

**Abstract.** In this paper, we discuss issues related to the efficient implementation of Shanks' NUCOMP algorithm for computing the reduced composite of two binary quadratic forms. In particular, we describe how efficient versions of NUCOMP can be implemented for computations in imaginary quadratic number and function fields, as well as computations in the infrastructure of real quadratic number and function fields. We present numerous computations clearly illustrating the efficiency of NUCOMP in each of these settings.

# Computational Aspects of NUCOMP

M. J. Jacobson, Jr. and A. J. van der Poorten

Department of Computer Science, University of Manitoba, Winnipeg, MB, R3T 2N2  
ceNTRe for Number Theory Research, Macquarie University, Sydney 2109, Australia  
jacobson@cs.umanitoba.ca (Mike Jacobson)  
alf@math.mq.edu.au (Alf van der Poorten)

## 1 Introduction

In 1989, Shanks introduced the NUCOMP algorithm [10] for computing the reduced composite of two positive definite binary quadratic forms of discriminant  $\Delta$ . Essentially by applying reduction before composing the two forms, the intermediate operands are reduced from size  $O(\Delta)$  to  $O(\Delta^{1/2})$  in most cases and at worst to  $O(\Delta^{3/4})$ . Shanks made use of this to extend the capabilities of his hand-held calculator to computations involving forms with discriminants with as many as 20 decimal digits, even though his calculator had only some 10 digits precision. Improvements by Atkin (described in [3], [4]) have also made NUCOMP very effective for computations with forms of larger discriminant.

Although there is nothing in Shanks' original description which suggests that NUCOMP is only applicable to positive definite forms, for years there were no documented applications in any other setting. Recently, van der Poorten [12] has shown that, with very little extra effort, NUCOMP can also be applied to computations in the infrastructure involving indefinite binary quadratic forms. This opens the door to practical improvements in real quadratic field-based applications such as regulator computation and key exchange protocols in the infrastructure.

Until now NUCOMP has been applied exclusively to computations in number fields. However, Cantor's algorithm [2] for adding reduced divisors on hyperelliptic curves (equivalently ideal multiplication in function fields) is virtually identical to the composition and reduction algorithms for binary quadratic forms; the main difference being that the coefficients of the binary quadratic forms are polynomials over a finite field rather than integers. Thus, there is no reason to believe that NUCOMP cannot also be applied in function fields. Intuitively, one would expect that applying NUCOMP will reduce the degrees of the intermediate operands from  $O(2g)$  to at most  $O(3g/2)$ , where  $g$  is the genus of the hyperelliptic curve or function field. Furthermore, by combining van der Poorten's ideas [12] we can also apply NUCOMP to computations in the infrastructure of a real quadratic function field.

---

\* The first author acknowledges the assistance of an Australian Research Council International Research Exchange Grant, held by the second author, in allowing him to visit Macquarie University, Sydney, thus initiating the present work.

In this paper, we show that NUCOMP does in fact yield significant improvements in speed over ordinary composition with reduction in all of the above settings for certain sizes of discriminants. We begin with a description of NUCOMP as presented in [12] which incorporates the improvements described in [3], followed by versions which are suitable for implementation in function fields over any finite field, even or odd characteristic. We then present extensive computations in imaginary quadratic number fields, imaginary quadratic function fields, real quadratic fields, and real quadratic function fields, all of which clearly demonstrate the efficiency of NUCOMP.

## 2 Description of the Algorithm

### 2.1 Number Fields

Let  $\varphi_1 = u_1X^2 + v_1XY + w_1Y^2 = (u_1, v_1, w_1)$  and  $\varphi_2 = u_2X^2 + v_2XY + w_2Y^2 = (u_2, v_2, w_2)$  be two binary quadratic forms of discriminant  $\Delta = v_1^2 - 4u_1w_1 = v_2^2 - 4u_2w_2$ . Algorithm 1 is based on Algorithm 3 from [12]. The modifications in computing the near reduced composite (Step 6 and Step 7) are from [3]. The relative generator  $\gamma$  can be used for distance computations in real quadratic fields, or not computed at all when working in imaginary quadratic fields.

**Algorithm 1 (NUCOMP).** Given two quadratic forms  $\varphi_1 = (u_1, v_1, w_1)$  and  $\varphi_2 = (u_2, v_2, w_2)$  with the same discriminant  $\Delta$ , compute  $\varphi_3 = (u_3, v_3, w_3)$  and  $\gamma$  such that  $\varphi_3 = (1/\gamma)\varphi_1\varphi_2$ . Precompute  $L = |\Delta|^{1/4}$ .

1. If  $w_1 < w_2$  swap  $\varphi_1$  and  $\varphi_2$ . Set  $s \leftarrow \frac{1}{2}(v_1 + v_2)$ ; then  $m \leftarrow v_2 - s$ .
2. Use Euclid's extended algorithm to compute  $(b, c, F)$  such that  $bu_2 + cu_1 = F = \gcd(u_1, u_2)$ . If  $F \mid s$ , set  $G \leftarrow F$ ,  $A_x \leftarrow G$ ,  $B_x \leftarrow mb$ ,  $B_y \leftarrow u_1/G$ ,  $C_y \leftarrow u_2/G$ ,  $D_y \leftarrow s/G$ , and go to Step 5.
3. If  $F \nmid s$ , use Euclid's extended algorithm again to compute  $(x, G)$  so that  $xF + ys = G = \gcd(F, s)$ , and set  $H \leftarrow F/G$ . Also set  $B_y \leftarrow u_1/G$ ,  $C_y \leftarrow u_2/G$ ,  $D_y \leftarrow s/G$ .
4. Compute  $l \leftarrow y(b(w_1 \bmod H) + c(w_2 \bmod H)) \bmod H$ ,  $B_x \leftarrow b(m/H) + l(B_y/H)$ .
5. Set  $b_x \leftarrow B_x \bmod B_y$  and  $b_y \leftarrow B_y$ . Then execute a partial Euclidean algorithm on  $b_x, b_y$ :
  - (a) Set  $x \leftarrow 1$ ,  $y \leftarrow 0$ ,  $z \leftarrow 0$ .
  - (b) If  $|b_y| > L$  and  $b_x \neq 0$  go to substep 5(c). Otherwise, if  $z$  is odd set  $b_y \leftarrow -b_y$ ,  $y \leftarrow -y$ . Then set  $a_x \leftarrow Gx$ ,  $a_y \leftarrow Gy$ . Go to Step 6.
  - (c) Let  $q \leftarrow \lfloor b_y/b_x \rfloor$  and simultaneously  $t \leftarrow b_y \bmod b_x$ . Now set  $b_y \leftarrow b_x$  and  $b_x \leftarrow t$ . Then set  $t \leftarrow y - qx$ , followed by  $y \leftarrow x$  and  $x \leftarrow t$ . Finally let  $z \leftarrow z + 1$  and go to substep 5(b).
6. [ $z = 0$ ] If  $z \neq 0$  go to Step 7. Otherwise, compute the near reduced composite  $\varphi_3 = (u_3, v_3, w_3)$  as follows:
  - (a)  $Q_1 \leftarrow c_y b_x$ ,  $c_x \leftarrow (Q_1 - m)/B_y$
  - (b)  $d_x \leftarrow (b_x D_y - w_2)/B_y$

- (c)  $u_3 \leftarrow b_y c_y$
  - (d)  $w_3 \leftarrow b_x c_x - G d_x$
  - (e)  $v_3 \leftarrow v_2 - 2Q_1$
- Go to Step 8.
7. [ $z \neq 0$ ] Compute the near reduced composite  $\varphi_3 = (u_3, v_3, w_3)$  as follows:
- (a)  $c_x \leftarrow (C_y b_x - m x) / B_y$
  - (b)  $Q_1 \leftarrow b_y c_x, Q_2 \leftarrow Q_1 + m$
  - (c)  $d_x \leftarrow (D_y b_x - w_2 x) / B_y$
  - (d)  $Q_3 \leftarrow y d_x, Q_4 \leftarrow Q_3 + D_y, d_y \leftarrow Q_4 / x$
  - (e) If  $b_x \neq 0$  set  $c_y \leftarrow Q_2 / b_x$ ; otherwise set  $c_y \leftarrow (c_x d_y - w_1) / d_x$ .
  - (f)  $u_3 \leftarrow b_y c_y - a_y d_y$
  - (g)  $w_3 \leftarrow b_x c_x - a_x d_x$
  - (h)  $v_3 \leftarrow G(Q_3 + Q_4) - Q_1 - Q_2$
8. Set  $\gamma \leftarrow x + y(v_3 - \sqrt{\Delta}) / (2G u_3)$

*Remark.* As in the regular composition algorithm, it is important to compute only the required coefficients in Euclid's extended algorithm. If only one of the two multipliers is required, some gain in speed will be obtained by not computing the second.

There are two subtle differences between our presentation of NUCOMP here as opposed to that in [12]. First, we ensure that  $w_2 < w_1$  by initially swapping  $\varphi_1$  and  $\varphi_2$  if necessary. The quantity  $w_2$  is used to compute  $\varphi_3$  in Steps 6 and 7, so this simple operation makes sure that it is the smaller of the two third coefficients. Second, we iterate the partial Euclidean algorithm until *both* values  $b_x$  and  $b_y$  are less than  $L = |\Delta|^{1/4}$ . According to computational experiments, taking these extra Euclidean steps resulted in a small improvement in the overall execution time.

Steps 6 and 7 incorporate the modifications from [3]. In the following we prove that these modifications are equivalent to the corresponding steps of Algorithm 3 of [12].

**Proposition 1.** *If  $z = 0$  after Step 5 of Algorithm 1 (NUCOMP), then Step 6 correctly computes  $\varphi_3$ .*

*Proof.* Since  $z = 0$ , we have  $c_y = C_y$  and  $d_y = D_y$ . From Algorithm 3 of [12] we get

$$\begin{aligned}
v_3 &= (a_x d_y + a_y d_x) - (b_x c_y + b_y c_x) \\
&= G d_y - b_x c_y - b_y c_x && (x = 1, y = 0) \\
&= G d_y - b_x c_y - c_y b_x + m && (c_x = (c_y b_x - m) / B_y \text{ and } b_y = B_y) \\
&= s + m - 2b_x c_y && (d_y = s / G) \\
&= v_2 - 2Q_1 && (s + m = v_2) .
\end{aligned}$$

**Proposition 2.** *If  $z \neq 0$  after Step 5 of Algorithm 1 (NUCOMP), then Step 7 correctly computes  $\varphi_3$ .*

*Proof.* Clearly  $c_x, d_x, u_3, w_3$  are as in Algorithm 3 of [12]. Now

$$c_y = Q_2/b_x = (Q_1 + m)/b_x = (b_y c_x + m)/b_x$$

$$d_y = Q_4/x = (Q_3 + D_y)/x = (y d_x + D_y)/x,$$

so  $c_y$  and  $d_y$  are also correct. From Algorithm 3 of [12] we get

$$\begin{aligned} v_3 &= (a_x d_y + a_y d_x) - (b_x c_y + b_y c_x) \\ &= G(x d_y + y d_x) - b_x c_y - b_y c_x && (a_x = Gx, a_y = Gy) \\ &= G(Q_3 + Q_4) - Q_1 - b_x c_y && (Q_4 = y d_x + D_y = x d_y) \\ &= G(Q_3 + Q_4) - Q_1 - Q_2 && (Q_2 = b_y c_x + m = b_x c_y) . \end{aligned}$$

The following algorithm, NUDUPL, corresponds to the special case of NUCOMP where  $\varphi_1 = \varphi_2$ , i.e., squaring a form. As with NUCOMP, a relative generator  $\gamma$  with respect to  $\varphi_1^2$  is also produced. Algorithm 2 is based on Algorithm 4 from [12], with a few efficiency modification added. The modifications in computing the near reduced composite (Step 6 and Step 7) are from [3].

**Algorithm 2 (NUDUPL).** Given a quadratic form  $\varphi = (u, v, w)$ , compute  $\varphi_3 = (u_3, v_3, w_3)$  and  $\gamma$  such that  $\varphi_3 = (1/\gamma)\varphi_1^2$ .

1. Use Euclid's extended algorithm to compute  $(y, G)$  such that  $xu + yv = G = \gcd(u, v)$  and set  $A_x \leftarrow G, B_y \leftarrow u/G, D_y \leftarrow v/G$ .
2. Compute  $B_x \leftarrow yw \bmod B_y$ .
3. Set  $b_x \leftarrow B_x$  and  $b_y \leftarrow B_y$ . Then execute a partial Euclidean algorithm on  $b_x, b_y$ :
  - (a) Set  $x \leftarrow 1, y \leftarrow 0, z \leftarrow 0$ .
  - (b) If  $|b_y| > L$  and  $b_x \neq 0$  go to substep 3(c). Otherwise, if  $z$  is odd set  $b_y \leftarrow -b_y, y \leftarrow -y$ . Then set  $a_x \leftarrow Gx, a_y = Gy$ . Go to Step 4.
  - (c) Let  $q \leftarrow \lfloor b_y/b_x \rfloor$  and simultaneously  $t \leftarrow b_y \bmod b_x$ . Now set  $b_y \leftarrow b_x$  and  $b_x \leftarrow t$ . Then set  $t \leftarrow y - qx$ , followed by  $y \leftarrow x$  and  $x \leftarrow t$ . Finally let  $z \leftarrow z + 1$  and go to substep 3(b).
4. [ $z = 0$ ] If  $z \neq 0$  go to Step 5. Otherwise, compute the near reduced composite  $\varphi_3 = (u_3, v_3, w_3)$  as follows:
  - (a)  $d_x \leftarrow (b_x D_y - w)/B_y$
  - (b)  $u_3 \leftarrow b_y^2, w_3 \leftarrow b_x^2$
  - (c)  $v_3 \leftarrow v - (b_x + b_y)^2 + u_3 + w_3$
  - (d)  $w_3 \leftarrow w_3 - G d_x$
Go to Step 6.
5. [ $z \neq 0$ ] Compute the near reduced composite  $\varphi_3 = (u_3, v_3, w_3)$  as follows:
  - (a)  $d_x \leftarrow (b_x D_y - wx)/B_y$
  - (b)  $Q_1 \leftarrow d_x y, d_y \leftarrow Q_1 + D_y$
  - (c)  $v_3 \leftarrow G(d_y + Q_1)$
  - (d)  $d_y \leftarrow d_y/x$
  - (e)  $u_3 \leftarrow b_y^2, w_3 \leftarrow b_x^2$

- (f)  $v_3 \leftarrow v_3 - (b_x + b_y)^2 + u_3 + w_3$   
 (g)  $u_3 \leftarrow u_3 - a_y d_y, w_3 \leftarrow w_3 - a_x d_x$   
 6. Set  $\gamma \leftarrow x + y(v_3 - \sqrt{\Delta})/(2Gu_3)$

**Proposition 3.** *If  $z = 0$  after Step 3 of Algorithm 2 (NUDUPL), then Step 4 correctly computes  $\varphi_3$ .*

*Proof.* From Algorithm 4 of [12] we have

$$\begin{aligned}
 v_3 &= (a_x d_y + a_y d_x) - 2b_x b_y \\
 &= (a_x + a_y)(d_x + d_y) - (b_x + b_y)^2 + u_3 + w_3 \quad (u_3 = b_y^2 - a_y d_y, w_3 = b_x^2 - a_x d_x) \\
 &= G(d_x + d_y) - (b_x + b_y)^2 + u_3 + w_3 \quad (x = 1, y = 0) \\
 &= Gd_x + Gd_y - (b_x + b_y)^2 + u_3 + (b_x^2 - Gd_x) \\
 &= v - (b_x + b_y)^2 + b_y^2 + b_x^2 \quad (D_y = b_1/G) .
 \end{aligned}$$

**Proposition 4.** *If  $z \neq 0$  after Step 3 of Algorithm 2 (NUDUPL), then Step 5 correctly computes  $\varphi_3$ .*

*Proof.* Clearly  $d_x, d_y, u_3, w_3$  are as in Algorithm 4 from [12]. From Algorithm 4 from [12] we have

$$\begin{aligned}
 v_3 &= (a_x d_y + a_y d_x) - 2b_x b_y \\
 &= a_x d_y + a_y d_x - (b_x + b_y)^2 + b_y^2 + b_x^2 \\
 &= Gx d_y + a_y d_x - (b_x + b_y)^2 + b_y^2 + b_x^2 \quad (a_x = Gx) \\
 &= GD_y + 2a_y d_x - (b_x + b_y)^2 + b_y^2 + b_x^2 \quad (x d_y = d_x y + D_y) \\
 &= G(D_y + 2Q_1) - (b_x + b_y)^2 + b_y^2 + b_x^2 \\
 &= G((Q_1 + D_y) + Q_1) - (b_x + b_y)^2 + b_y^2 + b_x^2 .
 \end{aligned}$$

## 2.2 Function Fields — Odd Characteristic

Let  $K = GF(q)$ ,  $q = p^n$  for some odd prime  $p$ , be a finite field of odd characteristic. Given a square-free, monic polynomial  $\Delta$  with coefficients over  $K$ , the quadratic congruence function field of discriminant  $\Delta$  is formed by adjoining  $\sqrt{\Delta}$  to the field of rational functions  $K(X)$ . The resulting field is very similar algebraically to a quadratic number field. In particular, one can study equivalence classes of ideals, infrastructure, and other properties of quadratic number fields.

Ideals in function fields are represented here almost exactly as in number fields; the two polynomials  $u(X) = u$  and  $v(X) = v$  represent the  $K[X]$ -module  $uK[x] + (v + \sqrt{\Delta})K[x]$  of norm  $u$ , where  $u \mid v^2 - \Delta$ . If we set  $w = (v^2 - \Delta)/u$ , then we have a three coefficient representation  $\varphi = (u, v, w)$  of the ideal. When viewed in this light, one realizes that the composition algorithms for binary quadratic forms, including NUCOMP and NUDUPL, generalize almost immediately to function fields. The main difference is that the formulas presented above will

compute  $\varphi_3 = (u_3, 2v_3, w_3)$  rather than  $(u_3, v_3, w_3)$ , which is easily corrected as long as the ground field has odd characteristic. The modifications to Algorithm 1 (NUCOMP) for function fields over constant fields of odd characteristic are as follows:

- Step 1.  $s \leftarrow v_1 + v_2, m \leftarrow v_2 - v_1$
- Step 6(e).  $v_3 \leftarrow v_2 - Q_1$
- Step 7(h).  $v_3 \leftarrow [G(Q_3 + Q_4) - Q_1 - Q_2]/2$
- Step 8.  $\gamma \leftarrow x + y(v_3 - \sqrt{\Delta})/(Gu_3)$

The modifications for Algorithm 2 (NUDUPL) are the following:

- Step 1.  $\dots D_y \leftarrow 2v/G$
- Step 4(c).  $v_3 \leftarrow [2v - (b_x + b_y)^2 + u_3 + w_3]/2$
- Step 5(f).  $v_3 \leftarrow [v_3 - (b_x + b_y)^2 + u_3 + w_3]/2$
- Step 6.  $\gamma \leftarrow x + y(v_3 - \sqrt{\Delta})/(Gu_3)$

In practice, the relative generator  $\gamma$  is not explicitly computed in function fields. Computing the degree of  $\gamma$  is sufficient, since it is more convenient to work with distances, i.e., the degrees of principal ideal generators and relative generators [9].

As in number fields, the main advantage of NUCOMP over composition in function fields is that the sizes of the intermediate operands remain small. In function fields, the size of the operands is measured by polynomial degree. Since reduced ideals in function fields satisfy  $\deg(u) \leq g$ , we want to use the partial Euclidean algorithm (Step 5 of NUCOMP and Step 3 of NUDUPL) to force  $\deg(b_x), \deg(b_y) < L \approx g/2$ , so that  $\deg(u_3) \approx 2\deg(b_y) \approx g$  and  $\varphi_3$  will be almost reduced. We found that taking  $L = (g + 2)/2$  for imaginary quadratic function fields ( $\deg(\Delta)$  is odd) and  $L = (g + 1)/2$  for real quadratic function fields seemed to work the best.

### 2.3 Function Fields — Even Characteristic

Let  $K = GF(q)$ ,  $q = 2^n$ , and let  $\rho$  be a root of the equation  $y^2 + h(X)y = f(X)$  defined over  $K[X]$ . Adjoining  $\rho$  to the field of rational functions yields a quadratic congruence function field. As in the odd characteristic case, we can represent ideals in the function field by triples  $(u, v, w)$  where  $w = (v^2 + h(X) + f(X))/u$ . The composition and reduction algorithms are very similar, the main difference being that the conjugate ideal of  $(u, v, w)$  is given by  $(u, v + h(X), w)$ .

The modifications to Algorithm 1 (NUCOMP) for function fields over constant fields of even characteristic follow easily from Remark 5.4 of [12], and are described below. As above,  $\rho$  is a root of  $y^2 + h(X)y = f(X)$  and we write  $h$  for  $h(X)$ .

- Step 1.  $m \leftarrow v_1 + v_2, s \leftarrow m + h$
- Step 6(e).  $v_3 \leftarrow v_2 + Q_1$
- Step 7(d).  $Q_3 \leftarrow yd_x, d_y = (Q_3 + s)/x$

- Step 7(h).  $v_3 \leftarrow Q_3 + Q_1 + v_1$
- Step 8.  $\gamma \leftarrow x + y(v_3 + h + \rho)/(Gu_3)$

The modifications for Algorithm 2 (NUDUPL) are the following:

- Step 1. Use Euclid's extended algorithm to compute  $(y, G)$  such that  $xu + yh = G = \gcd(u, h)$  and set  $A_x \leftarrow G$ ,  $B_y \leftarrow u/G$ , and  $D_y \leftarrow h/G$ .
- Step 4(c).  $v_3 \leftarrow v + b_x b_y$
- Step 5(b,c,d)  $v_3 \leftarrow d_x d_y$ ,  $d_y = (v_3 + s)/x$
- Step 5(f).  $v_3 \leftarrow v_3 + b_y b_x + v$
- Step 6.  $\gamma \leftarrow x + y(v_3 + h + \rho)/(Gu_3)$

### 3 Performance in Practice

In the following, the algorithms for composition in all cases are the optimized ideal multiplication and squaring algorithms from [5, Chapter 2]. In our experience, composition can be performed more efficiently using ideals rather than binary quadratic forms. The NUCOMP and NUDUPL algorithms are implemented as described above, but the reduction algorithm is the optimized version from [5, Chapter 2]. Thus, we are using the most efficient ideal arithmetic and reduction using standard ideal multiplication known to us, as well as the most efficient NUCOMP and reduction with forms, allowing for as unbiased a comparison as possible. All runtimes are given in CPU seconds, and the computations are performed on an 800 MHz Pentium III processor running Linux. The algorithms were implemented using the NTL computer algebra library [11] with the GNU gmp multiprecision integer package installed as the integer arithmetic kernel, and compiled with the GNU g++ compiler version 2.91.66.

#### 3.1 Imaginary Quadratic Fields

In order to compare the performance of NUCOMP and NUDUPL versus composition, we have implemented the Diffie-Hellmann key exchange protocol in the class group of an imaginary quadratic order [1]. For each discriminant size given in Table 1, we performed 5000 key exchanges with both NUCOMP and composition, using random discriminants of the given size and random exponents of the same bit-length as and bounded by  $\sqrt{|\Delta|}$ . Each communication partner performs two exponentiations per key exchange, so we expect each partner to perform about  $\log_2 |\Delta|$  NUDUPL or ideal squaring operations and half as many NUCOMP or ideal multiplication operations per key exchange. The total time for all 5000 key exchanges per communication partner and the average time for a single key exchange per partner, using composition and NUCOMP, are given in the table, as well as the ratio of the total time for all key exchanges using NUCOMP over the total time using ideal multiplication. Our computations show that NUCOMP is already more efficient for discriminants of 64 bits, and becomes even more efficient as the discriminants grow in size.

Table 1: Imaginary quadratic field key exchange comparison.

$\lceil \log_2  \Delta  \rceil$	Comp. Time		NUCOMP Time		NUCOMP/comp
	Total	Avg.	Total	Avg.	
32	7.82	0.00	8.98	0.00	1.1491
64	26.92	0.01	24.26	0.00	0.9012
128	102.95	0.02	77.83	0.02	0.7560
256	394.35	0.08	284.75	0.06	0.7221
512	1630.78	0.33	1057.69	0.21	0.6486
768	3848.80	0.77	2412.04	0.48	0.6267
1024	7291.36	1.46	4406.37	0.88	0.6043

### 3.2 Imaginary Quadratic Function Fields

We have also implemented the Diffie-Hellmann key exchange protocol in the class group of an imaginary quadratic congruence function field [7], where the ground field is any finite field of odd characteristic. The results in Table 2 were obtained using prime fields  $\mathbb{F}_p$  as ground fields, where the prime was selected to be the smallest odd prime with the given number of bits. The results in Table 3 were obtained using various extensions of  $\mathbb{F}_3$ , and those in Table 4 using extensions of  $\mathbb{F}_2$ . In each of the three tables, for each finite field and genus pair we performed a number of key exchanges using random function fields of the given genus and random exponents having the same bit-length as and bounded by  $q^g$ , where  $q$  is the cardinality of the finite field. In Table 2 and Table 4, for  $g \leq 5$  we performed 4000 key exchanges using both NUCOMP and composition, for  $5 < g \leq 10$  we performed 2000, for  $10 < g \leq 15$  we performed 1000, and for  $g > 15$  we performed 500. In Table 3, for  $g \leq 5$  we performed 200 key exchanges using both NUCOMP and composition, and for the remaining  $g$  we performed 100. Here, we expect each communication partner to perform  $2 \log_2 q^g$  NUDUPL or ideal squaring operations and half as many NUCOMP or ideal multiplication operations per key exchange. The ratio of the total time for all key exchanges using NUCOMP over the total time using ideal multiplication is given for each genus/field pair. In both tables we have not included computations for  $g = 1$  (elliptic curves), since in this case simple direct formulas exist for group arithmetic.

Table 2: Imaginary function field over  $\mathbb{F}_p$  key exchange — NUCOMP/composition.

$g$	$\lceil \log_2 p \rceil$						
	2	4	8	16	32	64	128
2	1.0778	1.2763	1.1848	1.1911	1.0979	1.0724	1.0371
3	1.2627	1.2492	1.3092	1.2922	1.1722	1.1562	1.1398
4	1.2450	1.2528	1.2698	1.2671	1.1225	1.1135	-
5	1.2365	1.1389	1.1426	1.1303	0.9997	0.9987	-
6	1.1331	1.1015	1.0792	1.0831	0.9717	0.9756	-

Table 2: (continued)

	$\lceil \log_2 p \rceil$						
$g$	2	4	8	16	32	64	128
7	1.0987	1.0272	1.0089	1.0120	0.9111	0.9179	-
8	1.0000	0.9931	0.9869	0.9950	0.8971	-	-
9	0.9903	0.9503	0.9292	0.9329	0.8411	-	-
10	0.9568	0.9218	0.9199	0.9187	0.8360	-	-
11	0.8991	0.8821	0.8732	0.8721	0.8061	-	-
12	0.8722	0.8634	0.8667	0.8641	0.8018	-	-
13	0.8552	0.8265	0.8205	0.8216	0.7681	-	-
14	0.8339	0.8206	0.8209	0.8212	0.7668	-	-
15	0.7995	0.7741	0.7751	0.7740	0.7480	-	-
20	0.7252	0.7204	0.7187	0.7217	-	-	-
25	0.6815	0.6808	0.6834	0.6848	-	-	-
30	0.6556	0.6561	0.6601	0.6631	-	-	-

Table 3: Imaginary function field over  $GF(3^n)$  key exchange — NUCOMP/composition.

	$n$				
$g$	2	3	5	10	20
2	1.0575	1.0873	1.0058	0.9835	0.9734
3	1.1604	1.1103	1.0391	1.0300	1.0037
4	1.1005	1.0461	1.0150	1.0103	1.0018
5	1.0323	0.9764	0.9722	0.9441	0.9457
6	1.0028	0.9582	0.9672	0.9646	0.9602
7	0.9455	0.9074	0.8816	0.8706	0.8713
8	0.9171	0.8933	0.8832	0.8808	0.7806
9	0.7815	0.7257	0.6723	0.6395	0.5515
10	0.7551	0.7246	0.6808	0.6652	0.5901
11	0.7244	0.7166	0.6653	0.6477	0.5902
12	0.7160	0.7122	0.6751	0.6789	0.6324
13	0.6999	0.6740	0.6738	0.6312	0.6084
14	0.6991	0.6682	0.6632	0.6221	0.5855
15	0.6835	0.6578	0.6245	0.6015	0.5369
20	0.7068	0.6826	0.6570	0.5707	-
25	0.6733	0.6622	0.6509	0.6067	-
30	0.6703	0.6541	0.6368	0.5614	-

Table 4: Imaginary function field over  $GF(2^n)$  key exchange — NUCOMP/composition.

$g$	$n$							
	1	2	4	8	16	32	64	128
2	1.7143	1.2393	1.0495	1.0237	1.0145	0.9984	0.9893	0.9629
3	1.5154	1.2348	1.1231	1.1558	1.1291	1.1222	1.1115	1.0665
4	1.2981	1.1151	1.1528	1.1425	1.1182	1.0892	1.0749	-
5	1.3746	1.1348	1.0836	1.0668	1.0507	1.0230	1.0041	-
6	1.1875	1.0657	1.0770	1.0740	1.0503	1.0098	0.9818	-
7	1.2437	1.0145	1.0052	1.0025	0.9981	0.9519	0.9483	-
8	1.0511	1.0164	0.9977	0.9978	0.9956	0.9587	-	-
9	1.0796	0.9764	0.9430	0.9415	0.9447	0.9085	-	-
10	0.9820	0.9506	0.9333	0.9293	0.9346	0.9051	-	-
11	0.9813	0.9204	0.8967	0.8942	0.9018	0.8795	-	-
12	0.9443	0.8960	0.8885	0.8937	0.9011	0.8861	-	-
13	0.9063	0.8705	0.8590	0.8589	0.8674	0.8639	-	-
14	0.9114	0.8577	0.8526	0.8589	0.8691	0.8693	-	-
15	0.8805	0.8306	0.8249	0.8254	0.8416	0.8476	-	-
20	0.8086	0.7820	0.7874	0.7918	0.8051	-	-	-
25	0.7594	0.7397	0.7442	0.7545	0.7699	-	-	-
30	0.7222	0.7176	0.7270	0.7365	0.7528	-	-	-

According to our data, NUCOMP is more efficient than composition for function fields of fairly small genus, with the trade-off point lying between genus 5 and 10, depending on the ground field. In addition, NUCOMP becomes increasingly more efficient as both the genus and the size of the ground field increase (hence the discrepancies between the trade-off points for different ground fields). Both observations are explained by the fact that NUCOMP attempts to minimize the sizes of intermediate operands. In the case of function fields, we expect the degrees of the polynomial operands to be bounded by  $O(3g/2)$  as opposed to  $O(2g)$  for composition. As the genus increases, the difference between the degrees of the operands becomes greater, and the overall speed of NUCOMP as compared to composition also increases.

The fact that NUCOMP keeps the degrees of the intermediate operands small is also significant as the size of the ground field increases. If the cost of multiplying coefficients of the polynomials is expensive, then even small reductions in the polynomial degrees become beneficial. Thus, as the ground fields become larger, the trade-off points for which NUCOMP out-performs composition occur for smaller genus. This also explains why NUCOMP seems to perform better for  $GF(3^n)$  than for  $\mathbb{F}_p$  — multiplying polynomial coefficients from  $GF(3^n)$  is more expensive than multiplying coefficients from  $\mathbb{F}_p$  even when  $3^n \approx p$ .

### 3.3 Real Quadratic Fields

In real quadratic fields, the corresponding Diffie-Hellmann key exchange protocol takes place in the principal ideal class [8, 6]. The protocol essentially consists of each partner performing two binary exponentiations of principal ideals while keeping track of the principal ideal generator or its natural logarithm (distance). In practice, maintaining these distances to sufficient accuracy is somewhat problematic. We have used the approach of  $(f, p)$ -representations from [6] to keep track of the distances, using the same precision for the distance approximations for both composition and NUCOMP. Incorporating NUCOMP into the algorithms from [6] is fairly straightforward. Our implementation using NUCOMP always produced unique key ideals for discriminants of 128 bits or more, even though the accuracy of the distance approximations is only guaranteed theoretically for regular composition [6].

For each discriminant size given in Table 5, we have performed 5000 key exchanges using random discriminants of the given size and random exponents of the same bit-length as and bounded by  $\sqrt{\Delta}$ . Each communication partner performs two exponentiations per key exchange, so we expect each partner to perform about  $\log_2 \Delta$  NUDUPL or ideal squaring operations and half as many NUCOMP or ideal multiplication operations per key exchange. The total time for all 5000 key exchanges per communication partner and the average time for a single key exchange per partner, using regular composition and NUCOMP, are given in the table, as well as the ratio of the total time for all key exchanges using NUCOMP over the total time using ideal multiplication. Our computations show that NUCOMP is more efficient for discriminants of 32 bits or more, and as in the imaginary case, it becomes even more efficient as the discriminants grow in size.

Table 5: Real quadratic field key exchange comparison.

$\lceil \log_2 \Delta \rceil$	Comp. Time		NUCOMP Time		NUCOMP/comp
	Total	Avg.	Total	Avg.	
32	23.91	0.00	21.07	0.00	0.8814
64	92.29	0.02	68.62	0.01	0.7435
128	384.75	0.08	257.31	0.05	0.6688
256	1609.77	0.32	1034.65	0.21	0.6427
512	7167.95	1.43	4253.27	0.85	0.5934
768	17133.08	3.43	10157.62	2.03	0.5929
1024	32331.92	6.47	18701.37	3.74	0.5784

Upon comparing the data for key exchange in real quadratic fields with that of imaginary quadratic fields, one finds that the benefits of using NUCOMP are somewhat more pronounced in the real case. The ideal multiplication part of the algorithms are the same in both cases, but reduction is more expensive using  $(f, p)$ -representations because fairly high precision distance approximations must be maintained. Since one benefit of NUCOMP is that a large portion of the

reduction is done beforehand, it is to be expected that NUCOMP will yield a more substantial savings in the real case, since many of the expensive reduction steps involving the distance approximations are avoided.

One area in which NUCOMP and NUDUPL are especially effective is computations where one can take advantage of the relatively small operand sizes and use single precision arithmetic rather than multiprecision. Since NUCOMP requires operands of size  $O(\Delta^{3/4})$ , one can implement NUCOMP for fields with discriminant less than  $10^{15}$  using almost exclusively single precision arithmetic (assuming 32-bit word size). For discriminants larger than  $10^{10}$ , standard ideal arithmetic requires multiprecision arithmetic since the intermediate operands can be as large as  $O(\Delta)$ .

To illustrate the effect of NUCOMP and NUDUPL in such settings, we have implemented a simple  $O(\Delta^{1/4+\epsilon})$  baby-step giant-step regulator computation routine. For each discriminant size given in Table 6, where we denote  $\log_{10} |\Delta|$  by  $\text{size}(\Delta)$ , we have computed 10000 regulators using random discriminants of the given size. The total time for all 10000 regulator computations using both regular composition and NUCOMP are given in the table, as well as the ratio of the total time using NUCOMP over the total time using ideal multiplication. As expected, the effect of NUCOMP is rather dramatic in this case, cutting the total runtime in half.

Table 6: Quadratic field regulator comparison (single precision).

size( $\Delta$ )	Regular composition	NUCOMP	NUCOMP/regular
7	144.46	79.48	0.55019
8	248.48	127.77	0.51421
9	431.27	209.19	0.48506
10	735.21	345.60	0.47007
11	1392.63	606.11	0.43523
12	2584.00	1053.50	0.40770

### 3.4 Real Quadratic Function Fields

Unlike the case of real quadratic fields, maintaining distances in real quadratic function fields is easy, since they are integers (degrees of polynomials). The corresponding key exchange protocol in the principal class [9] is very similar to that in real quadratic number fields; each communication partner has to perform two binary exponentiations of principal ideals and maintain the corresponding distances. We have also implemented this protocol, and for each finite field and genus pair in Table 7, Table 8, and Table 9, we have performed a number of key exchanges using random field discriminants of the given genus and random exponents bounded by  $q^g$ . As in the imaginary function field case, we expect each communication partner to perform  $2 \log_2 q^g$  NUDUPL or ideal squaring operations and half as many NUCOMP or ideal multiplication operations per key exchange. In Table 7 and Table 9 we performed 4000 key exchanges using both NUCOMP and composition for  $g \leq 5$ , 2000 for  $5 < g \leq 10$ , 1000 for

$10 < g \leq 15$ , and 500 for  $g > 15$ . In Table 8, function fields over  $GF(3^n)$ , we performed 200 key exchanges for  $g \leq 5$  and 100 for the remaining  $g$ . The ratio of the total time for all key exchanges using NUCOMP over the total time using ideal multiplication is given for each genus/field pair. Again, we omit the data for  $g = 1$  (elliptic curves), since the explicit formulas for the group law are more efficient than composition or NUCOMP.

Table 7: Real function field over  $\mathbb{F}_p$  key exchange — NUCOMP/composition.

	$\lceil \log_2 p \rceil$						
$g$	2	4	8	16	32	64	128
2	1.1632	1.2673	1.2823	1.2719	1.2482	1.2647	1.2886
3	1.0928	1.2228	1.2651	1.2874	1.2223	1.2296	1.2338
4	1.2165	1.1511	1.1439	1.1447	1.0531	1.0693	-
5	1.1232	1.1393	1.1344	1.1363	1.0571	1.0656	-
6	1.0704	1.0563	1.0386	1.0449	0.9595	0.9769	-
7	1.0598	1.0491	1.0486	1.0485	0.9693	0.9782	-
8	1.0506	0.9835	0.9580	0.9603	0.8898	-	-
9	1.0026	0.9810	0.9722	0.9719	0.9013	-	-
10	0.9669	0.9261	0.9171	0.9216	0.8518	-	-
11	0.9643	0.9272	0.9240	0.9257	0.8641	-	-
12	0.9089	0.8842	0.8725	0.8724	0.8175	-	-
13	0.9038	0.8809	0.8749	0.8767	0.8291	-	-
14	0.8796	0.8509	0.8423	0.8457	0.7964	-	-
15	0.8709	0.8423	0.8351	0.8386	0.8090	-	-
20	0.7804	0.7692	0.7663	0.7703	-	-	-
25	0.7485	0.7475	0.7479	0.7513	-	-	-
30	0.7185	0.7146	0.7174	0.7195	-	-	-

Table 8: Real function field over  $GF(3^n)$  key exchange — NUCOMP/composition.

	$n$				
$g$	2	3	5	10	20
2	1.1369	1.2041	1.1960	1.1403	1.1280
3	1.1388	1.1599	1.1446	1.1076	1.0972
4	1.0905	1.0526	1.0364	1.0355	1.0388
5	1.0653	1.0353	1.0307	1.0159	1.0089
6	0.9940	0.9672	0.9763	0.9833	0.9840
7	0.9873	0.9579	0.9428	0.9281	0.9291
8	0.9246	0.9049	0.9047	0.9066	0.8265
9	0.8535	0.8049	0.7464	0.7154	0.6257
10	0.8033	0.7710	0.7322	0.7164	0.6465

Table 8: (continued)

	$n$				
$g$	2	3	5	10	20
11	0.7996	0.7840	0.7383	0.7198	0.6638
12	0.7715	0.7622	0.7276	0.7214	0.6859
13	0.7742	0.7477	0.7432	0.7060	0.6803
14	0.7554	0.7230	0.7211	0.6818	0.6487
15	0.7623	0.7300	0.6931	0.6748	0.6026
20	0.7587	0.7351	0.7076	0.6306	-
25	0.7376	0.7270	0.7184	0.6703	-
30	0.7380	0.7234	0.7065	0.6388	-

Table 9: Real function field over  $GF(2^n)$  key exchange — NUCOMP/composition.

	$n$							
$g$	1	2	4	8	16	32	64	128
2	0.8066	0.9766	1.0972	1.1664	1.1583	1.1793	1.1890	1.2176
3	0.8045	1.0597	1.1841	1.1910	1.1726	1.1839	1.1696	1.1595
4	0.8501	1.0464	1.0822	1.0741	1.0657	1.0662	1.0532	-
5	0.8989	1.0925	1.1082	1.1045	1.0940	1.0740	1.0484	-
6	0.9867	1.0351	1.0219	1.0108	1.0090	0.9911	0.9867	-
7	0.9488	1.0520	1.0258	1.0292	1.0291	0.9942	0.9945	-
8	1.0292	0.9834	0.9545	0.9579	0.9662	0.9462	-	-
9	1.0031	0.9837	0.9654	0.9710	0.9793	0.9549	-	-
10	1.0222	0.9360	0.9101	0.9141	0.9283	0.9148	-	-
11	0.9866	0.9358	0.9249	0.9289	0.9390	0.9266	-	-
12	0.9771	0.8956	0.8809	0.8851	0.8991	0.8966	-	-
13	0.9427	0.9012	0.8928	0.8988	0.9093	0.9097	-	-
14	0.9492	0.8676	0.8555	0.8605	0.8775	0.8871	-	-
15	0.9329	0.8697	0.8673	0.8744	0.8897	0.8909	-	-
20	0.8596	0.8103	0.8095	0.8123	0.8240	-	-	-
25	0.8176	0.7908	0.7954	0.8060	0.8242	-	-	-
30	0.7840	0.7630	0.7681	0.7769	0.7933	-	-	-

The same observations hold here as in the imaginary function field case. The performance of NUCOMP relative to composition improves as the genus increases and as the size of the ground field increases. In addition, the relative performance of NUCOMP is better for function fields over  $GF(3^n)$  than for  $\mathbb{F}_p$ . As in the imaginary case, this is due to the fact that NUCOMP keeps the degrees of the intermediate operands small, and that the difference between operand degrees becomes more critical to the overall execution time as the ground field arithmetic becomes more expensive.

Unlike the number field case, NUCOMP does not seem to have as dramatic an effect in the real case as in the imaginary case when working in function fields. In function fields, the computational differences between the imaginary and real cases is not nearly as drastic as in number fields, since floating point approximations are not used to maintain distances. In particular, the reduction algorithms are almost identical, the only difference being that extra reduction steps are taken in the real case to ensure that the resulting composite has distance close to a given quantity. Thus, we expect that the absolute difference between the total runtimes using NUCOMP and composition to be roughly the same for the imaginary and real function field cases. This is exactly what we observed. The difference between the ratios of total NUCOMP time to total composition time between the two cases is accounted for by the fact that the amount of extra work required for the real case is the same for both NUCOMP and composition.

## References

1. J. Buchmann and H.C. Williams, *A key-exchange system based on imaginary quadratic fields*, Journal of Cryptology **1** (1988), 107–118.
2. D.G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Math. Comp. **48** (1987), no. 177, 95–101.
3. H. Cohen, *A course in computational algebraic number theory*, Springer-Verlag, Berlin, 1993.
4. S. Düllmann, *Ein Algorithmus zur Bestimmung der Klassengruppe positiv definiter binärer quadratischer Formen*, Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany, 1991.
5. M.J. Jacobson, Jr., *Subexponential class group computation in quadratic orders*, Ph.D. thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1999.
6. M.J. Jacobson, Jr., R. Scheidler, and H.C. Williams, *The efficiency and security of a real quadratic field based key exchange protocol*, Public-Key Cryptography and Computational Number Theory (Warsaw, Poland), de Gruyter, 2001.
7. N. Koblitz, *Hyperelliptic cryptosystems*, Journal of Cryptology **1** (1989), 139–150.
8. R. Scheidler, J. Buchmann, and H.C. Williams, *A key-exchange protocol using real quadratic fields*, Journal of Cryptology **7** (1994), 171–199.
9. R. Scheidler, A. Stein, and H.C. Williams, *Key-exchange in real quadratic congruence function fields*, Designs, Codes and Cryptography **7** (1996), 153–174.
10. D. Shanks, *On Gauss and composition I, II*, Proc. NATO ASI on Number Theory and Applications (R.A. Mollin, ed.), Kluwer Academic Press, 1989, pp. 163–179.
11. V. Shoup, *NTL: A library for doing number theory*, Software, 2001; see <http://www.shoup.net/ntl>.
12. A.J. van der Poorten, *A note on NUCOMP*, manuscript submitted to Math. Comp., 2001.