computing  $P_{i+1}$  in Theorem 4.1. The values of  $C_i$  for  $0 \leq i < n$  are computed from the partial quotients  $q_i$  using the recurrence  $C_i = C_{i-2} - q_i C_{i-1}$  with initial values  $C_{-2} = -1$  and  $C_{-1} = 0$ . As the  $q_i$  for  $1 \leq i < n$  are not equal to 0, we see that the only time we can have  $C_i = 0$  is for  $i = -1$ , *i.e.*, when  $R_{-1} = L$  has  $\deg(L) \leq (\deg Q' + g)/2$ . However, in this case the cubing formulas from Section 3 (Algorithm 1) produce a reduced ideal directly, as

$$\deg L = \deg((Q')^2/S) = 2 \deg Q' - \deg S$$

and  $2 \deg Q' - \deg S \leq (\deg Q' + g)/2$  implies that  $3 \deg Q' - 2 \deg S = \deg Q \leq g$ . Thus, if we have  $\deg L \leq (\deg(Q') + g)/2$  we compute  $(Q, P)$  using the cubing formulas from Algorithm 1; otherwise, we use Theorem 4.1.

As with Algorithm 1, the output ideal  $(Q, P)$  is not in general normalized in the sense that  $P$  is not necessarily reduced modulo  $Q$ . In this case, no reduction steps need to be performed, so it is desirable to perform this normalization. Again, this can be done by computing  $q$  and  $r$  such that  $P = qQ + r$ , setting  $R = R + q(P + r - h)$ , and setting  $P = r$ .

Our NUCUBE algorithm is presented in Algorithm 2. As with Algorithm 1, NUCUBE works for an arbitrary quadratic function field  $\mathbb{F}_q(x, y)$ , and similar simplifications can be made in the odd and even characteristic cases. Although Theorem 4.2 only applies to imaginary quadratic function fields, the algorithm will also work for real and unusual quadratic function fields; finding and proving an analogous termination condition that guarantees a reduced output in these cases is the subject of further research.

**Algorithm 2** NUCUBE for Imaginary Quadratic Function Fields**Input:**  $\mathfrak{a} = (Q', P')$ ,  $R' = (f + hP' - (P')^2)/Q'$ **Output:** Reduced ideal  $(Q, P)$  equivalent to  $\mathfrak{a}^3$ , (optional)  $S$  and  $R = (f + hP - P^2)/Q$ 

- 1: Compute  $S' = u_1Q' + v_1(2P' - h)$  (only compute  $S'$  and  $v_1$ ).
- 2: **if**  $S' = 1$  **then**
- 3:   Set  $S = 1$ .
- 4:   Set  $N = Q'$ ,  $L = (Q')^2$ ,  $K = R'v_1(2 + v_1(v_1Q'R' - 2P' + h)) \pmod{L}$ .
- 5: **else**
- 6:   Compute  $S = u_2Q'S' + v_2((2P' - h)^2 + Q'R')$ .
- 7:   Set  $N = Q'/S$ ,  $L = NQ'$ ,  $K = R'(u_2v_1Q' + v_2(2P' - h)) \pmod{L}$ .
- 8: **end if**
- 9: **if**  $\deg L \leq (\deg Q' + g)/2$  **then**
- 10:   Set  $T = NK$ ,  $Q = NL$ , and  $P = P' + T$ .
- 11:   (optional) Set  $R = (SR' - K(2P' - h + T))/L$ .
- 12: **else**
- 13:   Set  $R_{-1} = L$ ,  $R_0 = K$ ,  $C_{-1} = 0$ ,  $C_0 = -1$ ,  $i = 0$ .
- 14:   **while**  $\deg R_i > (\deg(Q') + g)/2$  **do**
- 15:      $i = i + 1$
- 16:     Set  $q_i = \lfloor R_{i-2}/R_{i-1} \rfloor$ ,  $R_i = R_{i-2} - q_iR_{i-1}$ , and  $C_i = C_{i-2} - q_iC_{i-1}$ .
- 17:   **end while**
- 18:   Set  $P'' = P' + NK \pmod{L}$ .
- 19:   Set  $M_1 = (NR_i + (P'' - P')C_i)/L$ .
- 20:   Set  $M_2 = (R_i(P' + P'' - h) + R'S)/L$ .
- 21:   Set  $Q = (-1)^{i-1}(R_iM_1 - C_iM_2)$ .
- 22:   Set  $P = (NR_i + QC_{i-1})/C_i - P' + h$ .
- 23:   (optional) Set  $R = (f + hP - P^2)/Q$ .
- 24: **end if**
- 25: Compute  $q, r \in \mathbb{F}_q[x]$  such that  $P = qQ + r$  (division with remainder).
- 26: Set  $P = r$ .
- 27: (optional) Set  $R = R + q(P + r - h)$ .

## 5. IDEAL CUBING AND NUCUBE IN QUADRATIC NUMBER FIELDS

The quadratic number field  $\mathbb{Q}(\sqrt{\Delta})$  can also be expressed as  $\mathbb{Q}(\sqrt{D})$ , where  $D \in \mathbb{Z}$  is square-free and  $\Delta = (2/r)^2D$  with

$$r = \begin{cases} 1 & \text{if } D \equiv 2, 3 \pmod{4} \\ 2 & \text{if } D \equiv 1 \pmod{4}. \end{cases}$$

If  $D < 0$  we call  $\mathbb{Q}(\sqrt{\Delta})$  an imaginary quadratic field, and if  $D > 0$  we call it a real quadratic field. As shown in [13, Ch. 4], ideals of  $\mathbb{Q}(\sqrt{D})$  can be represented as

$$S(Q, P) = S \left( \frac{Q}{r}\mathbb{Z} + \left( \frac{P + \sqrt{D}}{r} \right)\mathbb{Z} \right)$$

with  $Q, P \in \mathbb{Z}$  and  $Q|(D - P^2)$ . The ideal  $S(Q, P)$  is primitive if  $S = 1$ , written as  $(Q, P)$  for short, and by [13, Theorems 5.6 and 5.9], if  $D < 0$  and  $(Q, P)$  is a primitive ideal with  $Q/r < \sqrt{|\Delta|}/2$ , then  $(Q, P)$  is reduced.

As shown in [13, Ch. 3], these ideals are associated with the quadratic irrationalities  $(P + \sqrt{D})/Q$ , and the results of Section 2 hold verbatim in the number field case using integers instead of polynomials in  $\mathbb{F}_q[x]$ ,  $y = \sqrt{D}$  and  $h = 0$ . Thus, it is a straightforward matter to derive ideal cubing formulas for quadratic number fields by adapting the results of Section 3. The resulting algorithm is presented in Algorithm 3, and as with the function field version works for both imaginary and real quadratic fields. We use the notation from Section 3 and set  $R = (D - P^2)/Q$ .

---

**Algorithm 3** Ideal Cubing for Quadratic Number Fields

---

**Input:**  $\mathfrak{a} = (Q', P')$ ,  $R' = (D - (P')^2)/Q'$

**Output:**  $\mathfrak{a}^3 = S(Q, P)$ , (optional)  $R = (D - P^2)/Q$

- 1: Compute  $S' = u_1(Q'/r) + v_1(2P'/r)$  (only compute  $S'$  and  $v_1$ ).
  - 2: **if**  $S' = 1$  **then**
  - 3:   Set  $S = 1$ .
  - 4:   Set  $N = Q'$ ,  $L = NQ'/r^2$ ,  $K = R'v_1(2 + v_1(v_1(Q'/r)(R'/r) - (2P'/r)) \pmod{L}$ .
  - 5: **else**
  - 6:   Compute  $S = u_2(S'Q'/r) + v_2((3(P')^2 + D)/r^2)$ .
  - 7:   Set  $N = Q'/S$ ,  $L = NQ'/r^2$ ,  $K = R'(u_2v_1(Q'/r) + v_2(2P'/r)) \pmod{L}$ .
  - 8: **end if**
  - 9: Set  $T = NK$ .
  - 10: Set  $Q = NL$ .
  - 11: Set  $P = P' + T$ .
  - 12: (optional) Set  $R = (rSR' - K(2P' + T))/L$ .
- 

As in the function field case, the ideal output by Algorithm 3 is not normalized in the sense that  $P$  is not necessarily reduced modulo  $Q$ . This is done as part of the subsequent reduction process, so it is not necessary perform the normalization if reduction will be immediately applied. If normalization is required, compute  $q'$  and  $r'$  such that  $P = q'Q + r'$ , set  $R = R + q'(P + r')$ , and set  $P = r'$ .

5.1. NUCUBE FOR QUADRATIC NUMBER FIELDS. The NUCUBE algorithm can also be adapted directly. The quadratic number field analogue of Theorem 4.1 is proved in [13, Theorem 5.21], and is the same except that the polynomial coefficients are integers,  $h = 0$ , and  $y$  is replaced by  $\sqrt{D}$ . To derive the number field version of NUCUBE, we take, for  $\mathfrak{a} = (Q', P')$ ,  $S'$  and  $S$  as defined in Algorithm 3, set

$$(31) \quad \begin{aligned} N &= Q'/S, \\ L &= (Q')^2/r^2S, \\ K &= \begin{cases} R'v_1(2 + v_1(v_1(Q'/r)(R'/r) - 2P'/r)) & \text{if } S' = 1 \\ R'(u_2v_1Q'/r + v_2(2P'/r)) & \text{if } S' > 1, \end{cases} \end{aligned}$$

and apply Theorem 5.21 of [13] to obtain an ideal  $(Q_{i+1}, P_{i+1})$  equivalent to  $\mathfrak{a}^3$ . If  $i$  is chosen such that  $R_i < \sqrt{Q'/r^2}|D|^{1/4} < R_{i-1}$ , we can ensure that  $(Q_{i+1}, P_{i+1})$  is at most two reduction steps away from being reduced. Although our paper focuses primarily on the case of imaginary quadratic number and function fields, we prove this result for real quadratic number fields, too, as extending to this case requires only a minor modification.

**Theorem 5.1.** *Given a reduced ideal  $\mathfrak{a} = (Q', P')$  of a quadratic field  $\mathbb{Q}(\sqrt{D})$ , set  $N$ ,  $L$ , and  $K$  as in (31) and compute the ideal  $(Q_{i+1}, P_{i+1})$  as described in [13, Theorem 5.21] (the number field version of Theorem 4.1). If  $i$  is chosen such that  $R_i < \sqrt{Q'/r^2} |D|^{1/4} < R_{i-1}$ , then  $(Q_{i+1}, P_{i+1})$  is at most two reduction steps away from being reduced.*

*Proof.* First, by (12) we have

$$\begin{aligned} |Q_{i+1}| &= |(G_i^2 - DB_i^2)/Q'| \\ &= |(N^2 R_i^2 + (P')^2 C_i^2 - 2NP'R_i C_i - ((P')^2 + R'Q')B_i^2)/((Q')^3/(rS)^2)| \\ &= \left| \frac{(Q')^2 R_i^2 r^2 S^2}{S^2 (Q')^3} + \frac{(P')^2 B_i^2}{Q} - \frac{2Q'P'R_i C_i r^2 S^2}{S(Q')^3} - \frac{(P')^2 B_i^2}{Q} - \frac{R'Q'B_i^2 r^2 S^2}{(Q')^3} \right| \\ &= \left| \frac{R_i^2 r^2}{Q'} - \frac{2P'R_i C_i r^2 S}{(Q')^2} - \frac{R'B_i^2 r^2 S^2}{(Q')^2} \right|. \end{aligned}$$

Put  $x = R_i^2 r^2 / Q'$ ,  $y = 2P'R_i C_i r^2 S / (Q')^2$ , and  $z = R'B_i^2 r^2 S^2 / (Q')^2$ . Then we have

$$0 \leq x = R_i^2 \frac{r^2}{Q'} \leq \frac{Q' \sqrt{|D|} r^2}{r^2 Q'} = \sqrt{|D|}$$

From (5) it follows that  $B_i R_{i-1} \leq L = (Q')^2 / r^2 S$  and

$$B_i = |C_i| \leq \frac{(Q')^2}{r^2 S} \frac{1}{R_{i-1}} \leq \frac{(Q')^2}{r^2 S} \frac{r}{\sqrt{Q'} |D|^{1/4}} = \frac{(Q')^{3/2}}{rS |D|^{1/4}}.$$

Thus, we obtain

$$|y| = \left| \frac{2P'R_i C_i r^2 S}{(Q')^2} \right| \leq \left| \frac{2P'R_i r^2 S}{(Q')^2} \frac{(Q')^2}{r^2 S R_{i-1}} \right| \leq 2|P'|.$$

Now assume  $D < 0$ . In this case we have  $R' = -\frac{|D| + (P')^2}{Q'} < 0$  and therefore

$$0 \leq -z = -\frac{R'B_i^2 r^2 S^2}{(Q')^2} \leq -\frac{Q'R'}{\sqrt{|D|}} = \frac{|D| + (P')^2}{\sqrt{|D|}} = \sqrt{|D|} + \frac{(P')^2}{\sqrt{|D|}}.$$

Since  $\mathfrak{a}$  is reduced, we have  $Q' \leq 2\sqrt{|D|/3}$  and  $P' \leq \sqrt{|D|/3}$  and it follows that

$$2|P'| \leq x - y - z \leq \sqrt{|D|} + 2|P'| + \sqrt{|D|} + \frac{(P')^2}{\sqrt{|D|}} \leq 3.5\sqrt{|D|}.$$

As argued in [13, p.122] this implies that the ideal  $(Q_{i+1}, P_{i+1})$  is reduced after at most two reduction steps.

Now let  $D > 0$ . In this case we have  $R' = \frac{D - (P')^2}{Q'} > 0$  and

$$0 \leq z = \frac{R'B_i^2 r^2 S^2}{(Q')^2} \leq \frac{R'r^2 S^2}{(Q')^2} \frac{(Q')^3}{r^2 S^2 \sqrt{D}} = \frac{Q'R'}{\sqrt{D}} = \sqrt{D} - \frac{(P')^2}{\sqrt{D}}.$$

Putting the bounds on  $x$ ,  $y$ , and  $z$  together yields

$$-2|P'| - \sqrt{D} \leq x - y - z \leq \sqrt{D} + 2|P'| - \sqrt{D} + \frac{(P')^2}{\sqrt{D}} \leq \sqrt{D} + 2|P'|,$$

which implies

$$|Q_{i+1}| \leq \sqrt{D} + 2|P'| \leq 3\sqrt{D}.$$

By [13, Theorem 5.13],  $(Q_{i+1}, P_{i+1})$  is reduced after at most one reduction step.  $\square$

Again, we need to eliminate the possibility that  $C_i$  is zero when computing  $P_{i+1}$ . As with the function field case, this can only happen for  $i = -1$ , *i.e.*, when  $R_{-1} = L < \sqrt{Q'/r^2} |D|^{1/4}$ . However, in this case, the cubing formulas from Algorithm 3 produce a reduced ideal directly, as the bound on  $L$  implies that  $Q/r < \sqrt{\Delta}/2$ , meaning that the ideal  $(Q, P)$  is reduced.

Again as before, the ideal  $(Q, P)$  is not normalized in general. If it is not reduced, the normalization will occur as part of the subsequent reduction process. If it is already reduced, then normalization can be performed as described above after Algorithm 3.

Our NUCUBE algorithm for quadratic fields is presented in Algorithm 4. As with Algorithm 3, NUCUBE works for both real and imaginary quadratic fields.

---

**Algorithm 4** NUCUBE for Quadratic Number Fields

---

**Input:**  $\mathfrak{a} = (Q', P')$ ,  $R' = (D - (P')^2)/Q'$

**Output:** Almost reduced ideal  $(Q, P)$  equivalent to  $\mathfrak{a}^3$ , (optional)  $S$  and  $R = (D - P^2)/Q$

- 1: Compute  $S' = u_1(Q'/r) + v_1(2P'/r)$  (only compute  $S'$  and  $v_1$ ).
  - 2: **if**  $S' = 1$  **then**
  - 3:   Set  $S = 1$ .
  - 4:   Set  $N = Q'$ ,  $L = (Q'/r)^2$ ,  $K = R'v_1(2 + v_1(v_1(Q'/r)(R'/r) - (2P'/r)) \pmod{L}$ .
  - 5: **else**
  - 6:   Compute  $S = u_2(S'Q'/r) + v_2((3(P')^2 + D)/r^2)$ .
  - 7:   Set  $N = Q'/S$ ,  $L = NQ'/r^2 = (Q')^2/r^2S$ ,  $K = R'(u_2v_1(Q'/r) + v_2(2P'/r)) \pmod{L}$ .
  - 8: **end if**
  - 9: **if**  $L < \sqrt{Q'/r^2} |D|^{1/4}$  **then**
  - 10:   Set  $T = NK$ ,  $Q = NL$ , and  $P = P' + T$ .
  - 11:   (optional) Set  $R = (rSR' - K(2P' + T))/L$ .
  - 12: **else**
  - 13:   Set  $R_{-1} = L$ ,  $R_0 = K$ ,  $C_{-1} = 0$ ,  $C_0 = -1$ ,  $i = 0$ .
  - 14:   **while**  $R_i > \sqrt{Q'/r^2} |D|^{1/4}$  **do**
  - 15:      $i = i + 1$
  - 16:     Set  $q_i = \lfloor R_{i-2}/R_{i-1} \rfloor$ ,  $R_i = R_{i-2} - q_i R_{i-1}$ , and  $C_i = C_{i-2} - q_i C_{i-1}$ .
  - 17:   **end while**
  - 18:   Set  $P'' = P' + NK \pmod{L}$ .
  - 19:   Set  $M_1 = (NR_i + (P'' - P')C_i)/L$ .
  - 20:   Set  $M_2 = (R_i(P' + P'') + R'S)/L$ .
  - 21:   Set  $Q = (-1)^{i-1}(R_i M_1 - C_i M_2)$ .
  - 22:   Set  $P = (NR_i + QC_{i-1})/C_i - P'$ .
  - 23:   (optional) Set  $R = (D - P^2)/Q$ .
  - 24: **end if**
  - 25: Compute  $q', r' \in \mathbb{Z}$  such that  $P = q'Q + r'$  (division with remainder).
  - 26: (optional) Set  $R = R + q'(P + r')$ .
  - 27: Set  $P = r'$ .
- 

It is important to note that, as with the NUCOMP algorithm [13, Ch. 5], the partial extended Euclidean algorithm step (Steps 14–17) must be implemented carefully in order to get the best performance out of this algorithm. In particular, using

an adaptation of Lehmer's algorithm [15, 14] significantly speeds the algorithm; this is done in our implementations described in Section 6.

5.2. ALTERNATIVE IDEAL REPRESENTATION. Another common representation of ideals in quadratic number fields used, for example, in [3], is  $\mathfrak{a} = (a, b) = a\mathbb{Z} + (b + \sqrt{\Delta})/2\mathbb{Z}$ , where  $c = (b^2 - \Delta)/(4a) \in \mathbb{Z}$ . This representation is sometimes preferred as it gives an immediate correspondence to the binary quadratic form  $aX^2 + bXY + cY^2$ . Thus, algorithms for performing arithmetic with ideals in this representation can be used directly for the corresponding operations on binary quadratic forms.

Algorithms using the  $(Q, P)$  representation can be translated by setting  $Q = ra$ ,  $P = (r/2)b$ , and  $R = -rc$ . The resulting versions of Algorithm 3 and Algorithm 4 are listed in the Appendix.

## 6. NUMERICAL RESULTS

We performed our experiments on a 2.4 GHz Intel Core 2 Duo MacBook running Mac OS X version 10.5.7. We used NTL [18] for finite field and polynomial arithmetic, with GMP as the base layer for large integer arithmetic. We compiled our programs with the GNU C++ compiler version 4.0.1.

In order to test the efficiency of our cubing formulas, we implemented several exponentiation algorithms that take advantage of fast cubing operations, namely, the left-to-right and right-to-left double-base chain methods. For validity checking and comparison purposes, we also implemented the classical square-and-multiply (aka binary) and NAF algorithms. For each of those, we considered both the NUCOMP and non-NUCOMP approaches. However, we did not try to mix NUCOMP and non-NUCOMP primitives within the same exponentiation routine. In total, this represents eight different exponentiation algorithms. We considered the most general situation, which occurs for example in the second stage of the Diffie-Hellman key exchange agreement, where both the exponent  $n$  and the base, in our case a reduced ideal, are not known in advance. Our timings represent the average time taken by each algorithm over 100 random exponentiations. For fair comparison we used the same values (fields, ideals, exponents) for every method.

Given  $n > 0$ , a double-base chain computing  $n$  is an expansion of the form

$$(32) \quad n = \sum_{i=1}^{\ell} 2^{a_i} 3^{b_i}, \quad \text{with } a_i, b_i \geq 0,$$

such that the sequence  $(a_i, b_i)_{i>0}$  decreases for the product order. Both the right-to-left and left-to-right exponentiation algorithms consist of finding a double-base chain representing the exponent and allowing for the exponentiation to be performed à la Horner.

The right-to-left algorithm, denoted db-Rl in the following, starts by dividing  $n$  by 3 and by 2 as much as possible. Then, adding or subtracting 1 leads to a multiple of 3, and one can repeat the process until reaching 0. The left-to-right algorithm, called db-Lr, is based on a greedy approach: find the closest integer to  $n$  of the form  $2^a 3^b$ , subtract and continue until reaching 0. In order to find the closest  $\{2, 3\}$ -integer<sup>1</sup> from  $n$ , we implemented the fast algorithm proposed in [2] by Berthé and Imbert.

The efficiency of the left-to-right approach is very dependent upon several parameters, namely the relative costs between a multiplication, a squaring, and a cubing,

<sup>1</sup>A  $S$ -integer is an integer whose prime factors all belong to  $S$ .

and the length of the double-base chain used to perform the exponentiation. In order to get optimal performance, an upper bound on the summands in (32) must be set such that  $2^{a_i}3^{b_i} \leq 2^A3^B$  for  $1 \leq i \leq \ell$ . For our experiments, we measured the time over 100 random exponentiations for  $B$  ranging from 0 (a signed binary expansion) to  $\lceil \log_3 n \rceil$ , with  $A$  set accordingly such that  $2^A3^B$  is the smallest number of that form greater than or equal to  $n$ . The timings we report for db-Lr in the next sections correspond to the minimum time obtained over all those tested bounds. (Check the authors' personal web pages for progress and up-to-date results.)

6.1. NUMBER FIELDS. We considered random imaginary fields  $\mathbb{Q}(\sqrt{\Delta})$  with  $\Delta < 0$  of size ranging from 32 to 8192 bits, with random exponent of size half the size of  $\Delta$ . For each algorithm, the average time for one exponentiation (in ms) is given in Table 1. In Table 2, we report timings for discriminants and exponents of cryptographic relevant sizes. The discriminant sizes were selected to correspond to NIST's five recommended security levels [16] as described in [12]. In both tables, the fastest results are highlighted. The prefix "N" in the algorithm designations denotes the NUCOMP version of the corresponding exponentiation algorithm.

TABLE 1. Exponentiation timings (in ms) for random imaginary number fields  $\mathbb{Q}(\sqrt{\Delta})$  of size ranging from 32 to 8192 bits, and random exponents of size  $\text{size}(\Delta)/2$  bits

	size of $\Delta$ (in bits)								
	32	64	128	256	512	1024	2048	4096	8192
bin	0.12	0.32	0.98	3.64	14.77	62.24	292.46	1616.19	9786.51
naf	0.11	0.29	0.87	3.23	13.09	55.77	260.28	1444.94	8703.83
db-Rl	0.10	0.28	0.89	3.31	13.45	58.36	282.38	1582.18	9842.75
db-Lr	0.11	0.29	0.90	3.27	13.18	55.94	261.49	1416.15	8626.10
Nbin	0.22	0.46	1.05	3.02	9.48	30.55	115.81	505.61	2521.85
Nnaf	0.20	0.42	0.94	2.68	8.41	27.10	102.81	448.11	2222.31
Ndb-Rl	0.14	0.31	0.74	2.28	7.31	25.64	102.76	477.98	2484.81
Ndb-Lr	0.13	0.29	0.75	2.33	7.33	25.57	100.72	453.13	2237.36

TABLE 2. Exponentiation timings (in ms) for imaginary number fields  $\mathbb{Q}(\sqrt{\Delta})$  and exponents of cryptographic relevant sizes

	size of $\Delta$ / exponent size (in bits)				
	795 / 160	1384 / 224	1732 / 256	3460 / 384	5704 / 512
bin	14.85	38.76	59.00	232.15	673.87
naf	13.38	34.44	52.42	207.57	597.91
db-Rl	13.58	36.45	56.23	227.39	671.40
db-Lr	13.12	34.32	52.20	204.66	592.06
Nbin	8.15	17.93	25.17	78.70	196.81
Nnaf	7.24	15.93	22.32	69.83	174.30
Ndb-Rl	6.54	15.01	21.62	72.89	188.68
Ndb-Lr	6.53	14.86	21.20	69.77	174.19

For discriminants of size less than 4096 bits, the double-base algorithms, either right-to-left or left-to-right, are faster than the binary approaches (NAF and square-multiply). Not surprisingly, the NUCOMP versions become more interesting for

multiple precision operands ( $\Delta \geq 2^{128}$ ). They are twice as fast than the non-NUCOMP versions for 1024-bit discriminants, and almost three times faster for discriminants of size 2048 bits. For bigger discriminants ( $\text{size}(\Delta) \geq 4096$ ), we remark that Nnaf is the best choice. In these cases, our ideal cubing is more expensive than a squaring followed by a multiplication and the best timings for Ndb-Lr are obtained for double-base chains where the largest power of 3 is equal or very close to 0. These chains therefore correspond to signed binary expansions. Such representations can be computed faster using the NAF recoding algorithm, which explains the results for large discriminants. For cryptographic relevant sizes however, the exponents are much smaller and Ndb-Lr is always faster.

**6.2. FUNCTION FIELDS.** We ran numerous experiments using random imaginary function fields of genus  $g$  ranging from 2 to 20 over finite fields  $\mathbb{F}_q$  of size ranging from 4 to 512 bits. We restricted ourselves to the most common cases where  $q$  is either an odd prime or a power of 2. For our random experiments, we used exponents of size approximately  $g \times \log_2 q$  bits. In Tables 3 and 4, we report timings for function fields of various genera, defined over various finite fields of odd and even characteristic, respectively. We do not give the timings for the binary algorithms, which are naturally always slower than the NAF variants.

We also considered cryptographically-relevant function fields of small genus  $g = 2, 3, 4$  over finite fields of various sizes selected to provide 80, 112, 128, 192, and 256 bits of security, using random exponents of sizes 160, 224, 256, 384, and 512 bits, respectively. As described in [10], our parameters' sizes are based on the best-known attack on the DLP for small genus [8]. The results are given in Tables 5 and Table 6.

The behavior for function fields is very similar to the number field case, with double-base algorithms giving good results up to the point where the cube operation becomes more expensive than the combination square/multiplication. In particular, double-base methods offer an interesting alternative to the classical exponentiation algorithms for cryptographic applications.

We summarize all our experiments for random imaginary quadratic function fields in Tables 7 and 8, where we give the fastest exponentiation algorithm for  $g$  ranging from 2 to 20 and  $q$  ranging from 4 to 512 bits.

TABLE 3. Timings (in ms) for one exponentiation for function fields of genus  $g = 2, 3, 7,$  and  $20$  defined over  $\mathbb{F}_p$  with  $p$  prime, and exponents of sizes  $\simeq g \log_2 p$

	size of $p$ / exponent size (in bits)					
<b>Genus 2</b>	16/31	32/64	64/127	128/255	256/511	512/1024
naf	0.36	0.81	6.28	16.18	46.67	155.67
db-Rl	0.33	0.75	5.89	14.88	43.77	150.70
db-Lr	0.34	0.77	5.88	14.70	43.12	147.31
Nnaf	0.44	0.94	6.98	17.41	47.88	151.15
Ndb-Rl	0.39	0.86	6.44	16.05	46.26	152.63
Ndb-Lr	0.39	0.88	6.45	16.02	45.77	149.31
<b>Genus 3</b>	16/46	32/96	64/190	128/383	256/767	512/1535
naf	0.68	1.62	13.67	35.41	106.87	367.73
db-Rl	0.64	1.50	13.10	33.13	100.24	368.00
db-Lr	0.65	1.52	12.93	32.66	98.64	356.94
Nnaf	0.74	1.66	13.03	33.23	96.32	316.00
Ndb-Rl	0.69	1.58	12.77	32.24	95.89	332.92
Ndb-Lr	0.70	1.60	12.58	31.85	94.42	322.53
<b>Genus 7</b>	16/106	32/223	64/442	128/892	256/1788	512/3582
naf	3.11	7.56	81.78	206.31	656.24	2515.03
db-Rl	2.92	7.14	83.65	208.44	720.47	2610.04
db-Lr	2.89	7.13	80.75	202.85	666.21	2515.15
Nnaf	2.93	7.00	69.26	175.04	551.29	2157.34
Ndb-Rl	2.73	6.61	70.49	177.70	586.55	2279.67
Ndb-Lr	2.71	6.67	68.32	173.81	560.69	2183.60
<b>Genus 20</b>	16/302	32/636	64/1263	128/2548	256/5108	512/10232
naf	30.40	72.58	1134.77	3195.74	9687.28	34519.20
db-Rl	32.17	76.40	1294.09	3555.40	10626.89	37497.01
db-Lr	30.54	73.40	1144.94	3245.20	9700.58	34609.60
Nnaf	22.93	56.35	787.01	2105.68	7439.68	25737.79
Ndb-Rl	23.90	59.20	915.89	2579.84	8378.98	28828.97
Ndb-Lr	23.07	57.27	802.53	2123.23	7483.74	25813.70

TABLE 4. Timings (in ms) for one exponentiation for function fields of genus  $g = 2, 7,$  and  $12$  defined over finite fields  $\mathbb{F}_{2^m}$ , with exponents of sizes  $g \times m$

	size of $q$ / exponent size (in bits)					
<b>Genus 2</b>	16/32	32/64	64/128	128/256	256/512	512/1024
naf	1.55	3.75	10.25	18.18	61.27	257.63
db-Rl	1.45	3.63	10.10	16.89	58.28	243.79
db-Lr	1.46	3.61	9.98	16.79	57.78	239.95
Nnaf	1.83	4.29	11.31	19.48	62.23	251.28
Ndb-Rl	1.74	4.24	11.41	19.11	64.27	263.73
Ndb-Lr	1.74	4.15	11.14	18.73	62.96	254.47
<b>Genus 7</b>	16/112	32/224	64/448	128/896	256/1792	512/3584
naf	18.20	48.53	150.61	235.72	880.02	4107.18
db-Rl	18.15	49.16	156.55	245.45	930.15	4559.18
db-Lr	17.73	47.83	150.58	237.15	886.37	4119.82
Nnaf	17.98	47.50	143.29	218.41	833.95	3848.70
Ndb-Rl	18.05	48.73	151.78	235.87	882.02	4167.07
Ndb-Lr	17.55	47.14	142.84	223.25	843.72	3861.96
<b>Genus 12</b>	16/192	32/384	64/768	128/1536	256/3072	512/6144
naf	61.75	168.62	531.15	810.32	3307.53	16859.27
db-Rl	63.90	177.20	571.08	889.33	3710.25	19120.08
db-Lr	62.93	169.40	537.03	812.97	3313.26	16827.14
Nnaf	54.76	149.82	468.65	726.72	2812.44	13405.59
Ndb-Lr	57.35	159.86	514.94	799.22	3208.80	15688.03
Ndb-Rl	55.09	150.59	474.98	733.58	2819.55	13402.69

TABLE 5. Timings (in ms) for cryptographic relevant function fields of genus 2, 3, and 4, defined over  $\mathbb{F}_p$  for some primes  $p$  chosen to provide 80, 112, 128, 192 and 256 bits of security, with exponent ranging from 160 to 512 bits

	size of $q$ / exponent size				
<b>Genus 2</b>	80/160	112/224	128/256	192/384	256/512
naf	9.17	13.60	16.20	30.39	46.58
db-Rl	8.52	12.58	14.89	28.29	43.92
db-Lr	8.46	12.41	14.80	27.86	43.43
Nnaf	10.12	14.79	17.44	31.93	47.83
Ndb-Rl	9.26	13.65	16.04	30.22	46.40
Ndb-Lr	9.21	13.52	15.99	29.71	45.82
<b>Genus 3</b>	60/160	84/224	96/256	144/384	192/512
naf	11.44	19.10	22.13	42.35	61.58
db-Rl	10.82	17.96	20.75	39.95	57.19
db-Lr	10.70	17.71	20.49	39.16	56.12
Nnaf	10.99	18.02	20.84	39.15	56.50
Ndb-Rl	10.62	17.51	20.23	38.66	55.05
Ndb-Lr	11.13	18.59	21.58	41.24	59.67
<b>Genus 4</b>	54/160	75/224	86/256	128/384	171/512
naf	14.59	24.32	28.15	45.91	76.70
db-Rl	14.29	23.88	27.51	44.23	74.37
db-Lr	14.02	23.31	26.97	43.21	72.50
Nnaf	13.79	22.70	26.10	42.40	69.98
Ndb-Rl	13.46	22.41	25.79	41.51	69.45
Ndb-Lr	13.24	21.99	25.32	40.63	67.48

TABLE 6. Timings (in ms) for cryptographic relevant function fields of genus 2, 3, and 4, defined over binary fields  $\mathbb{F}_{2^m}$  of size chosen to provide 80, 112, 128, 192 and 256 bits of security, with exponent ranging from 160 to 512 bits

	size of $q$ / exponent size				
<b>Genus 2</b>	83/160	113/224	131/256	193/384	257/512
naf	10.23	15.85	35.17	48.37	80.97
db-Rl	9.59	14.86	34.98	46.92	79.92
db-Lr	9.58	14.77	34.29	46.15	78.81
Nnaf	11.38	17.22	37.31	50.52	82.85
Ndb-Rl	10.99	16.83	38.76	52.06	87.41
Ndb-Lr	10.84	16.54	37.43	50.41	85.01
<b>Genus 3</b>	61/160	89/224	97/256	149/384	193/512
naf	19.60	21.45	24.67	90.84	100.80
db-Rl	19.48	20.72	24.01	91.91	100.91
db-Lr	19.24	20.29	23.55	89.75	98.95
Nnaf	19.88	21.77	25.05	88.41	96.78
Ndb-Rl	20.65	21.87	25.24	94.71	102.91
Ndb-Lr	19.95	21.23	24.48	88.68	98.26
<b>Genus 4</b>	59/160	79/224	89/256	131/384	173/512
naf	25.83	27.01	31.73	116.19	94.68
db-Rl	26.41	27.18	31.69	120.93	94.37
db-Lr	25.85	26.39	30.91	116.44	91.92
Nnaf	25.87	26.97	31.67	113.70	93.35
Ndb-Rl	27.36	28.08	32.74	124.06	97.89
Ndb-Lr	26.13	26.89	31.61	113.93	93.19

TABLE 7. Fastest algorithms for function fields defined over finite fields  $\mathbb{F}_p$  with  $p$  prime

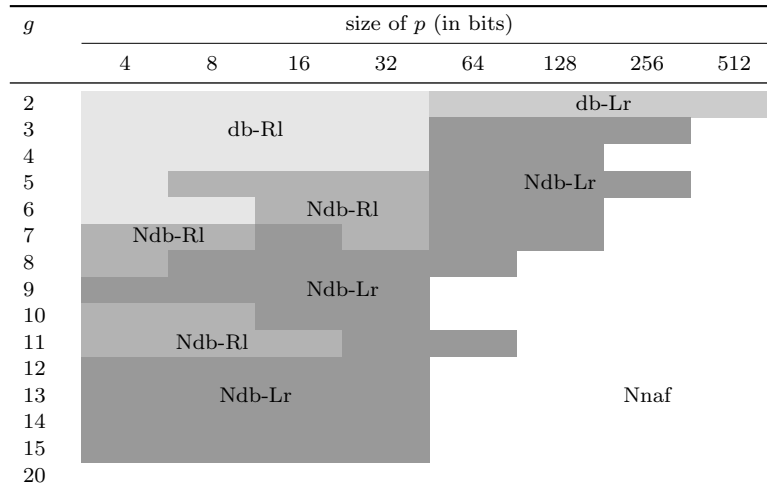


TABLE 8. Fastest algorithms for function fields defined over finite fields  $\mathbb{F}_{2^m}$  of even characteristic

$g$	$m$							
	4	8	16	32	64	128	256	512
2	db-Rl							
3								
4	db-Lr							
5								
6	Ndb-Lr							
7								
8	Nnaf							
9								
10								
11								
12								
13								
14								
15								
20								

## 7. CONCLUSIONS AND OPEN PROBLEMS

One natural question to ask is whether there is any advantage to be gained by exploring specialized formulas for higher powers, for example, quintupling. Our specialized cubing formulas require one application of the extended Euclidean algorithm fewer than using a squaring and a multiplication. It is conceivable that higher powers might save even more, for example, requiring only one application in many cases. Deriving such specialized formulas seems to be a difficult process, but it may be worth investigating.

Our algorithms in the function field case are all generic in the sense that they work for any genus and any finite field. The most interesting cases for cryptographic applications are genus 2 and, to a somewhat lesser extent, 3. In those cases, explicit formulas for ideal arithmetic have been developed that are significantly faster than the generic formulas. Explicit formulas for ideal cubing could also be developed in order to investigate the performance of double base number system exponentiation in these low genus cases.

It would also be interesting to investigate double base infrastructure operations using our cubing algorithms for real quadratic number and function fields. The formulas presented in this paper all work in the real setting as presented, but in the function field case it remains to devise a bound for the NUCUBE algorithm that guarantees that the output is reduced or close to reduced. Infrastructure applications also require an explicit representation of the relative generator  $\theta_{i+1}$  such that  $\mathfrak{a}_{i+1} = (\bar{\theta}_{i+1})\mathfrak{a}^3$ , but it is well-known how to compute this without having to compute the non-reduced  $\mathfrak{a}^3$  (see [13] and [11]). These topics are the subject of further research.

## REFERENCES

- [1] R. Avanzi, M. J. Jacobson, Jr., and R. Scheidler, *Efficient divisor reduction on hyperelliptic curves*, Submitted to Advances in Mathematics of Communications, 2009.

- [2] V. Berthé and L. Imbert, *Diophantine approximation, Ostrowski numeration and the double-base number system*, Discrete Mathematics & Theoretical Computer Science **11** (2009), no. 1, 153–172.
- [3] J. Buchmann and U. Vollmer, *Binary quadratic forms: An algorithmic approach*, Algorithms and Computation in Mathematics, vol. 20, Springer-Verlag, Berlin, 2007.
- [4] D. G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Math. Comp. **48** (1987), no. 177, 95–101.
- [5] V. S. Dimitrov, L. Imbert, and P. K. Mishra, *The double-base number system and its application to elliptic curve cryptography*, Math. Comp. **77** (2008), no. 262, 1075–1104.
- [6] C. Doche and L. Imbert, *Extended double-base number system with applications to elliptic curve cryptography*, Progress in Cryptology - INDOCRYPT 2006, Lecture Notes in Computer Science, vol. 4329, Springer, 2006, 267–302.
- [7] C. Doche and T. Lange, *Arithmetic of elliptic curves*, Handbook of Elliptic and Hyperelliptic Curve Cryptography (H. Cohen and G. Frey, eds.), Chapman & Hall/CRC, 2006, 267–302.
- [8] P. Gaudry, E. Thomé, N. Thériault, and C. Diem, *A double large prime variation for small genus hyperelliptic index calculus*, Math. Comp. **76** (2007), no. 257, 475–492.
- [9] M. J. Jacobson, Jr., R. E. Sawilla, and H. C. Williams, *Efficient ideal reduction in quadratic fields*, International Journal of Mathematics and Computer Science **1** (2006), 83–116.
- [10] M. J. Jacobson, Jr., R. Scheidler, and A. Stein, *Cryptographic protocols on real and imaginary hyperelliptic curves*, Advances in Mathematics of Communications **1** (2007), no. 2, 197–221.
- [11] M. J. Jacobson, Jr., R. Scheidler, and A. Stein, *Fast arithmetic on hyperelliptic curves via continued fraction expansions*, Advances in Coding Theory and Cryptology (T. Shaska, W.C. Huffman, D. Joyner, and V. Ustimenko, eds.), Series on Coding Theory and Cryptology, vol. 3, World Scientific Publishing, 2007, Invited Paper, 201–244.
- [12] M. J. Jacobson, Jr., R. Scheidler, and H. C. Williams, *An improved real quadratic field based key exchange procedure*, Journal of Cryptology **19** (2006), 211–239.
- [13] M. J. Jacobson, Jr. and H. C. Williams, *Solving the Pell equation*, CMS Books in Mathematics, Springer-Verlag, 2009, ISBN 978-0-387-84922-5.
- [14] T. Jebelean, *A double-digit Lehmer-Euclid algorithm for finding the GCD of long integers*, J. Symbolic Computation **19** (1995), no. 1-3, 145–157.
- [15] D. H. Lehmer, *Euclid’s algorithm for large numbers*, The American Mathematical Monthly **45** (1938), no. 4, 227–233.
- [16] National Institute of Standards and Technology (NIST), *Recommendation for key management - part 1: General (revised)*, NIST Special Publication 800-57, March, 2007, See: <http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1.3-8-07.pdf>.
- [17] D. Shanks, *On Gauss and composition I, II*, Proc. NATO ASI on Number Theory and Applications (R.A. Mollin, ed.), Kluwer Academic Press, 1989, 163–179.
- [18] V. Shoup. *NTL: A library for doing number theory*, <http://www.shoup.net/ntl>.
- [19] A. V. Sutherland, “Order computations in generic groups”, Ph.D. thesis, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

APPENDIX A. ALGORITHMS USING  $(a, b)$  REPRESENTATION OF IDEALS IN  
QUADRATIC NUMBER FIELDS

We list here the versions of Algorithm 3 and Algorithm 4 for cubing ideals in number fields using the  $(a, b)$  representation of ideals.

---

**Algorithm 5** Ideal Cubing for Quadratic Number Fields,  $(a, b)$  Representation

---

**Input:**  $\mathfrak{a} = (a', b')$ ,  $c' = ((b')^2 - \Delta)/(4a')$   
**Output:**  $\mathfrak{a}^3 = S(a, b)$ , (optional)  $c = (b^2 - \Delta)/(4a)$   
 1: Compute  $S' = u_1 a' + v_1 b'$  (only compute  $S'$  and  $v_1$ ).  
 2: **if**  $S' = 1$  **then**  
 3:   Set  $S = 1$ .  
 4:   Set  $N = a'$ ,  $L = (a')^2$ ,  $K = c' v_1 (2 - v_1 (b' - v_1 a' c')) \pmod{L}$ .  
 5: **else**  
 6:   Compute  $S = u_2 (S' a') + v_2 ((b')^2 - ac)$ .  
 7:   Set  $N = a'/S$ ,  $L = Na'$ ,  $K = c' (u_2 v_1 a' + v_2 b') \pmod{L}$ .  
 8: **end if**  
 9: Set  $T = NK$ .  
 10: Set  $a = NL$ .  
 11: Set  $b = b' - 2T$ .  
 12: (optional) Set  $c = (Sc + K(T - b)) / L$ .

---



---

**Algorithm 6** NUCUBE for Quadratic Number Fields,  $(a, b)$  Representation

---

**Input:**  $\mathfrak{a} = (a', b')$ ,  $c' = ((b')^2 - \Delta)/(4a')$   
**Output:** Almost reduced ideal  $(a, b)$  equivalent to  $\mathfrak{a}^3$ , (optional)  $S$  and  $c = (b^2 - \Delta)/(4a)$   
 1: Compute  $S' = u_1(a') + v_1(b')$  (only compute  $S'$  and  $v_1$ ).  
 2: **if**  $S' = 1$  **then**  
 3:   Set  $S = 1$ .  
 4:   Set  $N = a'$ ,  $L = (a')^2$ ,  $K = c' v_1 (2 - v_1 (b' - v_1 a' c')) \pmod{L}$ .  
 5: **else**  
 6:   Compute  $S = u_2 (S' a') + v_2 ((b')^2 - ac)$ .  
 7:   Set  $N = a'/S$ ,  $L = Na'$ ,  $K = c' (u_2 v_1 a' + v_2 b') \pmod{L}$ .  
 8: **end if**  
 9: **if**  $L < \sqrt{a'/2} |\Delta|^{1/4}$  **then**  
 10:   Set  $T = NK$ ,  $a = NL$ , and  $b = b' - 2T$ .  
 11:   (optional) Set  $c = (Sc + K(T - b)) / L$ .  
 12: **else**  
 13:   Set  $R_{-1} = L$ ,  $R_0 = K$ ,  $C_{-1} = 0$ ,  $C_0 = -1$ ,  $i = 0$ .  
 14:   **while**  $R_i > \sqrt{a'/2} |\Delta|^{1/4}$  **do**  
 15:      $i = i + 1$   
 16:     Set  $q_i = \lfloor R_{i-2}/R_{i-1} \rfloor$ ,  $R_i = R_{i-2} - q_i R_{i-1}$ , and  $C_i = C_{i-2} - q_i C_{i-1}$ .  
 17:   **end while**  
 18:   Set  $b'' = b' - 2NK \pmod{L}$ .  
 19:   Set  $M_1 = (NR_i + (b'' - b')C_i)/L$ .  
 20:   Set  $M_2 = (R_i(b' + b'') + c'S)/L$ .  
 21:   Set  $a = (-1)^{i-1} (R_i M_1 - C_i M_2)$ .  
 22:   Set  $b = (NR_i + aC_{i-1})/C_i - b'$ .  
 23:   (optional) Set  $c = (b^2 - \Delta)/(4a)$ .  
 24: **end if**  
 25: Compute  $q', r' \in \mathbb{Z}$  such that  $b = q'(2a) + r'$  (division with remainder).  
 26: (optional) Set  $c = c - q'(b + r')/2$ .  
 27: Set  $b = r'$ .

---