Improved Exponentiation and Key Agreement in the Infrastructure of a Real Quadratic Field

Vanessa Dixon*, Michael J. Jacobson Jr.**, and Renate Scheidler

Department of Computer Science, University of Calgary
2500 University Drive NW, Calgary, Alberta, Canada T2N 1N4
vanessa.dixon@gmail.com, {jacobs,rscheidl}@ucalgary.ca

Abstract. We describe improvements to the performance of a key agreement protocol based in the infrastructure of a real quadratic field through investigating fast methods for exponentiating ideals. We present adaptations of non-adjacent form and signed base-3 exponentiation and compare these to the binary method. To adapt these methods, we introduce new algorithms for squaring, cubing, and dividing w-near (f, p) representations of ideals in the infrastructure. Numerical results from an implementation of the key agreement protocol using our new algorithms and all three exponentiation methods are presented, demonstrating that non-adjacent form exponentiation improves the speed of key establishment for most of the currently recommended security levels.

Keywords: real quadratic field, infrastructure, (f, p) representation, non-adjacent form exponentiation, signed base-3 exponentiation, cryptographic key agreement.

1 Introduction

In 1988, Buchmann and Williams [3] presented a key establishment protocol analogous to that of Diffie-Hellman [6], but performed in the class group of an imaginary quadratic field. Interestingly, the security of this key agreement is related to computing the class number of the field, which is known to be at least as hard as integer factorization [15, p. 360]. The following year, Buchmann and Williams proposed a method for performing an analogous key agreement protocol in the infrastructure of the principal class of a real quadratic field [4], an abelian group-like structure that was discovered by Shanks in 1972 [18]. This contribution was noteworthy because it represented the first such protocol for which the underlying structure is not a group. Furthermore, the security of this protocol is believed to be independent of the hardness assumptions used in other public-key systems, such as the difficulty of extracting discrete logarithms on algebraic curves or in finite fields. Thus, although such systems are known to sucumb to quantum algorithms, they are nevertheless a viable alternative

* The results in this paper are from the first author's M.Sc. thesis.
** The second and third authors are supported in part by NSERC of Canada.

A. Hevia and G. Neven (Eds.): LATINCRYPT 2012, LNCS 7533, pp. 214–233, 2012.

to other widely used public-key cryptosystems that can be used in the event that efficient classical algorithms are found for discrete logarithm computation or integer factorization.

The infrastructure is comprised of the set of reduced principal ideals in the maximal order of a real quadratic field, each paired with an approximation of a floating-point *distance* that keeps track of the ideal's position in the infrastructure. The challenge is to maintain sufficient accuracy throughout the key establishment protocol to make sure that both parties obtain a shared key ideal at the end. One method for handling this problem is to use a concept called w*-near* (f, p) *representations*. These were introduced in [15], based on ideas originally presented in [13].

In this paper, we consider three methods for exponentiation of w-near (f, p) representations: binary exponentiation (BINEXP), non-adjacent form exponentiation (NAFEXP), and a signed base-3 exponentiation (SB3EXP). The first of these was previously investigated in [15], while the other two are new. The non-adjacent form of an integer is sparser than its binary form, so NAFEXP requires fewer multiplication/division steps than BINEXP on average. The base-3 representation of an integer is shorter than its binary form, so SB3EXP requires fewer cubings than BINEXP requires squarings. Hence, if cubing and division have similar time requirements to squaring and multiplying, we would expected that these algorithms improve the speed at which exponentiation can be done.

A major ingredient of all our exponentiation techniques is the efficient multiplication method for w-near (f, p) representations given as Algorithm 11.3 on pp. 275-276 of [15]. Algorithms for squaring, cubing, and division of (f, p)-representations are also required for NAFEXP and SB3EXP and are newly introduced herein. A comparative precision analysis is provided for each of these new algorithms and for key agreement using each of our three exponentiation methods. All relevant algorithms were implemented in C using the GNU Multiprecision Arithmetic Library, and test trials were run to examine their efficiency. Our timing results show that NAFEXP mostly outperforms BINEXP, whereas SB3EXP does not.

2 Infrastructure of a Real Quadratic Field

For a general introduction to real quadratic fields and their infrastructure, the reader is referred to [15]. Throughout this paper, we fix a positive square-free integer $D > 1$, and set $r = 2$ if $D \equiv 1 \pmod 4$, $r = 1$ otherwise. The field and 2-dimensional $\mathbb{Q}$-vector space $\mathbb{K} = \mathbb{Q} \oplus \mathbb{Q}\sqrt{D}$ is a *real quadratic field*. Its *discriminant* is $\Delta = 4D/r^2$, and its *maximal order* is the subring and rank 2 $\mathbb{Z}$-module $\mathcal{O} = \mathbb{Z} \oplus \mathbb{Z}\omega$, where $\omega = (r - 1 + \sqrt{D})/r$.

2.1 Ideals and Infrastructure

An $\mathcal{O}$*-ideal* (or *ideal* for short) is an additive subgroup of $\mathcal{O}$ that is closed under multiplication by elements in $\mathcal{O}$. An $\mathcal{O}$-ideal $\mathfrak{a}$ is *principal* if it consists of all the

$\mathcal{O}$-multiples of some element $\alpha \in \mathcal{O}$, called a *generator* of $\mathfrak{a}$; we write $\mathfrak{a} = (\alpha)$. It will at times be useful to consider *fractional* principal ideals $\mathfrak{a} = (\alpha)$ with $\alpha \in \mathbb{K}$, which are no longer subsets of $\mathcal{O}$, but $(d)\mathfrak{a} \subseteq \mathcal{O}$ for some $d \in \mathbb{Z}$. Two non-zero $\mathcal{O}$-ideals $\mathfrak{a}$ and $\mathfrak{b}$ are *equivalent* is there exists a non-zero $\theta \in \mathbb{K}$ with $\mathfrak{b} = (\theta)\mathfrak{a}$. Note that the non-zero principal $\mathcal{O}$-ideals are exactly the $\mathcal{O}$-ideals that are equivalent to $\mathcal{O}$.

The non-zero $\mathcal{O}$-ideals are exactly the rank 2 $\mathbb{Z}$-submodules of $\mathcal{O}$ of the form $\mathfrak{a} = \mathbb{Z}\,SQ/r \oplus \mathbb{Z}\,S(P + \sqrt{D})/r$ with $S, Q, P \in \mathbb{Z}$, $S, Q > 0$, r dividing Q and rQ dividing $D - P^2$. Here, S and Q are unique and P is unique modulo Q. The *norm* of $\mathfrak{a}$ is the positive integer $N(\mathfrak{a}) = S^2Q$. An $\mathcal{O}$-ideal $\mathfrak{a}$ is *primitive* if $S = 1$, in which case we simply write $\mathfrak{a} = (Q, P)$. The *conjugate* of a primitive ideal $\mathfrak{a} = (Q, P)$ is the primitive ideal $\overline{\mathfrak{a}} = (Q, -P)$; note that $\mathfrak{a}\overline{\mathfrak{a}} = (N(\mathfrak{a})) = (Q)$.

A primitive ideal $\mathfrak{a} = (Q, P)$ is *reduced* if P can be taken modulo Q so that $0 < \sqrt{D} - P < Q < \sqrt{D} + P$. This forces $0 < P < \sqrt{D}$ and $0 < Q < 2\sqrt{D}$, so P and Q are bounded. Consequently, the number of reduced $\mathcal{O}$-ideals is finite.

The *infrastructure* of $\mathbb{K}$ is the set $\mathcal{R}$ of all reduced principal $\mathcal{O}$-ideals. Each infrastructure ideal $\mathfrak{a}$ is associated with its unique *distance* $\delta(\mathfrak{a}) = \log \alpha \in \mathbb{R}^{\geq 0}$, where $\alpha \in \mathcal{O}$ is the smallest generator of $\mathfrak{a}$ that is at least 1. The infrastructure is thus a finite set that is ordered by distance, where the first ideal is $\mathcal{O}$ and has distance 0. *Baby steps* move cyclically through the infrastructure, obtaining from any infrastructure ideal $\mathfrak{a}$ the next infrastructure ideal $\rho(\mathfrak{a})$ in the distance ordering, along with the *relative distance* $\delta(\rho(\mathfrak{a})) - \delta(\mathfrak{a}) \in \mathbb{R}^{>0}$.

The product $\mathfrak{ab}$ of two reduced principal ideals $\mathfrak{a}, \mathfrak{b}$ is generally a non-reduced (and even non-primitive) principal $\mathcal{O}$-ideal; more exactly, $\mathfrak{ab} = (S)\mathfrak{c}$ with $S \in \mathbb{Z}^+$ and $\mathfrak{c}$ a primitive $\mathcal{O}$-ideal. *Reduction* applies the same arithmetic as the baby step operation to $\mathfrak{c}$, producing a reduced ideal equivalent to $\mathfrak{ab}$. Suppose $\mathfrak{a}$ and $\mathfrak{b}$ are infrastructure ideals, and let $\mathfrak{r} = (\theta)\mathfrak{ab} \in \mathcal{R}$, with $\theta \in \mathbb{K}$, be the first reduced ideal thus obtained. Then the operation that computes $\mathfrak{r}$ and θ from $\mathfrak{a}$ and $\mathfrak{b}$ is a *giant step*. Instead of using multiplication with subsequent reduction, giant steps can be performed more efficiently using the NUCOMP algorithm first proposed by Shanks [19]. The distance "error" $\log \theta = \delta(\mathfrak{r}) - \delta(\mathfrak{a}) - \delta(\mathfrak{b}) \in \mathbb{R}$ is generally very small compared to the distances of $\mathfrak{a}$ and $\mathfrak{b}$. It follows that under the giant step operation, the infrastructure behaves almost like a finite abelian group, where the identity is $\mathcal{O}$ and the "inverse" operation is conjugation; associativity fails, but only barely since the distance is nearly additive under giant steps.

2.2 (f, p) Representations

When performing infrastructure arithmetic, one needs to keep track of the relative distances of ideals which are real numbers. These are approximated using *w-near* (f, p) *representations* of ideals, a concept first introduced in [13] and subsequently developed in [14].

Definition 1. *[15, Definition 11.1, p. 267 and Section 11.2, p. 270] Let* $p \in \mathbb{Z}^+$, $f \in \mathbb{R}^{\geq 1}$ *and* $\mathfrak{a}$ *an* $\mathcal{O}$*-ideal. An* (f, p) representation *of* $\mathfrak{a}$ *is a triple of parameters* $(\mathfrak{b}, d, k)$ *where*

1. $\mathfrak{b}$ *is an* $\mathcal{O}$*-ideal equivalent to* $\mathfrak{a}$*,* $d \in \mathbb{Z}^+$ *with* $2^p < d \leq 2^{p+1}$*,* $k \in \mathbb{Z}$*, and*
2. *there exists* $\theta \in \mathbb{K}$ *such that* $\mathfrak{b} = (\theta)\mathfrak{a}$ *with* $|2^{p-k}\theta/d - 1| < f/2^p$*.*

The (f, p) *representation* $(\mathfrak{b}, d, k)$ *is* reduced *if* $\mathfrak{b}$ *is a reduced* $\mathcal{O}$*-ideal, and is* w-near *if in addition*

3. $k < w$ *and*
4. *if* $\rho(\mathfrak{b}) = (\phi)\mathfrak{b}$*, then there exist* $k', d' \in \mathbb{Z}$ *with* $k' \geq w$ *and* $2^p < d' \leq 2^{p+1}$ *such that* $|2^{p-k'}\theta\phi/d' - 1| < f/2^p$*.*

The intuition behind (f, p)-representations is that $d2^{k-p}$ is an approximation of the (generally unknown) relative generator θ of $\mathfrak{b}$ with respect to the (generally unknown) ideal $\mathfrak{a}$ with an accuracy of $f2^{-p}$. In this respect, p can be regarded as the *precision* of the approximation which will be fixed throughout our computations, and f as a measure of the error which will increase with each computation (error propagation). Also, since $2^p < d \leq 2^{p+1}$ and $d2^{k-p} \approx \theta$, k represents a rough approximation of the relative distance $\log \theta$ from $\mathfrak{b}$ to $\mathfrak{a}$.

Property 4 of Definition 1 implies that if $(\mathfrak{b}, d, k)$ is a w-near (f, p) representation of $\mathfrak{a}$, then $(\rho(\mathfrak{b}), d', k')$ is a reduced (but not necessarily w-near) (f, p) representation of $\mathfrak{a}$. In this case, the bounds in Lemma 11.3, p. 270, of [15] imply that qualitatively, θ and k are approximated by 2^w and w, respectively. Moreover, if $(\mathfrak{c}, g, h)$ is another w-near (f, p) representation of $\mathfrak{a}$, then by [15, Theorem 11.4, p. 271], $\mathfrak{b} \in \{\rho^{-2}(\mathfrak{c}), \rho^{-1}(\mathfrak{c}), \mathfrak{c}, \rho(\mathfrak{c}), \rho^2(\mathfrak{c})\}$. Hence, any two w-near (f, p) representations of the same ideal are within a few baby steps of one another. When exponentiation of w-near (f, p) representations is used in cryptographic key agreement, the preservation of the w-near property throughout the exponentiation algorithm will thus guarantee that the keys computed by Alice and Bob will be close to each other in the infrastructure. Experimentally it has been verified that if f is much less than 2^p, it is most often the case that $\mathfrak{b} = \mathfrak{c}$ (see also [15, Theorem 11.5, p. 272]).

Using w-near (f, p) representations also leads to computational improvements when performing infrastructure arithmetic. Each giant step $\mathfrak{r} = (\theta)\mathfrak{a}\mathfrak{b}$ performed on infrastructure ideals $\mathfrak{a}$ and $\mathfrak{b}$ produces a distance error $\log \theta$ which is generally negative. Qualitatively, this means that a small distance shortfall of $\log \theta$ is introduced in each giant step. By calculating the average distance lost due to reduction following ideal multiplication, the number of operations required to account for this lost distance can be decreased. One purpose of using w-near (f, p) representations is to counter this "head wind": the approximations of the distances of $\mathfrak{a}$ and $\mathfrak{b}$ after a giant step are adjusted (increased) so that the distance lost during the giant step is added back, thereby reducing the computational overhead required when exponentiation and key agreement are performed.

We now provide a summary of known and new results on basic arithmetic involving (f, p) representations. Our first theorem gives parameters for the product of two such representations.

Theorem 1. [15, Theorem 11.2, p. 268] *Let* $(\mathfrak{b}', d', k')$ *be an* (f', p) *representation of an* $\mathcal{O}$*-ideal* $\mathfrak{a}'$ *and* $(\mathfrak{b}'', d'', k'')$ *an* (f'', p) *representation of an* $\mathcal{O}$*-ideal* $\mathfrak{a}''$*.*

If $d'd'' \leq 2^{2p+1}$, *put* $e = \lceil d'd''/2^p \rceil$ *and* $h = k' + k''$, *else put* $e = \lceil d'd''/2^{p+1} \rceil$ *and* $h = k' + k'' + 1$. *Then* $(\mathfrak{b}'\mathfrak{b}'', e, h)$ *is an* (f, p) *representation of* $\mathfrak{a}'\mathfrak{a}''$ *where* $f = f' + f'' + 2^{-p} f' f'' + 1$.

Given an (f, p) representation of some ideal, we can compute a w-near $(f + 9/8, p)$ representation of the same ideal using an algorithm called WNEAR. The complete algorithm can be found in [15, Algorithm 11.2, pp. 454-456].

Multiplication of two w-near representations, obtaining a w-near representation of the product, is achieved using the WMULT algorithm [15, Algorithm 11.3, pp. 275-276]. WMULT first computes the ideal product followed by some reduction steps along with the necessary parameters for a reduced representation of that product. This increases the f-value of Theorem 1 by 9/8; see [15, Algorithm 11.1, p. 269]. Now the w-near property is re-established for the resulting representation of the product using WNEAR, which increases the f-value by another 9/8. Hence, given (f', p) and (f'', p) representations $(\mathfrak{b}', d', k')$ and $(\mathfrak{b}'', d'', k'')$ of $\mathfrak{a}'$ and $\mathfrak{a}''$, respectively, WMULT computes a w-near (f, p) representation of $\mathfrak{a}'\mathfrak{a}''$, where $f = f' + f'' + 2^{-p} f' f'' + 13/4$. For the complete algorithm, see [15, Algorithm 11.3, pp. 275-276].

Squaring is the special case of multiplication applied to two identical inputs and requires $\mathfrak{a}' = \mathfrak{a}''$, $\mathfrak{b}' = \mathfrak{b}''$, $d' = d''$ and $k' = k''$ in Theorem 1. The corresponding algorithm, WDUPL, is presented in the Appendix as Algorithm A.3. It contains simplifications over the general WMULT algorithm which achieve some computational improvements when performing exponentiation on (f, p) representations.

3 Cubing and Division with (f, p) Representations

An integer expressed in terms of a larger base has fewer terms compared to a smaller base, thus requiring fewer steps when used as an exponent in exponentiation. For example, it might be desirable to represent the exponent in base-3. This type of exponentiation in turn requires a cubing method for (f, p) representations and, in case signed digits are allowed, a division technique for (f, p) representations.

An algorithm for cubing ideals can be found in [11, Algorithm 4, p. 16]. This technique outputs an ideal that is at most two steps away from being reduced. The following theorem gives the parameters of an (f, p) representation $((\mathfrak{b}')^3, e, h)$ of $(\mathfrak{a}')^3$, given an (f', p) representation of $\mathfrak{a}'$. This result is attributed to A. Silvester; a proof will appear in his doctoral dissertation [20].

Theorem 2. *Let* $(\mathfrak{b}', d', k')$ *be an* (f', p) *representation of an* $\mathcal{O}$*-ideal* $\mathfrak{a}'$. *If* $(d')^3 \leq 2^{3p+1}$, *put* $e = \lceil (d')^3/2^{2p} \rceil$ *and* $h = 3k'$. *If* $2^{3p+1} < (d')^3 \leq 2^{3p+2}$, *put* $e = \lceil d'^3/2^{2p+1} \rceil$ *and* $h = 3k + 1$. *If* $(d')^3 > 2^{3p+2}$, *put* $e = \lceil (d')^3/2^{2p+2} \rceil$ *and* $h = 3k' + 2$. *Then* $((\mathfrak{b}')^3, e, h)$ *is an* (f, p) *representation of* $(\mathfrak{a}')^3$, *where* $f = 3f' + 2^{-p} 3(f')^2 + 2^{-2p}(f')^3 + 1$.

Obtaining a w-near representation of a cube proceeds analogous to multiplication or squaring: first compute the ideal cube, then apply reduction and compute the

parameters of the corresponding reduced representation, then restore the w-near property and again compute the parameters of the resulting w-near representation. The overall algorithm, called WCUBE, is presented as Algorithm A.4 in the Appendix. Note that some minor mistakes from [11, Algorithm 4, p. 16] have been corrected.[1] As before, given an (f', p) representation $(\mathfrak{b}', d', k')$ of an $\mathcal{O}$-ideal $\mathfrak{a}'$, WCUBE computes a w-near (f, p) representation of $(\mathfrak{a}')^3$, where $f = 3f' + 2^{-p}3(f')^2 + 2^{-2p}(f')^3 + 13/4$.

In order to implement exponentiation using signed digits such as NAF or signed base-3 exponentiation, we require a *division* algorithm for (f, p) representations. First, we give a brief overview of how ideal division is accomplished. Let $\mathfrak{b}' = (\theta')\mathfrak{a}'$ and $\mathfrak{b}'' = (\theta'')\mathfrak{a}''$ with $\mathcal{O}$-ideals $\mathfrak{a}', \mathfrak{a}''$, $\mathfrak{b}'$, $\mathfrak{b}''$ and relative generators $\theta', \theta'' \in \mathbb{K}$. Then $\mathfrak{b}'\overline{\mathfrak{b}''}(\theta'')\mathfrak{a}'' = \mathfrak{b}'\overline{\mathfrak{b}''}\mathfrak{b}'' = \mathfrak{b}'(N(\mathfrak{b}'')) = (\theta')\mathfrak{a}'(N(\mathfrak{b}''))$. Suppose that $\mathfrak{a}''$ divides $\mathfrak{a}'$, i.e. there exists an $\mathcal{O}$-ideal $\mathfrak{c}$ such that $\mathfrak{a}' = \mathfrak{c}\mathfrak{a}''$. For example, in our context, we will always have $\mathfrak{a}' = (\mathfrak{a}'')^n$ for some $n \in \mathbb{Z}^+$. Then $\mathfrak{b}'\overline{\mathfrak{b}''} = (\theta' N(\mathfrak{b}'')/\theta'')\mathfrak{c}$. Thus, when "dividing" an (f', p) representation $(\mathfrak{b}', d', k')$ of $\mathfrak{a}'$ by an (f'', p) representation $(\mathfrak{b}'', d'', k'')$ of $\mathfrak{a}''$, the resulting (f, p) representation should approximate the new relative generator $\theta = \theta' N(\mathfrak{b}'')/\theta''$ of $\mathfrak{b}'\overline{\mathfrak{b}''}$ with respect to $\mathfrak{c}$. The following theorem provides the exact parameters. Its proof is rather long and detailed, so for the sake of brevity, we only state the result here; the complete proof can be found in [7, Theorem 3.6.2].

Theorem 3. [7, Theorem 3.6.2, p. 51] *Let $(\mathfrak{b}', d', k')$ be an (f', p) representation of an $\mathcal{O}$-ideal $\mathfrak{a}'$ and $(\mathfrak{b}'', d'', k'')$ an (f'', p) representation of an $\mathcal{O}$-ideal $\mathfrak{a}''$ dividing $\mathfrak{a}'$, with $\mathfrak{a}' = \mathfrak{c}\mathfrak{a}''$ for some $\mathcal{O}$-ideal $\mathfrak{c}$. Define $\kappa \in \mathbb{Z}$ via $2^\kappa < N(\mathfrak{b}'') \leq 2^{\kappa+1}$ and $d^* = d'N(\mathfrak{b}'')/(2^\kappa d'')$. If $1/2 < d^* \leq 1$, put $e = \lceil 2^{p+1}d^* \rceil$ and $h = k' - k'' + \kappa - 1$. If $1 < d^* \leq 2$, put $e = \lceil 2^p d^* \rceil$ and $h = k' - k'' + \kappa$. If $2 < d^* < 4$, put $e = \lceil 2^{p-1}d^* \rceil$ and $h = k' - k'' + \kappa + 1$. Then $(\mathfrak{b}'\overline{\mathfrak{b}''}, e, h)$ is an (f, p) representation of $\mathfrak{c}$ where $f = f' + f''/(1 - f''/2^p) + f'f''/(2^p - f'') + 1$.*

WDIV performs division on w-near representations, followed by WNEAR. It takes as input a w-near (f', p) representation $(\mathfrak{b}', d', k')$ of $\mathfrak{a}'$ and a w-near (f'', p) representation $(\mathfrak{b}'', d'', k'')$ of $\mathfrak{a}''$ where $\mathfrak{a}''$ divides $\mathfrak{a}'$, and computes a w-near (f, p) representation of the quotient ideal, with $f = f' + f''/(1 - f''/2^p) + f'f''/(2^p - f'') + 13/4$. It is presented in the Appendix as Algorithm A.5.

4 Non-adjacent Form and Signed Base-3 Exponentiation

To explore how to improve the performance of key agreement in the infrastructure of a real quadratic field, we examine various exponentiation algorithms and adapt them to the setting of w-near (f, p) representations. The well known method of binary exponentiation (BINEXP) was already presented in [15]; it is

[1] The algorithm as presented in [11] computes $N = Q'/S$, $L = NQ'/r^2$, $K = R_0 v_1(2 + v_1(v_1(Q_0/r)(R_0/r)(2P_0)/r))$ (mod L) on line 3 and $M_2 = (R_i(P' + P'') + R'S)/L$ on line 19. The corrections herein are attributed to M. Jacobson, A. Silvester, and V. Dixon.

given for the sake of completeness as Algorithm A.6 in the Appendix. Here, we investigate non-adjacent form exponentiation (NAFEXP) and a signed base-3 exponentiation method (SB3EXP) for w-near (f, p) representations.

The *non-adjacent form* (NAF) of an integer is the signed base-2 representation for which no two adjacent digits are both non-zero. Given a positive integer n, the non-adjacent form of n is $n = \sum_{i=0}^{\ell_N} n_i 2^{\ell_N - i}$ where $n_0 = 1$, $n_i \in \{-1, 0, 1\}$, and $n_i n_{i-1} = 0$ for $1 \leq i \leq \ell_N$. The NAF of an integer can be computed using for example [10, Algorithm 3.30, p. 98].

The NAF has several useful properties that make it amenable for exponentiation; see [10, Theorem 3.29, p. 98] for example. The average number of required operations decreases from $(3/2)\log n$ to $(4/3)\log n$ as compared to BINEXP. However, because the binary digits are signed, both multiplication and division steps are required. If our division algorithm WDIV is approximately as fast as the multiplication algorithm WMULT, then we expect an improvement.

The algorithm NAFEXP, using non-adjacent form to exponentiate w-near (f, p) representations of $\mathcal{O}$-ideals, is given in the Appendix as Algorithm A.7. It is an adaptation of [10, Algorithm 3.31, p. 99], a binary NAF method for point multiplication on elliptic curves, adapted to work in the infrastructure of a real quadratic field. Theorem 3.29 of [10] implies that, compared to BINEXP, NAFEXP requires at most one more squaring, but reduces the average number of multiplications from $\ell_B/2$ to $\ell_N/3 \leq (\ell_B + 1)/3$ multiplications and divisions; here $\ell_B + 1$ is the binary length.

In order to use NAFEXP for key establishment, we must determine bounds on the error estimate f in the final w-near (f, p) representation of an ideal $\mathfrak{a}^n$. The proof is again long and very technical, so we only quote the final result. It establishes an upper bound a_i, given recursively in terms of a_{i-1}, on the value $f = f_i$ after the i-th step of the exponentiation. Solving the recurrence for a_i provides an upper bound a_{ℓ_N} on the final value $f = f_{\ell_N}$. The complete proof can be found in [7, Section 4.2.1].

Theorem 4. [7, Theorem 4.2.4, p. 73] *Let $p \geq 8$, $n \geq 2$, $h \geq \max\{16, \log_2 n\}$ and $f_0 < 2^{p-4}$. Put $m = 3.54 f_0 + 10.72$. If $hmn < 2^p$, then after NAFEXP has executed on input an (f_0, p) representation and an exponent n, the resulting (f, p) representation satisfies $f < mn$, and hence $f < 2^p/h \leq 2^{p-4}$.*

Both the binary form and the non-adjacent form of an integer are base-2 representations. A base-3 representation could be advantageous because it has a shorter length. However, cubing an (f, p) representation is a more costly operation than squaring. This led us to investigate if using a signed base-3 exponentiation would provide advantages over the binary method of exponentiation.

The signed base-3 representation of an integer n is $n = \sum_{i=0}^{\ell_S} n_i 3^{\ell_S - i}$, where $n_0 = 1$ and $n_i \in \{-1, 0, 1\}$ for $1 \leq i \leq \ell_S$. This representation can be computed by repeatedly dividing by 3 and choosing the remainders in $\{-1, 0, 1\}$. The premier advantage of this representation is its shorter length compared to the NAF or binary representation, namely $\ell_S \leq \log_3 n + 1 = \log_2 n / \log_2 3 + 1$. However, unlike NAF, it is not generally true that no two adjacent digits are non-zero. Since every digit in the signed base-3 representation can be either $0, 1$

or -1 with an expected uniform distribution, the average density of non-zero bits is 2/3. So the number of multiplication and division operations is on average

$$2\ell_S/3 \leq (2\log_3 n + 1)/3 = (2\log_2 n)/(3\log_2 3) + 1/3 \approx 0.42\log_2 n + 0.33.$$

The average number of operations for SB3EXP is thus less than that required for BINEXP, but greater than that required for NAFEXP.

SB3EXP employs a signed base-3 representation of the exponent in a cube and multiply/divide method for exponentiating w-near (f, p) representations, using the algorithms WCUBE, WMULT and WDIV from the previous two sections. The algorithm is given in the Appendix as Algorithm A.8. The precision analysis and proof proceed analogously to that of NAFEXP. Again, we only quote the final result; the complete proof can be found in [7, Section 4.3.1].

Theorem 5. [7, Theorem 4.3.3, p. 84] *Let $p \geq 8$, $n \geq 2$, $h \geq \max\{16, \log_2 n\}$ and $f_0 < 2^{p-4}$. Put $m = 13.7f_0 + 41.3$. If $hmn < 2^p$, then after SB3EXP has executed on input an (f_0, p) representation and an exponent n, the resulting w-near (f, p) representation satisfies $f < mn$, and hence $f < 2^p/h \leq 2^{p-4}$.*

5 Key Agreement Protocols

NAFEXP and SB3EXP can be used in a Diffie-Hellman type key agreement protocol in which two parties, Alice and Bob, establish a shared secret cryptographic key suitable for use in a block cipher such as AES. The protocol is based on the key agreement procedure presented by Buchmann and Williams [4]. Alice and Bob agree on a real quadratic field $\mathbb{K}$, an ideal $\mathfrak{g}$ in the infrastructure of $\mathbb{K}$, and an exponent bound B. Informally, the protocol proceeds as follows.

Protocol 6. (Infrastructure cryptographic key agreement [15, p. 365]**)**

1. Alice secretly generates a random integer a with $0 < a < B$. She computes an infrastructure ideal $\mathfrak{a} = (\theta_a)\mathfrak{g}^a$ with $\theta_a \approx 1$ and sends $\mathfrak{a}$ to Bob.
   Bob secretly generates a random integer b with $0 < b < B$. He computes an infrastructure ideal $\mathfrak{b} = (\theta_b)\mathfrak{g}^b$ with $\theta_b \approx 1$ and sends $\mathfrak{b}$ to Alice.
2. Alice computes $\mathfrak{k}_a = (\theta_\alpha)\mathfrak{b}^a$, where $\theta_\alpha \approx 1$.
   Bob computes $\mathfrak{k}_b = (\theta_\beta)\mathfrak{a}^b$, where $\theta_\beta \approx 1$.

Clearly, $\mathfrak{k}_a$ and $\mathfrak{k}_b$ are both equivalent to $\mathfrak{g}^{ab}$. Ensuring that the relative generators in each step are close to 1 guarantees that $\mathfrak{k}_a = (\alpha)\mathfrak{g}^{ab}$ and $\mathfrak{k}_b = (\beta)\mathfrak{g}^{ab}$ where $\alpha, \beta \approx 1$. By tracking the relative generators with enough precision, it is possible to ensure that $\mathfrak{k}_a = \mathfrak{k}_b$. However, if the precision requirements are relaxed, then the computation speed and memory requirements of the protocol may be improved. In this case, the key ideal $\mathfrak{k}_a$ computed by Alice may not be the same as Bob's key ideal $\mathfrak{k}_b$, but will instead be within a few baby steps of $\mathfrak{k}_b$, resulting in a small set of possible key ideals. The ambiguity arising from this can be resolved by encrypting and decrypting a message [15, p. 369].

Approximating the relative generators can be accomplished using w-near (f,p) representations. We adapt the key agreement protocol from [15, Protocol 14.1, pp. 368-369] to use the exponentiation algorithms NAFEXP and SB3EXP from the previous section. In this protocol, Alice and Bob agree on a discriminant Δ, a w-near (f,p) representation $(\mathfrak{g}_0, d_0, k_0)$ of some reduced principal ideal in the infrastructure of $\mathbb{Q}(\sqrt{\Delta})$, an exponent bound B and a precision value p which depends on the exponentiation algorithm used and must be chosen large enough so that Alice's and Bob's keys are close to each other in the infrastructure (see Theorem 8 below). As described in [7, p. 88], the value $w = \lceil (\log \Delta)/4 \rceil$ is selected in order to minimize the number of required adjustment steps in the infrastructure. We also use the fact that by [15, Lemma 14.2, p. 367], if $(\mathfrak{b}, d, k)$ is a w-near (f,p) representation of some $\mathcal{O}$-ideal and $r < p$, then $(\mathfrak{b}, d', k)$ is a w-near $(f + 2^r, p)$ representation of the same $\mathcal{O}$-ideal with $d' = 2^r \lceil 2^{-r} d \rceil$. This means that d-values can be truncated by r bits at the expense of an error increase of 2^r. In the protocol, EXP is one of BINEXP, NAFEXP or SB3EXP, and $r = \lfloor \log_2 B \rfloor$.

Protocol 7. (Key agreement using (f,p) representations)

1. Alice secretly generates a random integer a with $0 < a < B$, computes $(\mathfrak{a}, d_a, k_a) = \mathrm{EXP}((\mathfrak{g}_0, d_0, k_0), a, w, p)$, and sends $(\mathfrak{a}, \lceil 2^{-r} d_a \rceil, k_a)$ to Bob.
   Bob secretly generates a random integer b with $0 < b < B$, computes $(\mathfrak{b}, d_b, k_b) = \mathrm{EXP}((\mathfrak{g}_0, d_0, k_0), b, w, p)$, and sends $(\mathfrak{b}, \lceil 2^{-r} d_b \rceil, k_b)$ to Alice.
2. Alice computes $(\mathfrak{k}, d, k) = \mathrm{EXP}((\mathfrak{a}, 2^r \lceil 2^{-r} d_a \rceil, k_a), b, w, p)$.
   Bob computes $(\mathfrak{m}, e, h) = \mathrm{EXP}((\mathfrak{b}, 2^r \lceil 2^{-r} d_b \rceil, k_b), a, w, p)$.

If all parameters are chosen appropriately, then we generally expect that $\mathfrak{k} = \mathfrak{m}$, which is the shared key ideal. In general, by Theorem 8 below, $\mathfrak{k}$ is within two baby steps of $\mathfrak{m}$ in either direction, and a common key ideal can be established through a test encryption/decryption as mentioned above.

The security of Protocol 7 rests on the assumption that the principal ideal problem is hard. Given an $\mathcal{O}$-ideal $\mathfrak{a}$, the *principal ideal problem* is to determine whether $\mathfrak{a}$ is principal and, if so, compute an approximation of $\log \alpha$ where $\mathfrak{a} = (\alpha)$; see [15, Definition 13.21, p. 331]. Using an algorithm described in [15, Section 13.5, pp. 331-333], it has been established [15, Theorem 13.24, p. 332] that assuming the Generalized Riemann Hypothesis and the Extended Riemann Hypothesis, the principal ideal problem in a real quadratic field of discriminant $\Delta \geq 42$ can be solved in expected time

$$L_\Delta[1/2, \sqrt{2} + o(1)] = \exp\left((\sqrt{2} + o(1))(\log|\Delta|)^{1/2}(\log\log|\Delta|)^{1/2}\right).$$

6 Error Analysis for Key Agreement

In order for Protocol 7 to be successful, the precision p must be sufficiently high to ensure that Alice and Bob's keys are within two baby steps (backwards or forwards) of each other. For BINEXP, this was analyzed in [15], while the corresponding results for NAFEXP and SB3EXP are new. The proofs are very technical and are thus omitted herein.

Theorem 8. *Let $p, B \in \mathbb{Z}^+$ with $B \geq 14$ and set $r = \lfloor \log_2 B \rfloor$. Let $a, b, \in \mathbb{Z}$ with $0 < a, b < B$, and let $(\mathfrak{g}_0, d_0, k_0)$ be a $\lceil (\log \Delta)/4 \rceil$-near $(17/8, p)$ representation of a reduced principal ideal $\mathfrak{g}$. Set*

$$C = \begin{cases} 66 & \text{if EXP} = \text{BINEXP}, \\ 68 & \text{if EXP} = \text{NAFEXP}, \\ 982 & \text{if EXP} = \text{SB3EXP}. \end{cases}$$

If $2^p \geq CB^2 \max\{16, \log_2 B\}$, then $(\mathfrak{k}, d, k)$ and $(\mathfrak{m}, e, h)$ as given in round 2 of Protocol 7 are w-near (f, p) representations of $\mathfrak{g}^{ab}$ with $f < 2^{p-4}$. Hence $\mathfrak{k} \in \{\rho^{-2}(\mathfrak{m}), \rho^{-1}(\mathfrak{m}), \mathfrak{m}, \rho(\mathfrak{m}), \rho^2(\mathfrak{m})\}$.

Proof. For BINEXP, see [15, Theorem 14.3, p. 367, and Theorem 14.4, p. 368]. For NAFEXP, see [7, Theorems 5.3.1, p. 93, and Theorem 5.3.2, p. 94]. Finally, for SB3EXP, see [7, Theorem 5.4.1, p. 96, and Theorem 5.4.2, p. 97].

Theorem 8 can be used directly for key agreement: Alice and Bob simply use a precision value p such that $p \geq \log_2(CB^2 \max\{16, \log_2 B\})$ where C is the appropriate value as specified in the theorem. Note that the above bounds on p imply that NAFEXP requires at most one more bit of precision than BINEXP, and SB3EXP at most 15 bits more, to guarantee the result of Theorem 8.

7 Numerical Results

The algorithms BINEXP, NAFEXP and SB3EXP were implemented in C on a Dell Power Edge R910 server provided by the Department of Mathematics and Statistics at the University of Calgary. This server has 64 logical CPUs Intel(R) Xeon(R) CPU X7550 @ 2.00GHz with 128G RAM. The operating system is Red Hat Enterprise Linux Server 5.6. The GNU Multiple Precision Arithmetic Library (GMP) [8] was used for integer arithmetic.

We used discriminants with the bit lengths recommended in [2], which provide the same level of security as block ciphers with 112, 128, 192, and 256 bit keys as recommended by NIST [1]. We also used the exponent bound $B = 2^{2k}$ where k is the number of bits in the corresponding block cipher (112, 128, 192, 256) [15, pp. 372-373]. This bound ensures that an attack on the protocol using a baby-step giant-step method would take time approximately 2^k, which is roughly the same time required to solve the principal ideal problem (which is believed to be hard) using index calculus [15, p. 372]. The values of p and w were computed as described in the previous section. The parameters used in Protocol 7 that were not trial dependent are listed in Table 1.

The choice of the discriminant Δ of $\mathbb{K}$ is important for cryptographic security. Heuristically, to ensure that the infrastructure has a cardinality of order $\sqrt{\Delta}$, the best choices are prime discriminants $\Delta \equiv 1 \pmod 4$; see the discussion on p. 371 of [15]. Hence, for each trial, we generated a random probable prime value $\Delta = D \equiv 1 \pmod 4$ using random number generating and probable prime finding algorithms in GMP.

Table 1. Parameters for the trials. The values of p_B, p_N, p_S correspond to the precision values of BINEXP, NAFEXP, and SB3EXP, respectively.

$\log_2 B$	$\log_2 \Delta$	w	p_B	p_N	p_S
224	1341	335	462	462	446
256	1818	454	527	527	530
384	3586	896	783	783	787
512	5957	1489	1040	1040	1043

For better performance, the partial GCD steps (lines 7-12 of WDUPL and lines 11-16 of WCUBE) were implemented using Lehmer's algorithm [16,17]; this is not reflected in the pseudocode as it appears in the Appendix .

The entries in Table 2 are the result of 10 000 trials for each discriminant size. We used a w-near $(17/8, p)$ representation of $\rho^5(\mathcal{O})$, obtained by running WNEAR on input the reduced $(1, p)$ representation $(\rho^5(\mathcal{O}), 2^p + 1, 0)$ of $\rho^5(\mathcal{O})$, as the input ideal in each trial. Two exponents $a, b < B$ were also generated randomly for each trial. The time required to perform Protocol 7 (double exponentiation) was recorded, and we took the average over all trials. Columns 3, 4 and 6 record these times using BINEXP, NAFEXP and SB3EXP, respectively. Columns 5 and 7 show the percentage difference between the new methods (NAFEXP and SB3EXP) and BINEXP, computed using the formula $100\% \cdot (t_B - t)/t_B$ where t_B is the average CPU time from column 3 of Table 2 and t is the average CPU time from column 4 for NAFEXP or column 6 for SB3EXP from the same table. We see that NAFEXP is initially faster than BINEXP on average, as is SB3EXP for the smallest discriminant bit length, but as the size of the discriminant grows, the computational advantage diminishes. We conjecture that, especially in the SB3EXP case, this may be due to the relative costs of WDIV and WCUBE with respect to WMULT and WDUPL; further investigation is required to determine this and reduce any observed discrepancies. Another factor may be that the binary expansion does not require pre-computation or storage, whereas the NAF and SB3EXP must be pre-computed. Using a "right-to-left" variant of NAF may alleviate this problem.

Table 2. Average CPU times (in seconds) per key agreement per partner with Lehmer's partial GCD algorithm for 1000 trials

$\log_2 B$	$\log_2 \Delta$	BINEXP	NAFEXP	% diff	SB3EXP	% diff
224	1341	0.122770	0.109070	11.2	0.119250	2.9
256	1818	0.193320	0.169720	7.8	0.197480	-2.2
384	3586	0.712660	0.674110	5.4	0.823520	-15.6
512	5957	1.932860	2.029010	-5.0	2.593800	-32.4

It is of interest to find out how often Alice and Bob might not compute the same key ideal using the methods described above. For 10 000 trials using a 1341-bit discriminant, no mismatches were found for the ideals computed with any of the three exponentiation methods. The number of mismatches for BINEXP was found to decrease with discriminant size in previous work [15, Table 14.1, p. 369].

We also wished to determine whether the precision bounds established in our analysis herein were tight. Using 200 trials, we determined for each exponentiation method the minimum precision value for which all of the 200 trial key agreements were successful (i.e. established ideals that were within ± 2 baby steps of each other). We found that BINEXP required 15-16 fewer bits of precision than the theoretical lower bound. For NAFEXP we could use 14-17 fewer bits of precision, and for SB3EXP, 17-19 fewer bits of precision were sufficient. The precision results of these trials are listed in [7, Table 5.9]. The 200 precision tests were timed in order to determine if an improved precision analysis could change our efficiency results. We found that the timings with the reduced precision followed the same trends as those of Table 2, and that the performance improvement was negligible. These results are listed in [7, Table 5.10].

8 Conclusions

We investigated methods for exponentiation in the infrastructure of a real quadratic field that were used in a key agreement protocol. We found that using the non-adjacent form exponentiation method for w-near (f, p) representations improved the computing time for most discriminant sizes tested and, according to our analysis, typically did not increase the precision requirements compared to the corresponding binary exponentiation method. However, it remains an open problem to determine why NAFEXP slows in comparison to BINEXP as the discriminant size increases. Our preliminary investigation of this behavior showed that the time devoted to computing the non-adjacent form of the exponents did not fully account for why NAFEXP slowed down in comparison to BINEXP. Similar further work is required for SB3EXP. It remains to be determined whether this observed slowing is a result of the implementation of the methods or if it is a property of the methods themselves.

There are exponentiation methods that improve on NAFEXP. One could employ a sliding window NAF method [10, Algorithm 3.38, p. 101] which uses pre-computations to decrease the number of operations in the actual algorithm. An adaptation of a double-base method such as the ternary/binary method described in [5] would also be of interest as this could limit the use of the cubing steps compared to SB3EXP while still reducing the overall number of operations compared to BINEXP. To use these methods, an investigation of the precision requirements would be needed.

Further optimization of our algorithms and their implementations is possible. For example, the NUCOMP and WNEAR methods could be combined in such a way so that NUCOMP passes certain values to WNEAR, thereby reducing computation effort. Our precision analysis assumed a worst case scenario, so the

required precision is not tight in practice. It is also worthwhile to investigate whether $w = (\log \Delta)/4$ is the best choice of w for SB3EXP; if this choice of w is not optimal, then the performance of our ternary algorithm could be improved.

Recent results from hyperelliptic curve cryptography, particularly those from [12], employ scalar multiplication methods in which a large number of giant steps is replaced by a series of (much faster) baby steps. These could potentially be adapted to the real quadratic field infrastructure setting.

Section 7 established that using NAFEXP improved the efficiency of infrastructure based key agreement compared to using BINEXP, which is not the case, for example, in the original Diffie-Hellman protocol [6]. Improvements of this kind are important for practical use. It is essential to have a varied set of cryptographic protocols for key agreement due to the constant pressure of technology and advancement of cryptographic attacks. The methods developed herein could also be used to develop a signature scheme similar to that of Guillou and Quisquater [9], which would expand the breadth of cryptographic protocols available for use.

References

1. Barker, E., Barker, W., Polk, W., Smid, M.: Recommendation for key management - part 1: General (revised). NIST Special Publication 800-57, National Institute of Standards and Technology (NIST) (March 2007), http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1_3-8-07.pdf
2. Biasse, J.-F., Jacobson Jr., M.J., Silvester, A.K.: Security Estimates for Quadratic Field Based Cryptosystems. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 233–247. Springer, Heidelberg (2010), http://dl.acm.org/citation.cfm?id=1926211.1926229
3. Buchmann, J., Williams, H.C.: A key-exchange system based on imaginary quadratic fields. Journal of Cryptology 1, 107–118 (1988)
4. Buchmann, J., Williams, H.C.: A Key Exchange System Based on Real Quadratic Fields. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 335–343. Springer, Heidelberg (1990), http://dl.acm.org/citation.cfm?id=646754.705067
5. Ciet, M., Joye, M., Lauter, K., Montgomery, P.: Trading inversions for multiplications in elliptic curve cryptography. Designs, Codes and Cryptography 39, 189–206 (2006)
6. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
7. Dixon, V.: Fast Exponentiation in the Infrastructure of a Real Quadratic Field. Master's thesis, University of Calgary, Calgary, Alberta (2011)
8. Free Software Foundation: The GNU Multiple Precision Arithmetic Library (2011), http://gmplib.org
9. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)

10. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography, pp. 98–99. Springer Science and Buisness Media, LLC (2004)
11. Imbert, L., Jacobson Jr., M.J., Schmidt, A.: Fast ideal cubing in imaginary quadratic number and function fields. Advances in Mathematics of Communications 4(2), 237–260 (2010)
12. Jacobson Jr., M.J., Scheidler, R., Stein, A.: Cryptographic aspects of real hyperelliptic curves. Tatra Mountains Mathematical Publications 45, 1–35 (2010)
13. Jacobson Jr., M.J., Scheidler, R., Williams, H.C.: The efficiency and security of a real quadratic field based key exchange protocol. In: Public Key Cryptography and Computational Number Theory (Warsaw 2000), pp. 89–112. Walter de Gruyter, Berlin (2001)
14. Jacobson Jr., M.J., Scheidler, R., Williams, H.C.: An improved real quadratic field based key exchange procedure. Journal of Cryptology 19, 211–239 (2006)
15. Jacobson Jr., M.J., Williams, H.C.: Solving the Pell Equation. CMS Books in Mathematics. Springer (2009) iSBN 978-0-387-84922-5
16. Jebelean, T.: A double-digit Lehmer-Euclid algorithm for finding the GCD of long integers. Journal of Symbolic Computation 19, 145–157 (1995)
17. Lehmer, D.H.: Euclid's algorithm for large numbers. The American Mathematical Monthly 45(4), 227–233 (1938)
18. Shanks, D.: The infrastructure of real quadratic fields and its applications. In: Proc. 1972 Number Theory Conf., Boulder, Colorado, pp. 217–224 (1972)
19. Shanks, D.: On Gauss and composition I, II. In: Proceedings NATO ASI on Number Theory and Applications, pp. 163–204. Kluwer, Dordrecht (1989)
20. Silvester, A.: Doctoral Dissertation, University of Calgary (in progress, 2012)

A Appendix

For reference, pseudocode for our new algorithms is listed below. WDUPL, WCUBE, and WDIV all require the sub-algorithms REMOVE (for adjusting the relative generator approximation obtained after the operation) and WNEAR (for adjusting the output to satisfy the definition of a w-near (f,p) representation). The specifications of these algorithms are listed for convenience, together with references to their descriptions. Complete descriptions of the remaining algorithms can be found in [7].

Algorithm A.1. REMOVE [15, Algorithm A.1, p. 448]

Input: $(\mathfrak{b},e,h),T,C,s,p$, where $((\mu)\mathfrak{b},e,h)$ is an (f,p) representation of some ideal $\mathfrak{a}$ with $\mu = |(A+B\sqrt{D})/C| \geq 1$ ($A,B,C \in \mathbb{Z}$ and $C \neq 0$), $T = 2^sA + B\lfloor 2^s\sqrt{D}\rfloor$, and $s \in \mathbb{Z}^{\geq 0}$ with $2^s|C| > 2^{p+4}|B|$.

Output: An $(f+9/8,p)$ representation $(\mathfrak{b},e',h')$ of $\mathfrak{a}$.

Algorithm A.2. WNEAR [15, Algorithm 11.2, pp. 454-456]

Input: $(\mathfrak{b},d,k),w,p$, where $(\mathfrak{b},d,k)$ is an (f,p) representation of some $\mathcal{O}$-ideal $\mathfrak{a}$. Here, $\mathfrak{b} = \left[Q/r,(P+\sqrt{D})/r\right]$, where $P+\sqrt{D} \geq Q$, $0 \leq \lfloor\sqrt{D}\rfloor - P \leq Q$.

Output: A w-near $(f+9/8,p)$ representation $(\mathfrak{c},g,h)$ of $\mathfrak{a}$.

Algorithm A.3. WDUPL

Input: $(\mathfrak{b}', d', k'), w, p$, where $(\mathfrak{b}', d, k)$ is a w-near (f', p) representation of some ideal $\mathfrak{a}$. Here $\mathfrak{b}' = [Q'/r, (P' + \sqrt{D})/r]$.

Output: A reduced w-near (f, p) representation $(\mathfrak{c}, d, k)$ of $\mathfrak{a}^2$ where $f = 2f' + 2^{-p}(f')^2 + 13/4$.

/* *Finding a reduced* $\mathfrak{b}$ *and* $\mu = |(A + B\sqrt{D})/C| \in \mathbb{K}$ *satisfying* $(\mu)\mathfrak{b} = (\mathfrak{b}')^2$. */

Compute $S = Z(Q'/r) + Y(2P'/r)$ where $S = \gcd(Q'/r, 2P'/r)$ for $S, Y \in \mathbb{Z}$.
Set $N = Q'/(Sr)$, $L = Q'/S$, $K = YR'$ (mod L).
Set $R_{-1} = L$, $R_0 = K$, $C_{-1} = 0$, $C_0 = -1$ and $i = 0$.
if $R_{-2} < \lfloor \sqrt{2rD^{1/2}} \rfloor$ **then**
  Put $Q_{i+1} = (Q')^2/(rS^2)$ and $P_{i+1} \equiv P' + YR'Q'/(rS)$ (mod Q_{i+1}).
else
  while $R_i > \sqrt{2r}|D|^{1/4}$ **do**
    $i \leftarrow i + 1$
    $q = \lfloor R_{i-2}/R_{i-1} \rfloor$
    $R_i = R_{i-2} - q_i R_{i-1}$
    $C_i = C_{i-2} - q_i C_{i-1}$
  end while
  $M_2 = (R_i 2P' + rSR'C_i)/L$
  $Q_{i+1} = (-1)^{i-1}(R_i^2/r - C_i M_2)$
  $P_{i+1} = (NR_i + Q_{i+1}C_{i-1})/C_i - P'$
end if
/* *Final reduction steps* */
$j = 1$
$Q'_{i+1} = |Q_{i+1}|$
$k_{i+1} = \lfloor (\sqrt{D} - P_{i+1})/Q'_{i+1} \rfloor$
$P'_{i+1} = k_{i+1}Q'_{i+1} + P_{i+1}$
$\sigma = \mathrm{sign}(Q_{i+1})$
$B_{i-1} = \sigma|C_{i-1}|$
$B_{i-2} = |C_{i-2}|$.
if $P'_{i+1} + \lfloor \sqrt{D} \rfloor < Q'_{i+1}$ **then**
  $j = 2$
  $q_{i+1} = \lfloor (P_{i+1} + \lfloor \sqrt{D} \rfloor)/Q'_{i+1} \rfloor$
  $P_{i+2} = q_{i+1}Q'_{i+1} - p_{i+1}$, $Q_{i+2} = (D - P_{i+2}^2)/Q'_{i+1}$
  $Q'_{i+2} = |Q_{i+2}|$
  $k_{i+2} = \lfloor (\sqrt{D} - P_{i+2})/Q'_{i+2} \rfloor$
  $P'_{i+2} = k_{i+2}Q'_{i+2} + P_{i+2}$
  $B_{i+1} = q_{i+1}B_i + B_{i+1}$.
  if $P'_{i+2} + \lfloor \sqrt{D} \rfloor < Q'_{i+2}$ **then**
    $j = 3$
    $q_{i+2} = \lfloor (P_{i+2} + \lfloor \sqrt{D} \rfloor)/Q_{i+2} \rfloor$
    $P_{i+3} = q_{i+2}Q_{i+2} - p_{i+2}$

36: $Q_{i+3} = (D - P_{i+3}^2)/Q_{i+2}$
37: $Q'_{i+3} = |Q_{i+3}|$
38: $k_{i+3} = \lfloor(\sqrt{D} - P_{i+3})/Q'_{i+3}\rfloor$
39: $P'_{i+3} = k_{i+3}Q'_{i+3} + P_{i+3}$
40: $B_{i+2} = q_{i+2}B_{i+1} + B_{i+2}$.
41: **end if**
42: **end if**
43: Put $\mathfrak{b} = [Q'_{i+j}/r, (P'_{i+1} + \sqrt{D})/r]$, $A = S(Q_{i+j}B_{i+j-2} + P_{i+j}B_{i+j-1})$, $B = -SB_{i+j-1}$, and $C = Q_{i+j}$.

/ Using Theorem 1 with equal inputs */*
44: **if** $(d')^2 \leq 2^{2p+1}$ **then**
45: Put $e = \lfloor(d')^2/2^p\rfloor$, $h = 2k'$.
46: **else**
47: Put $e = \lfloor(d')^2/2^{p+1}\rfloor$, $h = 2k' + 1$.
48: **end if**

/ Bounding μ and calling REMOVE. */*
49: Find $s \geq 0$ such that $2^sQ > 2^{p+4}B$ and put $T = 2^sA + B\lfloor 2^s\sqrt{D}\rfloor$.
50: $(\mathfrak{b}, e', h') = \text{REMOVE}((\mathfrak{b}, e, d), T, C, s, p)$.

/ Calling WNEAR */*
51: $(\mathfrak{c}, d, k) = \text{WNEAR}((\mathfrak{b}, e', h'), w, p)$

Algorithm A.4. WCUBE

Input: $(\mathfrak{b}', d', k')$, w, p where $(\mathfrak{b}', d', k')$ is a reduced w-near (f', p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}'$. Here $\mathfrak{b}'' = \left[Q'/r, (P' + \sqrt{D})/r\right]$.

Output: A w-near $(3f' + 3(f')^2 2^{-p} + (f')^3 2^{-2p} + 13/4, p)$ representation $(\mathfrak{c}, d, k)$ of $(\mathfrak{a}')^3$.

/ Finding a reduced* $\mathfrak{b}$ *and* $\mu = (A + B\sqrt{D})/C \in \mathbb{K}$ *for which* $(\mu)\mathfrak{b} = (\mathfrak{b}')^2$. *Lines 1-22 are [11, Algorithm 4, p. 16]. */*

1: Find $S', v_1 \in \mathbb{Z}$ such that $S' = \gcd(Q'/r, 2P'/r)$ and $S' = u_1(Q'/r) + v_1(2P'/r)$.
2: **if** $S' = 1$ **then**
3: Set $S = 1$, $N = Q'/r$, $L = NQ'$, and $K = R'v_1(2 - v_1(v_1(Q'/r)(R'/r) + (2P')/r)) \pmod{L}$.
4: **else**
5: Compute $S = u_2(S'Q'/r) + v_2((3(P')^2 + D)/r^2)$, $N = Q'/(rS)$, $L = NQ'$, and $K = R'(u_2v_1(Q'/r) + v_2(2P'/r)) \pmod{L}$.
6: **end if**
7: **if** $L < \sqrt{Q'/r^2}|D|^{1/4}$ **then**
8: Set $Q = NL$ and $P = P' + NK$.
9: **else**
10: Set $R_{-1} = L, R_0 = K, C_{-1} = 0, C_0 = -1, i = 0$.
11: **while** $R_i > \sqrt{Q'/r^2}|D|^{1/4}$ **do**
12: $i \leftarrow i + 1$
13: $q_i = \lfloor R_{i-2}/R_{i-1} \rfloor$
14: $R_i = R_{i-2} - q_i R_{i-1}$
15: $C_i = C_{i-2} - q_i C_{i-1}$
16: **end while**
17: $P'' = P' + NK \pmod{L}$
18: $M_1 = (NR_i + (P'' - P')C_i)/L$
19: $M_2 = (R_i(P' + P'') + rR'SC_i)/L$
20: $Q = (-1)^{i-1}(R_iM_1 - C_iM_2)$
21: $P = (NR_i + QC_{i-1})/C_i - P'$
22: **end if**

/ Final reduction steps (using the same method as NUCOMP). */*

23: $j = 1$
24: $Q'_{i+1} = Q_i = |Q|$
25: $k = \lfloor (\sqrt{D} - P_{i+1})/Q'_{i+1} \rfloor$
26: $P'_{i+1} = k_{i+1}Q'_{i+1} + P_{i+1}$
27: $\sigma = \text{sign}(Q)$
28: $B_{i-1} = \sigma|C_{i-1}|$
29: $B_i = |C_i|$
30: **if** $P'_{i+1} + \lfloor \sqrt{D} \rfloor < Q'_{i+1}$ **then**
31: $j = 2$

32: $q_{i+1} = \lfloor (P_{i+1} + \lfloor \sqrt{D} \rfloor)/Q'_{i+1} \rfloor$
33: $P_{i+2} = q_{i+1}Q'_{i+1} - p_{i+1}$, $Q_{i+2} = (D - P^2_{i+2})/Q'_{i+1}$
34: $Q'_{i+2} = |Q_{i+2}|$
35: $k_{i+2} = \lfloor (\sqrt{D} - P_{i+2})/Q'_{i+2} \rfloor$
36: $P'_{i+2} = k_{i+2}Q'_{i+2} + P_{i+2}$
37: $B_{i+1} = q_{i+1}B_i + B_{i+1}$.
38: **if** $P'_{i+2} + \lfloor \sqrt{D} \rfloor < Q'_{i+2}$ **then**
39: $j = 3$
40: $q_{i+2} = \lfloor (P_{i+2} + \lfloor \sqrt{D} \rfloor)/Q_{i+2} \rfloor$
41: $P_{i+3} = q_{i+2}Q_{i+2} - p_{i+2}$
42: $Q_{i+3} = (D - P^2_{i+3})/Q_{i+2}$
43: $Q'_{i+3} = |Q_{i+3}|$
44: $k_{i+3} = \lfloor (\sqrt{D} - P_{i+3})/Q'_{i+3} \rfloor$
45: $P'_{i+3} = k_{i+3}Q'_{i+3} + P_{i+3}$
46: $B_{i+2} = q_{i+2}B_{i+1} + B_{i+2}$.
47: **end if**
48: **end if**
49: Put $\mathfrak{b} = [Q'_{i+j}/r, (P'_{i+1} + \sqrt{D})/r]$, $A = S(Q_{i+j}B_{i+j-2} + P_{i+j}B_{i+j-1})$, $B = -SB_{i+j-1}$, and $C = Q_{i+j}$.

/ Using Theorem 2 */*
50: **if** $(d')^3 \leq 2^{3p+1}$ **then**
51: Put $e = \lfloor (d')^3/2^{2p} \rfloor$ and $h = 3k'$.
52: **else if** $2^{3p+1} < (d')^3 \leq 2^{3p+2}$ **then**
53: Put $e = \lfloor (d')^3/2^{2p+1} \rfloor$ and $h = 3k' + 1$.
54: **else**
55: Put $e = \lfloor (d')^3/2^{2p+2} \rfloor$ and $h = 3k' + 2$.
56: **end if**

/ Bounding μ and calling REMOVE */*
57: Find $s \geq 0$ such that $2^sQ > 2^{p+4}B$ and put $T = 2^sA + B\lfloor 2^s\sqrt{D} \rfloor$.
58: $(\mathfrak{b}, e', h') = \text{REMOVE}((\mathfrak{b}, e, h), T, C, s, p)$.

/ Re-establishing the w-near property */*
59: $(\mathfrak{c}, d, k) = \text{WNEAR}((\mathfrak{b}, e', h'), w, p)$.

Algorithm A.5. WDIV

Input: $(\mathfrak{b}', d', k')$, $(\mathfrak{b}'', d'', k'')$, p where $(\mathfrak{b}', d', k')$ is a w-near (f', p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}'$ and $(\mathfrak{b}'', d'', k'')$ is a w-near (f'', p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}''$ dividing $\mathfrak{a}'$. Here,

$$\mathfrak{b}' = [Q'/r, (P' + \sqrt{D})/r] \text{ and } \mathfrak{b}'' = [Q''/r, (P'' + \sqrt{D})/r].$$

Output: $(\mathfrak{c}, d, k)$, a reduced $(f^{**} + 13/4, p)$ representation of $\mathfrak{a}'(\mathfrak{a}'')^{-1}$ where $f^{**} = f' + f''/(1 - f''2^{-p}) + f'f''/(2^p - f'')$.

/ Computing a reduced ideal* $\mathfrak{b}$ *and* $\mu = (A + B\sqrt{D})/C \in \mathbb{K}$ *where* $(\mu)\mathfrak{b} = \mathfrak{b}'\overline{\mathfrak{b}''}$*. */*

1: Let $\mathfrak{b}^* = [Q''/r, (qQ'' - P'' + \sqrt{D})/r]$ where $q = \lfloor (P'' + \sqrt{D})/Q'' \rfloor$.
2: **if** $Q' \geq Q''$ **then**
3: Compute $(\mathfrak{b}, A, B, C) = \text{NUCOMP}(\mathfrak{b}', \mathfrak{b}^*)$ where $\mathfrak{b} = [Q/r, (P+\sqrt{D})/r]$.
4: **else**
5: Compute $(\mathfrak{b}, A, B, C) = \text{NUCOMP}(\mathfrak{b}^*, \mathfrak{b}')$ where $\mathfrak{b} = [Q/r, (P+\sqrt{D})/r]$.
6: **end if**

/ Computing* e, h *for which* $(\mathfrak{b}''\overline{\mathfrak{b}''}, e, h)$ *is an* $(1 + f^{**}, p)$ *representation of* $\mathfrak{a}(\mathfrak{a}'')^{-1}$ *using Theorem 3. */*

7: Find κ such that $2^\kappa < N(\mathfrak{b}'') = Q''/r \leq 2^{\kappa+1}$.
8: **if** $d''2^{\kappa-1} < d'N(\mathfrak{b}'') \leq d''2^\kappa$ **then**
9: $e = \lceil 2^{p-\kappa+1} d'N(\mathfrak{b}'')/d'' \rceil$
10: $h = k' - k'' + \kappa - 1$
11: **else if** $d''2^\kappa < d'N(\mathfrak{b}'') \leq d''2^{\kappa+1}$ **then**
12: $e = \lceil 2^{p-\kappa} d'N(\mathfrak{b}'')/d'' \rceil$
13: $h = k' - k'' + \kappa$
14: **else**
15: $e = \lceil 2^{p-\kappa-1} d'N(\mathfrak{b}'')/d'' \rceil$
16: $h = k' - k'' + \kappa + 1$.
17: **end if**

/ Computing* e', h' *for which* $(\mathfrak{b}, e', h')$ *is an* $(17/8 + f^{**}, p)$ *representation of* $\mathfrak{a}'(\mathfrak{a}'')^{-1}$*. */*

18: Find $s \geq 0$ such that $2^s Q > 2^{p+4} B$.
19: Put $T = 2^s A + B\lfloor 2^s \sqrt{D} \rfloor$.
20: $(\mathfrak{b}, e', h') = \text{REMOVE}((\mathfrak{b}, e, h), T, C, s, p)$

/ Computing a reduced* w*-near* $(13/4 + f^{**}, p)$ *representation of* $\mathfrak{a}'(\mathfrak{a}'')^{-1}$*. */*

21: $(\mathfrak{c}, d, k) = \text{WNEAR}((\mathfrak{b}, e', h'), w, p)$

Algorithm A.6. BINEXP [15, Algorithm 11.4, pp. 276-277]

Input: $(\mathfrak{b}_0, d_0, k_0), n, w, p$ where $(\mathfrak{b}_0, d_0, k_0)$ is a w-near (f_0, p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}$ and $n \in \mathbb{N}$.

Output: A w-near (f, p) representation $(\mathfrak{b}, d, k)$ of $\mathfrak{a}^n$ for some $f \in [1, 2^p)$.

Compute $(n_0, \ldots, n_{\ell_B}) = \text{BIN}(n)$.
Set $(\mathfrak{b}, d, k) = (\mathfrak{b}_0, d_0, k_0)$.
for $i = 1 \to \ell_B$ **do**
  $(\mathfrak{b}, d, k) \leftarrow$WDUPL$((\mathfrak{b}, d, k), w, p)$
  if $n_i = 1$ **then**
    $(\mathfrak{b}, d, k) \leftarrow$WMULT$((\mathfrak{b}, d, k), (\mathfrak{b}_0, d_0, k_0), w, p)$
  end if
end for

Algorithm A.7. NAFEXP

Input: $(\mathfrak{b}_0, d_0, k_0), n, w, p$ where $(\mathfrak{b}_0, d_0, k_0)$ is a w-near (f_0, p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}$ and $n \in \mathbb{N}$.

Output: A w-near (f, p) representation $(\mathfrak{b}, d, k)$ of $\mathfrak{a}^n$ for some $f \in [1, 2^p)$.

Compute $(n_0, \ldots, n_{\ell_N}) = \text{NAF}(n)$.
Set $(\mathfrak{b}, d, k) = (\mathfrak{b}_0, d_0, k_0)$.
for $i = 1 \to \ell_N$ **do**
  $(\mathfrak{b}, d, k) \leftarrow$WDUPL$((\mathfrak{b}, d, k), w, p)$
  if $n_i = 1$ **then**
    $(\mathfrak{b}, d, k) \leftarrow$WMULT$((\mathfrak{b}, d, k), (\mathfrak{b}_0, d_0, k_0), w, p)$
  else if $n_i = -1$ **then**
    $(\mathfrak{b}, d, k) \leftarrow$WDIV$((\mathfrak{b}, d, k), (\mathfrak{b}_0, d_0, k_0), w, p)$
  end if
end for

Algorithm A.8. SB3EXP

Input: $(\mathfrak{b}_0, d_0, k_0), n, w, p$ where $(\mathfrak{b}_0, d_0, k_0)$ is a w-near (f_0, p) representation of an invertible $\mathcal{O}$-ideal $\mathfrak{a}$ and $n \in \mathbb{N}$.

Output: A w-near (f, p) representation $(\mathfrak{b}, d, k)$ of $\mathfrak{a}^n$ for some $f \in [1, 2^p)$.

Compute $(n_0, \ldots, n_{\ell_S}) = \text{SB3}(n)$.
Set $(\mathfrak{b}, d, k) = (\mathfrak{b}_0, d_0, k_0)$.
for $i = 1 \to \ell_S$ **do**
  $(\mathfrak{b}, d, k) \leftarrow$WCUBE$((\mathfrak{b}, d, k), w, p)$
  if $n_i = 1$ **then**
    $(\mathfrak{b}, d, k) \leftarrow$WMULT$((\mathfrak{b}, d, k), (\mathfrak{b}_0, d_0, k_0), w, p)$
  else if $n_i = -1$ **then**
    $(\mathfrak{b}, d, k) \leftarrow$WDIV$((\mathfrak{b}, d, k), (\mathfrak{b}_0, d_0, k_0), w, p)$
  end if
end for