Advances in Mathematics of Communications
Volume 1, No. 2, 2007, 197–221
Web site: http://www.aimSciences.org

CRYPTOGRAPHIC PROTOCOLS ON REAL HYPERELLIPTIC CURVES

M. J. Jacobson

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

R. Scheidler

Department of Mathematics and Statistics
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4

A. Stein

Department of Mathematics
University of Wyoming
1000 E. University Avenue
Laramie, WY 82071-3036, USA

(Communicated by Edlyn Teske)

Abstract. We present public-key cryptographic protocols for key exchange, digital signatures, and encryption whose security is based on the presumed intractability of solving the principal ideal problem, or equivalently, the distance problem, in the real model of a hyperelliptic curve. Our protocols represent a significant improvement over existing protocols using real hyperelliptic curves. Theoretical analysis and numerical experiments indicate that they are comparable to the imaginary model in terms of efficiency, and hold much more promise for practical applications than previously believed.

1. Introduction and Motivation

Cryptographic key exchange à la Diffie-Hellman has been proposed with a variety of different underlying group and group-like structures. Finite fields and elliptic curves have found their way into commercial applications, but in recent years, hyperelliptic curves of low genus have also become an increasingly popular choice. In 1989, Koblitz [16] first proposed the Jacobian of the conventional, i.e. imaginary, model of a hyperelliptic curve for key exchange. Several years later, an analogous key exchange protocol was presented for the real model of a hyperelliptic curve [23]. Its underlying key space was the set of reduced principal ideals in the coordinate ring of the curve, together with its group-like infrastructure. Unfortunately, the protocol [23] was significantly slower and more complicated than its imaginary

2000 *Mathematics Subject Classification:* Primary: 94A60, 14H45; Secondary: 14Q05.
Key words and phrases: Discrete logarithm based cryptography, real hyperelliptic curve, reduced divisor, infrastructure and distance, efficient implementation.
The first two authors are supported in part by NSERC of Canada.

cousin [16], while offering no additional security; the same was true for subsequent modifications presented in [22].

In this paper, we present improved public-key protocols for key exchange, digital signatures, and encryption. These schemes are based on the presumed intractability of computing the distance of a reduced principal divisor of a real hyperelliptic curve. Our protocols make use of two new scalar multiplication primitives that lead to significant improvements over the key exchange protocols in [23, 22]. These primitives make use of the fact that the infrastructure of a real hyperelliptic curve admits two operations. This is in contrast to the Jacobian of an imaginary hyperelliptic curve which admits only the "giant step" operation, i.e. divisor addition with subsequent reduction. In addition to giant steps, the infrastructure supports a much faster "baby step" operation that is analogous to a Gaussian reduction step as used in the imaginary setting, and can be used to jump from one reduced divisor to the next. We show, for example, how on average one eighth of the giant steps in the original key exchange protocol [23] can be replaced by baby steps, with an additional minor expense of just a few more baby steps. Since baby steps require linear time in the genus of the curve, whereas giant steps take quadratic time in general, we would expect that this speeds up the protocol by almost twelve percent over the procedures given in [23]. We show that similar results hold in most cases for the other protocols, the most significant improvement occurring for our digital signature protocol based on DSA, in which generating signatures is especially efficient.

Our protocols work for any real hyperelliptic curve and any genus and do not require any precomputed divisors. Assuming that giant steps in the real and imaginary models cost the same, a similar speed-up of the real setting as compared to the imaginary setting can be proved theoretically, even for genus as low as 2. Preliminary numerical data, using an optimized generic version of Cantor's algorithm for divisor arithmetic and a basic version of non-adjacent form (NAF) based scalar multiplication, indicate that the performance of our improved real protocols is comparable to those using the imaginary model.

However, there are issues that need to be addressed before a rigorous performance comparison between the real and imaginary settings can be obtained. There has been a lot of work in algorithmic improvement to the imaginary protocols that has not yet been generalized to the real mode. For example, in the low genus case, it needs to be seen how optimized explicit formulas for the real case compare to the explicit formulas in the imaginary case [17, 29, 21]. We expect that the explicit formulas for the giant step operation in the real case are comparable to those in the imaginary case and that, as in [26, 27], baby steps will be much faster than giant steps. If so, our new protocols would perform almost as well as the corresponding protocols in the imaginary case, even using explicit formulas for the small genus cases. The first results in this direction [6], in which explicit formulas for genus two arithmetic over prime fields using affine representation is presented, bear this out. It also remains to be seen how special constructions such as the one given in [8] carry over to the real setting. Finally, our protocols do not require any precomputations based on the base divisor, so we compare with imaginary protocols using NAF-based scalar multiplication. We expect that various well-known improvements to scalar multiplication involving precomputations such as window methods and joint-sparse forms, especially those that take advantage of a fixed divisor, can be generalized to the real case, but the development and comparison of these variations is beyond

the scope of this paper. Our goal is not to argue that protocols in the real model are faster than those in the imaginary model at this point, but rather that their performance is comparable and that further investigation is warranted.

We begin with a description of the set of divisors that represents the basic mathematical structure underlying several major cryptographic protocols using both imaginary and real hyperelliptic curves in Section 2. We provide only the very basics of the infrastructure in the real setting; a full and detailed divisor theoretic treatment of this phenomenon was provided in [13]. We describe our improved scalar multiplication primitives for real hyperelliptic curves in Section 3. Cryptographic protocols based on these primitives are presented in Section 4, including a discussion of their efficiency as compared to previous versions in the real case as well as protocols in the imaginary case. Security issues are touched on in Section 5, and numerical data, generated using generic algorithms for divisor arithmetic as described in Appendix A, are presented in Section 6. As some of this is still work in progress, we offer conclusions and some open problems in Section 7.

2. Hyperelliptic Curves

A considerable amount of literature has been devoted to hyperelliptic curves and their cryptographic applications. We therefore simply refer the reader to the elementary introduction [18], the survey article [12], and the description of both imaginary and real models in [13], and review only the basics and notation here.

Throughout this paper, let C be a hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$. That is, C is an absolutely irreducible non-singular curve of the form

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

where $f, h \in \mathbb{F}_q[x]$ and $h = 0$ if q is odd. We distinguish two scenarios[1] (see [12, Section 2.9], [5], and [13]):

- *Imaginary model/form.* f is monic, $\deg(f) = 2g + 1$, and $\deg(h) \leq g$ if q is even;
- *Real model/form.* If q is odd, then f is monic and $\deg(f) = 2g+2$. If q is even, then h is monic, $\deg(h) = g+1$, and either $\deg(f) \leq 2g+1$, or $\deg(f) = 2g+2$ and the leading coefficient of f is of the form $e^2 + e$ for some $e \in \mathbb{F}_q^*$.

We denote by $\mathbb{F}_q[C] = \mathbb{F}_q[x, y]$ the coordinate ring and by $\mathbb{F}_q(C) = \mathbb{F}_q(x, y)$ the function field of C.

It is easy to convert an imaginary hyperelliptic curve to a real model over the same base field and, if the curve has an $\mathbb{F}_q$-rational point, vice versa. Explicit formulas for relating divisor arithmetic in the two models were given in [20] for the case of odd characteristic; for q even, analogous variable transformations apply.

2.1. Imaginary Setting.

If C as given in (2.1) is imaginary, then the Jacobian of C over $\mathbb{F}_q$ has been widely used for cryptographic applications. Here, the pole ∞ of x is totally ramified in $\mathbb{F}_q(C)$, and every degree zero divisor D defined over $\mathbb{F}_q$ can be uniquely written in the form $D = D_x - \deg(D_x)\infty$ where D_x is a divisor not supported at ∞. A degree zero divisor D defined over $\mathbb{F}_q$ is *semi-reduced* if D_x is an effective divisor that is not divisible by the conorm of any divisor in $\mathbb{F}_q(x)$, and *reduced* if in addition $\deg(D_x) \leq g$. The finite divisors D_x, and hence the

[1] The conditions for the cases where $\deg(f) = 2g+2$ given here and in [12] are slightly different from those given in [5], but the two can easily be shown to be equivalent by simply multiplying the curve by a suitable scalar in $\mathbb{F}_q$.

degree zero divisors D, are in one-to-one correspondence with the fractional ideals of $\mathbb{F}_q[C]$, with effective divisors corresponding to integral ideals, semi-reduced divisors to principal ideals, and reduced divisors to reduced ideals. Every non-trivial semi-reduced divisor D has an explicit representation as a pair $a, b \in \mathbb{F}_q[x]$ of polynomials where a is unique and b is unique modulo a. Here, a is monic and divides $f + bh - b^2$. If b is chosen so that $\deg(b) < \deg(a)$, then the pair (a, b) is called the *Mumford representation* of D. We write $D = \operatorname{div}(a, b)$ and point out that in this fashion, divisor arithmetic can be reduced to simple polynomial arithmetic over $\mathbb{F}_q$.

It is well-known that every degree zero divisor class has a unique reduced representative. If $\mathcal{R}$ denotes the set of these reduced representatives, then we have an operation "$\oplus$" on $\mathcal{R}$ via $(D, D') \to D \oplus D'$ where $D \oplus D'$ is the unique reduced representative in the divisor class of $D + D'$. $\mathcal{R}$ is a group under this operation, where the inverse of a divisor $D = \operatorname{div}(a, b) \in \mathcal{R}$ is the *conjugate* (or *opposite*) divisor $\overline{D} = \operatorname{div}(a, -b - h)$, i.e. the image of D under the natural involution $(x, y) \to (x, -y - h(x))$ on C; note that $\deg(D_x) = \deg(\overline{D}_x) = \deg(a)$. We remark that $-b - h$ is reduced modulo a if necessary, so that the conjugate divisor is also in Mumford representation.

For reasons that will become evident later on, we refer to the operation $\oplus$ as a *giant step*. The conventional method for performing giant steps is divisor addition with subsequent Gaussian reduction (Cantor's algorithm [3]), although other more efficient methods such as NUCOMP [14] and explicit formulas for low genus curves [17, 29, 21] exist. For definitions of the terminology used above as well as other details, we refer the reader to [3] and [16], or to the sources [18], [12] and [13] mentioned earlier.

2.2. Real Setting. Details on the arithmetic on the real model of a hyperelliptic curve can be found for example in [28], [22], or [12], but all these sources employ only the terminology of ideals in the coordinate ring $\mathbb{F}_q[C]$ of C. Instead, we relate their descriptions to a divisor theoretic framework as given in [13]. We let C as given in (2.1) be real, and denote by ∞_1 and ∞_2 the two poles of x with respective normalized additive valuations ν_1 and ν_2. The class of the divisor $\infty_1 - \infty_2$ has degree zero and finite order R_x, the *x-regulator* of $\mathbb{F}_q(C)/\mathbb{F}_q(x)$. The divisor $R_x(\infty_1 - \infty_2)$ is thus principal and is the divisor of a *fundamental unit* of $\mathbb{F}_q(C)$, i.e. a generator of the infinite cyclic group $\mathbb{F}_q[C]^*/\mathbb{F}_q^*$.

The completion of $\mathbb{F}_q(C)$ with respect to both ∞_1 and ∞_2 is the field $\mathbb{F}_q\langle x^{-1}\rangle$ of Puiseux series in x^{-1} over $\mathbb{F}_q$, so there are two embeddings of $\mathbb{F}_q(C)$ into $\mathbb{F}_q\langle x^{-1}\rangle$, given by the respective Puiseux expansions of y and $\overline{y} = -y - h$. We choose an embedding ν so that $\nu(y) = -g - 1$ and number the two poles of x so that $\nu = \nu_1$.

In a similar manner to the imaginary case, every degree zero divisor of C can be uniquely written in the form

$$D = D_x - \deg(D_x)\infty_2 + \nu_1(D)(\infty_1 - \infty_2) \ ,$$

where D_x is a divisor that is coprime to both ∞_1 and ∞_2. Reducedness and semi-reducedness of degree zero divisors D defined over $\mathbb{F}_q$ can be defined via D_x exactly as in the imaginary case. Analogous to the imaginary setting, every semi-reduced divisor D can be described by a pair of polynomials a, b as described in Section 2.1, together with the value $\nu_1(D)$. As a point of interest, we mention that by Proposition 4.1 of [20], every degree zero divisor class has exactly one reduced representative D with $0 \le \nu_1(D) \le g - \deg(D_x)$. However, cryptographic protocols using arithmetic on these reduced representatives are less efficient than their imaginary counterparts.

Once again, we have the same bijective correspondence between finite divisors and fractional $\mathbb{F}_q[C]$-ideals. Rather than performing arithmetic in the Jacobian via unique representatives as in the imaginary case, we focus instead on one *ideal* class — usually the principal ideal class — and consider a certain set of pairwise non-equivalent reduced divisors arising from all the reduced ideals in this ideal class. Fix any reduced ideal $\mathfrak{a}$ of $\mathbb{F}_q[C]$, and define $\mathcal{R}_\mathfrak{a}$ to be the finite set of reduced divisors $D = D_x - \deg(D_x)\infty_2$ such that D_x corresponds to a reduced ideal $\mathfrak{b}$ equivalent to $\mathfrak{a}$ and $\nu_1(D) = 0$; write $D = D_\mathfrak{b}$. Let $\alpha \in \mathbb{F}_q(C)$ with $\mathfrak{b} = (\alpha)\mathfrak{a}$. Then $D_\mathfrak{b} = D_\mathfrak{a}$, or equivalently, $\mathfrak{b} = \mathfrak{a}$, if and only if $\nu_1(\alpha) \equiv 0 \pmod{R_x}$ (see for example [23]). It follows that the set $\mathcal{R}_\mathfrak{a}$ can be ordered by *distance* $\delta(D_\mathfrak{b}) = -\nu_1(\alpha)$ where α is chosen so that $0 \le -\nu_1(\alpha) < R_x$. Hence, if we set $r = |\mathcal{R}_\mathfrak{a}|$, we can write

$$\mathcal{R}_\mathfrak{a} = \{D_1, D_2, \ldots, D_r\}$$

with $D_1 = D_\mathfrak{a}$, and set $\delta_i = \delta(D_i)$ for $1 \le i \le r$. Then $\delta_1 = 0$, and δ_i strictly increases in the interval $[0, R_x[$ as i increases from 1 to r.

Since each $D \in \mathcal{R}_\mathfrak{a}$ is uniquely determined by its corresponding ideal, i.e. by its finite part D_x, D can be solely described by two polynomials $a, b \in \mathbb{F}_q[x]$ as explained above for the imaginary scenario, and we are justified in writing $D = \operatorname{div}(a, b)$. Each $D \in \mathcal{R}_\mathfrak{a}$ is also uniquely determined by its distance $\delta(D)$, but in practice, it is infeasible to actually determine $\delta(D)$ from a and b; in fact, the security of our cryptographic schemes is based on that very fact.

An operation similar to Gaussian reduction steps as used for divisor reduction on imaginary hyperelliptic curves can be applied to divisors in $\mathcal{R}_\mathfrak{a}$, except here, such a step moves from D_i to D_{i+1} $(1 \le i < r)$ and is referred to as a *baby step*. More exactly, if $D_i = \operatorname{div}(a_{i-1}, b_{i-1}) \in \mathcal{R}_\mathfrak{a}$, then a baby step produces the divisor $D_{i+1} = \operatorname{div}(a_i, b_i)$, where

$$q_{i-1} = \left\lfloor \frac{b_{i-1} + y}{a_{i-1}} \right\rfloor \ , \quad b_i = q_{i-1}a_{i-1} - b_{i-1} + h \ , \quad a_i = \frac{f + hb_i - b_i^2}{a_{i-1}} \ . \tag{2.2}$$

Here, $\lfloor \theta \rfloor$ denotes the polynomial part of $\theta \in \mathbb{F}_q\langle x^{-1} \rangle$. The baby step $D_i \to D_{i+1}$ corresponds to one step in the simple continued fraction expansion of the Puiseux series of $(b_0 + y)/a_0 \in \mathbb{F}_q\langle x^{-1} \rangle$. More efficient formulas for computing a baby step are presented in Appendix A. Using those formulas, the running time of a baby step is linear, i.e. $O(g)$ field operations in $\mathbb{F}_q$.

Writing $D_j = D_{\mathfrak{b}_j}$ for $1 \le j \le r = |\mathcal{R}_\mathfrak{a}|$, we have $\mathfrak{b}_{i+1} = (\beta_i)\mathfrak{b}_i$ where $\beta_i = -b_i/(a_i - y - h) = (b_i + y)/a_{i-1}$ (see again [23]). Therefore,

$$\delta_{i+1} - \delta_i = -\nu_1(\beta_i) = g + 1 - \deg(a_{i-1}) = \deg(q_{i-1}) \ . \tag{2.3}$$

So even if δ_{i+1} and δ_i are unknown, their difference can be found at virtually no extra cost with the baby step $D_i \to D_{i+1}$. Note that if $D_i = \mathbf{0} = \operatorname{div}(1, 0)$, then $q_{i-1} = \lfloor y \rfloor$, so $\delta_{i+1} = g + 1$; in all other cases, we have $\deg(q_{i-1}) \le g$ and hence $\delta_{i+1} \le \delta_i + g$.

Note that $r - 1$ iterations of (2.2), applied to any divisor in $\mathcal{R}_\mathfrak{a}$, generates all of $\mathcal{R}_\mathfrak{a}$. Baby steps can also be performed on semi-reduced divisors D, in which case at most $\lceil (\deg(D_x) - g)/2 \rceil$ baby steps produce a reduced divisor equivalent to D. We can now define giant steps $(D, D') \to D \oplus D'$ on $\mathcal{R}_\mathfrak{a}$ using almost the same arithmetic as in the imaginary case. The standard way of implementing a giant step is to compute the sum $E = D + D'$ using, for example, Cantor's algorithm [3], and then applying baby steps (2.2) to E until the first reduced divisor is reached; this divisor is $D \oplus D'$. Just as in the imaginary case, $D \oplus D'$ is reached after at most

$\lceil(\deg(E_x) - g)/2\rceil \leq \lceil g/2\rceil$ steps of (2.2) because $\deg(E_x) = \deg(D_x) + \deg(D'_x) \leq 2g$. The running time of a giant step is quadratic, i.e. $O(g^2)$ operations in $\mathbb{F}_q$ (see [26] for a more exact complexity analysis of both baby steps and giant steps).

The structure underlying all our real hyperelliptic curve cryptographic protocols is the set $\mathcal{R} = \mathcal{R}_{\mathfrak{a}}$ of reduced divisors $D_{\mathfrak{b}}$ where $\mathfrak{b}$ is a reduced *principal* ideal; here, we choose $\mathfrak{a} = (1) = \mathbb{F}_q[C]$ to be the trivial ideal, so $D_1 = \mathbf{0} = \operatorname{div}(1, 0)$ is the trivial divisor. Note that $\mathcal{R}$ is closed under conjugation, since the conjugate ideal of a principal ideal is again principal, and under giant steps, since the product of two principal ideals is again principal and baby steps preserve ideal equivalence.

In order to be able to conduct cryptography in $\mathcal{R}$ securely, we need to ensure that $\mathcal{R}$ has large cardinality. A baby step applied to the last divisor $D_{|\mathcal{R}|}$ generates a divisor D with $\nu_1(D) = -R_x$. Since $\delta_1 = 0$, $\delta_2 = g + 1$, and $1 \leq \delta_{i+1} - \delta_i \leq g$ for $2 \leq i \leq |\mathcal{R}| - 1$, we have

$$g + i - 1 \leq \delta_i \leq (i - 1)g + 1 \text{ for } 2 \leq i \leq |\mathcal{R}| \ ,$$

and hence $g + |\mathcal{R}| \leq R_x \leq |\mathcal{R}|g + 1$ as $\delta_{|\mathcal{R}|+1} = R_x$. The Hasse-Weil bounds imply that the order h of the Jacobian of the curve C over $\mathbb{F}_q$ is of order q^g, and R_x is a divisor of h. Most of the time, the ratio h/R_x, which is the ideal class number of $\mathbb{F}_q[C]$, is very small by the Friedman-Washington-Achter theorems; heuristics on these class numbers were formulated by Friedman and Washington [7] and recently proved by Achter [1]. In fact, the curve C can be chosen so that R_x is large with high probability, see [12]. It follows that R_x, and hence the cardinality of $\mathcal{R}$, is of magnitude q^g, i.e. exponentially large in the size of the curve C over $\mathbb{F}_q$, and comparable to the cardinality of the set $\mathcal{R}$ of reduced divisors defined for the imaginary setting.

It will be important to understand how distances behave under the operations on $\mathcal{R}$. We already saw that each baby step advances the distance by at least 1 and at most g, except for the first baby step which jumps from distance 0 to $g + 1$. To see what happens to the distance under conjugation, consider any divisor $D_{\mathfrak{b}} = \operatorname{div}(a, b) \in \mathcal{R}$ with corresponding ideal $\mathfrak{b}$. Then $\overline{\mathfrak{b}} = (a)\mathfrak{b}^{-1}$, so

$$\delta(D_{\overline{\mathfrak{b}}}) \equiv -\nu_1(a) - \delta(D_{\mathfrak{b}}) \equiv \deg(a) - \delta(D_{\mathfrak{b}}) \pmod{R_x} \ .$$

Note that $D_{\overline{\mathfrak{b}}}$ is not actually the conjugate divisor of $D_{\mathfrak{b}}$; rather, $D_{\overline{\mathfrak{b}}} = \overline{D_{\mathfrak{b}}} + \deg(a)(\infty_1 - \infty_2)$, so $\nu_1(\overline{D_{\mathfrak{b}}}) \neq 0$. For simplicity, we henceforth abuse the overline notation and identify $D_{\overline{\mathfrak{b}}} = \operatorname{div}(a, -b - h)$ with $\overline{D_{\mathfrak{b}}}$.

Finally, while $\mathcal{R}$ is closed under giant steps, it is not associative, i.e. it is not necessarily the case that $(D \oplus D') \oplus D'' = D \oplus (D' \oplus D'')$ for $D, D', D'' \in \mathcal{R}$. However, $\mathcal{R}$ is "almost" associative in the sense that the operation $\oplus$ is almost distance preserving. More exactly, we have

$$\delta(D \oplus D') = \delta(D) + \delta(D') - d \ \text{ where } \ 0 \leq d \leq 2g \ , \tag{2.4}$$

and d can be efficiently computed (see, for example, Theorem 3.7 of [28]). Since distances tend to be of order of magnitude R_x, i.e. of magnitude q^g by our previous remarks, they are exponentially large compared to the "error" d in (2.4) arising from the reduction steps. It follows that the distance of the divisor $D \oplus D'$ is extremely close to, and just below, the sum of the distances of the divisors D and D', with a "shortfall" d of at most $2g$. This behavior is referred to as the *infrastructure* of $\mathcal{R}$. In addition, the use of the terms "baby step" and "giant step" is now justified: the former yields a very small advance in distance, namely at most linear in g, whereas the latter generally results in an exponentially large distance jump.

Finally, we define for any integer m with $0 \leq m < R_x$ the divisor in $\mathcal{R}$ *below* m to be the unique divisor $D_i \in \mathcal{R}$ with $\delta_i \leq m < \delta_i$. Given two divisors $D, D' \in \mathcal{R}$, the divisor $E \in \mathcal{R}$ below $\delta(D) + \delta(D')$ can now be found by computing the giant step $E' = D \oplus D'$ and the error $d = \delta(D) + \delta(D') - \delta(E')$, and then applying at most d baby steps to E', increasing d by the appropriate advance in distance in each baby step, until d changes sign from negative or zero to positive. This can be done without knowledge of the distances $\delta(D)$ and $\delta(D')$, and requires one giant step and at most $2g$ baby steps by (2.4).

3. Scalar Multiplication

Most exponentiation/scalar multiplication based cryptographic schemes employ three types of algorithms. In the first scenario, a user performs exponentiation/scalar multiplication using some fixed known scalar. In the second case, each participant performs exponentiation/scalar multiplication on the same fixed group element, but with different exponents/scalars. Finally, in the third type of algorithm, all parties perform exponentiation/scalar multiplication on different group elements. For example, the fixed and variable group element scenarios are used in the first and second round of Diffie-Hellman key exchange, respectively, and the fixed scalar situation occurs in the decryption procedures of the IES and PSEC protocols (see pp. 189-191 of [11]).

Arithmetic using fixed exponents/scalars can be optimized using addition chains, at the expense of precomputation. Arithmetic using a fixed base can also be improved if precomputations involving the base are permitted, for example, using windowing methods. A thorough analysis of these scenarios, and generalization to the real case, is beyond the scope of this paper. Our primary focus is on arithmetic not requiring any precomputation, so we will henceforth only consider the variable and fixed base cases without precomputation.

In the setting of imaginary hyperelliptic curves, both the fixed and the variable group element scenarios require the computation of the reduced divisor in the class of nD, given a reduced divisor D and an integer n. For real hyperelliptic curves, the fixed base scenario corresponds to finding the divisor in $\mathcal{R}$ below a given distance (*fixed distance case*), while the variable base algorithm corresponds to the situation of generating from an integer n and a divisor $D \in \mathcal{R}$ of unknown distance $\delta(D)$ the divisor in $\mathcal{R}$ below $n\delta(D)$ (*variable distance case*).

To perform these operations on both real and imaginary hyperelliptic curves, we make use of the non-adjacent form (NAF) representation of the scalar in question. Every positive integer n has a unique representation

$$n = \sum_{i=0}^{l} b_i 2^{l-i} \ ,$$

where $b_0 \neq 0$, $b_i \in \{-1, 0, 1\}$, and no two consecutive digits b_i are non-zero. The NAF of n can be computed using, for example, Algorithm 3.30, p. 98, of [11]. Since $n > 0$, we have $b_0 = 1$. Note also that $2^{l+1} < 3n < 2^{l+2}$, so the length of the NAF representation of n is at most one more than that of the binary representation of n. If n is randomly chosen, then one third of all the digits in the NAF of n are expected to be non-zero, as opposed to one half of the bits for the usual binary representation of n (see p. 98 of [11]).

We make use of the fact that in both the real and the imaginary case, "inversion" of reduced divisors is essentially free: given any divisor $D \in \mathcal{R}$, the divisor $\overline{D}$ again

belongs to $\mathcal{R}$ and can be computed from D in constant time. This fact allows us to use NAF for all our scalar multiplication algorithms.

3.1. Imaginary Case. For completeness, we review scalar multiplication in the Jacobian of an imaginary hyperelliptic curve. The following algorithm SCALAR-MULT computes the reduced divisor in the class of nD, given a reduced divisor D, using the NAF representation of n via a generalization of the binary double and add method in which a coefficient of -1 indicates an addition by the conjugate of D.

SCALAR-MULT(D, n)
Input: $D \in \mathcal{R}$, $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
Output: The reduced divisor in the class of nD.
Algorithm:
1. set $E = D$;
2. for $i = 1$ to l do
   2.1. replace E by $E \oplus E$;
   2.2. if $b_i = 1$ then replace E by $E \oplus D$;
   2.3. if $b_i = -1$ then replace E by $E \oplus \overline{D}$;
3. output E.

This algorithm requires l doubles and on average $l/3$ adds. We use the terms "double" and "add" for our operation counts, bearing in mind that the underlying operation is not an actual doubling or addition of divisors, but a giant step, i.e. divisor addition followed by reduction. Basically, this corresponds to doubling, respectively, adding of divisor classes and enforcing a unique representation of each group element via reduction.

3.2. Real Case. We begin with the "variable distance" scenario: given a positive integer n and a divisor $D \in \mathcal{R}$ of unknown distance $\delta(D)$, find the divisor in $\mathcal{R}$ below $n\delta(D)$. Algorithm VAR-DIST1 accomplishes this using a natural generalization of Algorithm SCALAR-MULT. The idea is that the distance of nD will be close to $n\delta(D)$ as long as, after each giant step, we apply a small number of baby steps in order to compensate for the distance lost after reduction. This algorithm is essentially the same as POWERDIST in [23] and R-EXP in [22].

VAR-DIST1(D, n)
Input: $D \in \mathcal{R}$, $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
Output: The divisor in $\mathcal{R}$ below $n\delta(D)$.
Algorithm:
1. set $E = D$;
2. for $i = 1$ to l do
   2.1. compute $E' = E \oplus E$;
   2.2. apply baby steps to E' to compute the divisor in $\mathcal{R}$ below $2\delta(E)$ and call the result E;
   2.3. if $b_i \neq 0$ then
      2.3.1. if $b_i = 1$ then set $D' = D$;
      2.3.2. if $b_i = -1$ then set $D' = \overline{D}$;
      2.3.3. compute $E' = E \oplus D'$;
      2.3.4. apply baby steps to E' to compute the divisor in $\mathcal{R}$ below $\delta(E)+\delta(D')$ and call the result E;
3. output E.

We see that this algorithm is very similar to scalar multiplication on imaginary hyperelliptic curves; the main difference being the "adjustment" baby steps in steps 2.2 and 2.3.4. Let Δ be an upper bound on d, the giant step distance "errors" caused by reduction, i.e. an upper bound on the number of baby steps that need to be performed in each of these steps. We see that VAR-DIST1 requires l doubles and $l/3$ adds, plus an additional $4\Delta l/3$ baby steps, where $\Delta \geq d = 2\delta(E) - \delta(E')$ in step 2.2 and $\Delta \geq d = \delta(E) + \delta(D') - \delta(E')$ in step 2.3.4. By (2.4) we have $\Delta \leq 2g$. Note that in order to find the divisors below $2\delta(E)$ and $\delta(E) + \delta(D')$ in steps 2.2 and 2.3.4, respectively, it is necessary to explicitly compute d in order to know how many "adjustment" baby steps to apply. These facts make Algorithm VAR-DIST1 slower than the corresponding algorithm SCALAR-MULT from the imaginary case.

A simple modification of this algorithm handles the "fixed distance" scenario: given an integer n, find the divisor in $\mathcal{R}$ below n. Here, we recall the fact that the divisor $D_2 \in \mathcal{R}$ has distance $\delta_2 = g + 1$. We first need to apply a division with remainder to obtain the quantity $s = \lfloor n/(g+1) \rfloor$ in order to convert s to NAF. The procedure below is essentially algorithm R-BELOW in [22].

FIXED-DIST1(n)
Input: $n \in \mathbb{N}$, $s = \lfloor n/(g+1) \rfloor$ given in NAF.
Output: The divisor in $\mathcal{R}$ below n.
Algorithm:

1. compute $E' = \text{VAR-DIST1}(D_2, s)$ // $E' \in \mathcal{R}$ is the divisor below $s(g+1)$;
2. apply at most $n - s(g+1)$ baby steps to E' to compute the divisor $E \in \mathcal{R}$ below n;
3. output E.

Since $0 \leq n - s(g+1) \leq g$, this algorithm performs roughly $\log_2(g+1)$ fewer doubles and adds than Algorithm VAR-DIST1, but one additional division with remainder and at most g baby steps. Once again, this makes this algorithm generally significantly slower than scalar multiplication of divisors on imaginary hyperelliptic curves.

3.3. Improvements to the Real Case. We now outline our improvements to the real case. For the variable distance situation, we eliminate all adjustment baby steps by performing only a few baby steps at the beginning of the algorithm; this number of baby steps is independent of the size of the scalar and only depends on g. As a result, the number of doubles and adds is the same as for scalar multiplication on imaginary hyperelliptic curves, and the number of additional baby steps required is reduced significantly. For the fixed distance case, we replace the giant steps in step 2.3.3 of VAR-DIST1 by baby steps and eliminate all the adjustment baby steps. This results in a significant reduction in the number of adds compared to both FIXED-DIST1 and the imaginary scenario, i.e. SCALAR-MULT.

The key to our improvements is the following heuristics, which allow us to predict with high probability the relative distance between two divisors. We henceforth write D_+ for the divisor obtained by applying one baby step to the divisor $D \in \mathcal{R}$ and D_- for the divisor obtained by applying a baby step backwards from D, i.e. $(D_-)_+ = (D_+)_- = D$.

Heuristics (H): For sufficiently large q, the following properties hold with probability $1 - O(q^{-1})$:

(H1) $\delta(D_+) - \delta(D) = 1$ for all $D \in \mathcal{R} \setminus \{0\}$.

(H2) The quantity d in (2.4) is always equal to $\lceil g/2 \rceil$. That is, for all $D, D' \in \mathcal{R} \setminus \{0\}$, we have $\delta(D \oplus D') = \delta(D) + \delta(D') - \lceil g/2 \rceil$.

There is overwhelming numerical evidence as well as plausible theoretical considerations that support the above heuristics, especially for large q. In fact, for elliptic curves, i.e. the case $g = 1$, H1 is a provable fact. To justify Heuristic H1, observe that by (2.3), $\delta(D_+) - \delta(D)$ is equal to the degree of a partial quotient in the continued fraction expansion of $y \in \mathbb{F}_q\langle x^{-1} \rangle$, and such partial quotients are conjectured to have degree 1 with probability $1 - O(q^{-1})$. We point out that again by (2.3), Heuristic H1 is equivalent to the assumption that for $D = \operatorname{div}(a, b) \in \mathcal{R} \setminus \{0\}$, $\deg(a) = g$ with probability $1 - O(q^{-1})$.

To justify Heuristic H2, consider the following. If we write $D = \operatorname{div}(a, b)$ and $D' = \operatorname{div}(a', b')$, then at most $t = \lceil (\deg(a) + \deg(a') - g)/2 \rceil$ baby steps are required to obtain a reduced divisor when starting at $D \oplus D'$. If we assume $\deg(a) = \deg(a') = g$ according to Heuristic H1, then $t = \lceil g/2 \rceil$. Since each baby step yields an advance in distance of exactly 1, we should have $d = \lceil g/2 \rceil$ in (2.4).

Under the assumption of these heuristics, we can pin down distances exactly; in particular, we derive from (2.3) that $\delta_i = g + i - 1$ as long as no degenerate divisor, i.e. a divisor $D = \operatorname{div}(a, b) \in \mathcal{R} \setminus \{0\}$ with $\deg(a) < g$, is encountered. Furthermore, under this assumption, we no longer need to compute any relative distances during our computations. In particular, H2 states that a giant step produces a divisor in $\mathcal{R}$ of distance exactly $\lceil g/2 \rceil$ short of the target distance, i.e. the sum of the distances of the input divisors, and H1 states that precisely $\lceil g/2 \rceil$ baby steps are required to effect the adjustment, again assuming that no degenerate divisor is encountered.

Our first algorithm, VAR-DIST2, represents the improved variable distance situation. Put $d = \lceil g/2 \rceil$. The idea is to compute the divisor $D' \in \mathcal{R}$ of distance d from D and apply scalar multiplication using NAF to D' instead of to D directly. The result is that after any giant step, we will end up with a reduced divisor of distance precisely d more than the target without having to do any adjustments with baby steps. For example, assuming Heuristic H2, we know that the distance to the reduced divisor of a giant step is precisely d, so we have $\delta(D' \oplus D') = 2(\delta(D)+d) - d = 2\delta(D)+d$; similarly, $\delta((D' \oplus D') \oplus D') = (2\delta(D') + d) + (\delta(D') + d) - d = 3\delta(D') + d$, etc. In Proposition 3.1, we formally prove, assuming Heuristics H1 and H2, that the reduced divisor output at the end of Algorithm VAR-DIST2 has distance precisely $n\delta(D) + d$.

VAR-DIST2(D, n)
Input: $D \in \mathcal{R}$, $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
Output: The divisor in $\mathcal{R}$ of distance $n\delta(D) + d$.
Algorithm:

1. for $i = 1$ to $d - 1$ do
   1.1. replace D by D_+;
2. set $D' = D,\ D'' = D_+,\ E = D_+$;
3. for $i = 1$ to l do
   3.1. replace E by $E \oplus E$;
   3.2. if $b_i = 1$ then replace E by $E \oplus D''$;
   3.3. if $b_i = -1$ and g is even then replace E by $E \oplus \overline{D''}$;
   3.4. if $b_i = -1$ and g is odd then replace E by $E \oplus \overline{D'}$;
4. output E.

Proposition 3.1. *Assuming Heuristics H1 and H2, VAR-DIST2(D, n) outputs the divisor $E \in \mathcal{R}$ of distance $\delta(E) = n\delta(D) + d$.*

Proof. By Heuristic H1, we have $\delta(D') = \delta(D)+d-1$ and $\delta(D'') = \delta(E) = \delta(D)+d$ in step 2. Set $E_0 = E$, let E_i be the divisor obtained after the i-th iteration of the for loop in step 3, and set $d_i = \delta(E_i)$. Then $d_0 = \delta(D) + d$, and for $1 \leq i \leq l$ and $b_i \in \{0, 1\}$, d_i satisfies the recursion

$$d_i \equiv (2d_{i-1} - d) + b_i(\delta(D'') - d) \equiv 2d_{i-1} - d + b_i\delta(D) \pmod{R_x} \tag{3.1}$$

by (2.4) and Heuristic H2.

If $b_i = -1$ and g is even, then $g = 2d$, so the recursion for d_i is

$$\begin{aligned} d_i &\equiv (2d_{i-1} - d) + \delta(\overline{D''}) - d \\ &\equiv (2d_{i-1} - d) + g - \delta(D'') - d \\ &\equiv (2d_{i-1} - d) + g - (\delta(D) + d) - d \\ &\equiv (2d_{i-1} - d) + g - \delta(D) - 2d \\ &\equiv (2d_{i-1} - d) - \delta(D) \\ &\equiv 2d_{i-1} - d + b_i\delta(D) \pmod{R_x} \ . \end{aligned}$$

Finally, if $b_i = -1$ and g is odd, then $g = 2d - 1$, so the recursion becomes

$$\begin{aligned} d_i &\equiv (2d_{i-1} - d) + \delta(\overline{D'}) - d \\ &\equiv (2d_{i-1} - d) + g - \delta(D') - d \\ &\equiv (2d_{i-1} - d) + g - (\delta(D) + d - 1) - d \\ &\equiv (2d_{i-1} - d) + g - \delta(D) - (2d - 1) \\ &\equiv (2d_{i-1} - d) - \delta(D) \\ &\equiv 2d_{i-1} - d + b_i\delta(D) \pmod{R_x} \ . \end{aligned}$$

So in all cases, d_i satisfies the recursion (3.1). It follows that

$$\begin{aligned} d_l &\equiv 2^l d_0 - (2^l - 1)d + (n - 2^l)\delta(D) \\ &\equiv 2^l(\delta(D) + d) - (2^l - 1)d + (n - 2^l)\delta(D) \\ &\equiv n\delta(D) + d \pmod{R_x} \ , \end{aligned}$$

which proves our claim. □

For the fixed distance scenario, the goal is to compute the divisor in $\mathcal{R}$ whose distance is some known and easily computable function of the input scalar n using an analog of scalar multiplication with the NAF of n in which we perform the doublings as usual, but replace each add by a baby step. Roughly speaking, the resulting algorithm would trade $l/3$ adds on average for $l/3$ baby steps, which are significantly faster. The trick is to apply this method in such a way that the output is indeed the divisor in $\mathcal{R}$ of our desired distance for any positive integer n.

We again assume that we know $d = \lceil g/2 \rceil$ and that the fixed base is the divisor D_{d+3} obtained by applying $d + 2$ baby steps to $D_1 = 0$. This divisor has distance $\delta_{d+3} = g + d + 2$ by Heuristic H1 and will be included in the domain parameters of all our cryptographic protocols. The following algorithm computes the divisor in $\mathcal{R}$ of distance $2^l(g+1)+n+d$, where $l+1$ is the NAF length of our scalar, by applying the double and baby step strategy outlined above to D_{d+3}. Under Heuristics H1

and H2, we prove in Proposition 3.2 that the output is indeed the divisor in $\mathcal{R}$ of distance $2^l(g+1)+n+d$.

FIXED-DIST2(n)
Input: $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
Output: The divisor in $\mathcal{R}$ of distance $2^l(g+1)+n+d$.
Algorithm:
1. set $E = D_{d+3}$;
2. for $i = 1$ to l do
   2.1. replace E by $E \oplus E$;
   2.2. if $b_i = 1$ then replace E by E_+;
   2.3. if $b_i = -1$ then replace E by E_-;
3. output E.

Proposition 3.2. *Assuming Heuristics H1 and H2, FIXED-DIST2(n) outputs the divisor $E \in \mathcal{R}$ of distance $\delta(E) = 2^l(g+1)+n+d$.*

Proof. Set $E_0 = D_{d+3}$, let E_i be the divisor obtained after the i-th iteration of the for loop in step 2, and set $d_i = \delta(E_i)$. Then $d_0 = d+g+2$, $d_i = 2d_{i-1} - d + b_i$ for $1 \le i \le l$ by (2.4), and therefore

$$\begin{aligned} d_l &= 2^l d_0 - (2^l - 1)d + (n - 2^l) \\ &= 2^l(g+d+2) - (2^l-1)d + (n-2^l) \\ &= 2^l(g+1) + n + d \ , \end{aligned}$$

which proves our claim. □

As we will see in the next section, FIXED-DIST2 is sufficient for cryptographic purposes; we only require a divisor of some randomly produced but known distance as opposed to a divisor precisely of distance n. Nevertheless, as a point of interest, we show how to use the above algorithm on input n to compute the actual divisor in $\mathcal{R}$ of distance n. Since this requires another more costly precomputation, we will not use this method in any of our cryptographic protocols, except for computing domain parameters such as public keys as, for example, in the digital signature protocol described in the next section. However, the technique might be useful for number theoretic applications such as finding the regulator R_x or the divisor class number of the curve C. For this procedure, we precompute another divisor D^* of distance $2^l(g+1)+g$. Assuming Heuristics H1 and H2, D^* can be found as follows. Initialize $E_0 = D_{d+2}$ (the divisor of distance $g+1+d$), and for $1 \le i \le l$, let $E_i = E_{i-1} \oplus E_{i-1}$. Then $\delta(E_i) = 2^i(g+1)+d$ by (2.4) and Heuristic H2, so applying $g-d$ baby steps to E_l yields D^* by Heuristic H1. The total cost of this precomputation is therefore l doubles plus $(d+1)+(g-d) = g+1$ baby steps; the $d+1$ baby steps can be replaced by one backward baby step if we compute D_{d+2} from D_{d+3}. To find a divisor in $\mathcal{R}$ of distance n, we simply call the previous algorithm and add the conjugate $\overline{D^*}$ of D^* to the result.

FIXED-DIST3(n)
Precomputation: $D^* \in \mathcal{R}$ with $\delta(D^*) = 2^l(g+1)+g$.
Input: $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
Output: The divisor in $\mathcal{R}$ of distance n.
Algorithm:
1. compute E' = FIXED-DIST2(n);

2. compute $E = E' \oplus \overline{D^*}$;
3. output E.

Proposition 3.3. *Assuming Heuristics H1 and H2, FIXED-DIST3(n) outputs the divisor $E \in \mathcal{R}$ of distance $\delta(E) = n$.*

Proof. By Proposition 3.2, $\delta(E') = 2^l(g+1) + n + d$. Now $\delta(\overline{D^*}) \equiv g - \delta(D^*) \equiv -2^l(g+1) \pmod{R_x}$, so by (2.4) and Heuristic H2, the output divisor E of Algorithm FIXED-DIST3 satisfies

$$\begin{aligned}\delta(E) &\equiv \delta(E') + \delta(\overline{D^*}) - d \\ &\equiv (2^l(g+1) + n + d) - 2^l(g+1) - d \\ &\equiv n \pmod{R_x}\end{aligned}$$

as claimed. □

3.4. COMPARISON OF REAL AND IMAGINARY MODELS. In Table 1, we compare the expected computational effort of all the algorithms presented in this section. The imaginary setting does not distinguish between the fixed and variable divisor scenarios at the level of counting giant steps. We assume here that the NAF representations of the scalars have length $l+1$, with a proportion of one third of the digits non-zero in the NAF setting. For the real scenario, we assume Heuristics H1 and H2, and, as before, that Δ is an upper bound on the giant step distance "errors" caused by reduction. By (2.4), $\Delta \leq 2g$, but as argued above, on average we would expect $\Delta \approx g/2$ because of Heuristics H1 and H2.

TABLE 1. Operation counts for scalar multiplication in $\mathcal{R}$

	Doubles	Adds	Baby Steps
Imaginary	l	$l/3$	-
Real, VAR-DIST1	l	$l/3$	$4\Delta l/3$
Real, FIXED-DIST1	$l - \log_2(g+1)$	$(l - \log_2(g+1))/3$	g
Real, VAR-DIST2	l	$l/3$	d
Real, FIXED-DIST2	l	0	$l/3$
Real, FIXED-DIST3	l	1	$l/3$

Note that our improved algorithm for the variable distance scenario, VAR-DIST2, requires d more baby steps than imaginary scalar multiplication, whereas the improved fixed distance algorithm, FIXED-DIST2, replaces $l/3$ adds in the imaginary case by $l/3$ baby steps in the real situation. Also, our new algorithm VAR-DIST2 requires significantly fewer baby steps than VAR-DIST1, and FIXED-DIST2 and FIXED-DIST3 trade roughly $l/3$ adds for as many baby steps.

4. CRYPTOGRAPHIC PROTOCOLS USING REAL HYPERELLIPTIC CURVES

We now present real hyperelliptic curve variants of the most common discrete logarithm based protocols and compare them with their imaginary counterparts as described for elliptic curves in Sections 4.4-4.6, pp. 183-196, of [11], see also pp. 570f. of [4] for a version of the digital signature algorithm on imaginary hyperelliptic curves. The domain parameters for both situations include a prime power q, a hyperelliptic curve C over $\mathbb{F}_q$ of genus g, and an integer R that is equal to the

regulator R_x in the real case and to the order of the subgroup of the Jacobian of C in which we conduct our arithmetic in the imaginary case. The last component of the domain parameters is a divisor D. In the imaginary scenario, we only consider the most general situation where D is a randomly generated reduced divisor of order R; note that if D were chosen to be a degenerate divisor (see for instance [15]), this would speed up the fixed divisor scenario in the imaginary case. In the real scenario, we set $D = D_{d+3}$ where D_{d+3} is the divisor in $\mathcal{R}$ described in Section 3.3. Thus, the domain parameters are very similar in both cases.

In essence, we can convert any protocol using the imaginary model to the real model by replacing scalar multiplications of divisors by calls to either FIXED-DIST2(n) or VAR-DIST2(D, n). Any scalar multiplication in the imaginary scenario using a fixed divisor included in the domain parameters corresponds to a call to FIXED-DIST2. A scalar multiplication of a randomly produced divisor in the imaginary case corresponds to a call to VAR-DIST2 on the appropriate input.

To illustrate this idea, we first present a real hyperelliptic curve version of basic Diffie-Hellman key exchange.

Diffie-Hellman Key Exchange for Real Hyperelliptic Curves
Domain parameters: q, C, $R = R_x$, $D = D_{d+3}$.
Round 1:

1. Alice generates random $n_a \in [1, R-1]$, computes $D_A = \text{FIXED-DIST2}(n_a)$ and sends D_A to Bob;
2. Bob generates random $n_b \in [1, R-1]$, computes $D_B = \text{FIXED-DIST2}(n_b)$, and sends D_B to Alice;

Round 2:

1. Alice computes $K = \text{VAR-DIST2}(D_B, 2^l(g+1) + n_a + d \pmod R)$;
2. Bob computes $K = \text{VAR-DIST2}(D_A, 2^l(g+1) + n_b + d \pmod R)$.

By Propositions 3.1 and 3.2, at the end of this protocol Alice and Bob share the common key K which is the divisor in $\mathcal{R}$ of distance

$$\delta(K) \equiv (2^l(g+1) + n_a + d)(2^l(g+1) + n_b + d) + d \pmod R \ .$$

Note that there is a very small probability that Alice and Bob do not end up with the same key, as the proofs of the propositions assume the Heuristics H1 and H2, but the probability of this happening is negligible for sufficiently large q, particularly for the sizes of q required to ensure the intractability of breaking the protocol (see Section 5).

We point out that the above protocol can still be executed if R is not known; this was the case in our implementation, see Section 6. In this case, we simply omit the reduction modulo R of the scalar inputs to VAR-DIST2 in Round 2. Omitting this reduction does result in a slight degradation in efficiency for Round 2, as the scalar input to VAR-DIST2 will have roughly $\log_2(g+2)$ additional bits — this is discussed in more detail at the end of this section.

We next present a generalization of the digital signature algorithm (see Algorithms 4.29 and 4.30, p. 184, of [11]) to the real model. We assume that each user has a private key that is an integer $n \in [1, R-1]$ and a public key E where E is the reduced divisor in the class of nD in the imaginary setting and the divisor in $\mathcal{R}$ of distance n in the real setting; in the latter case, E can be computed by invoking FIXED-DIST3(n). Also, all participants have agreed on a public hash function H that maps messages to integers in $[1, R-1]$. Recall that every divisor

in $\mathcal{R}$ is uniquely represented by a pair of polynomials $a, b \in \mathbb{F}_q[x]$ with $\deg(a) \leq g$. Therefore, it is easy to convert a divisor $\mathrm{div}(a, b)$ to an integer by applying a key derivation function or an extractor as described in, for example, [11, p.189].

Digital Signature Algorithm for Real Hyperelliptic Curves
Domain parameters: q, C, $R = R_x$, $D = D_{d+3}$.
Public key: a divisor $E \in \mathcal{R}$;
Private key: $n = \delta(E)$;
Signature Generation: To sign a message m, the signer:

1. generates random $k \in [1, R-1]$ with $\gcd(k, R) = 1$;
2. computes $D_k = \text{FIXED-DIST2}(k)$ and converts D_k to an integer N;
3. sets $r \equiv N \pmod{R}$; if $r = 0$, returns to step 1;
4. computes $s \equiv k^{-1}(H(m) + nr) \pmod{R}$; if $s = 0$, returns to step 1;
5. signs m with signature (r, s).

Signature Verification: Upon receiving (m, r, s), the verifier:

1. verifies that $1 \leq r, s < R$; if not, rejects the signature;
2. computes $w \equiv s^{-1}$, $u_1 \equiv H(m)w$, $u_2 \equiv rw \pmod{R}$;
3. computes $E_1 = \text{FIXED-DIST2}(u_1)$ and $E_2 = \text{VAR-DIST2}(E, u_2)$;
4. computes $E_3 = E_1 \oplus E_2$; if $E_3 = 0$, rejects the signature, else converts E_3 to an integer N;
5. accepts the signature if $r \equiv N \pmod{R}$ and rejects it otherwise.

Note that if the signature is valid, then E_3 has distance

$$\begin{aligned}\delta(E_3) &\equiv \delta(E_1) + \delta(E_2) - d \\ &\equiv \left(2^l(g+1) + u_1 + d\right) + (u_2 n + d) - d \\ &\equiv 2^l(g+1) + s^{-1}(H(m) + rn) + d \\ &\equiv 2^l(g+1) + k + d \\ &\equiv \delta(D_k) \pmod{R}\ ,\end{aligned}$$

so $E_3 = D_k$, and hence $r \equiv N \pmod{R}$.

Given these examples, it is straightforward to adapt other protocols such as the station-to-station and MQV key agreement protocols, integrated encryption system (IES), and the provably secure elliptic curve (PSEC) encryption scheme to the real case. In the interest of space, we will not describe these protocols explicitly here, but refer instead to pp. 184-191 of [11] where the elliptic curve versions, which are analogous to the imaginary hyperelliptic curve versions, are described.

In Table 2 we compare the real and imaginary variants of five cryptographic protocols: the Station to Station Diffie-Hellman and MQV key agreement procedures, the DSA, and the IES and PSEC cryptosystems. We ignore any task that is identical for both scenarios, such as hashing, key derivation, x-or, MACs, and the like, and compare only the effort of scalar multiplication. Furthermore, we only consider scalars in NAF, since this is the most efficient scenario that does not require any precomputation. We assume that all scalars have the same NAF length $l + 1$ as R.

Column 2 of the table indicates whether the fixed (F) or the variable (V) divisor/distance scenario is used. Column 3 gives the number of doubles for each task specified in Column 1, which is the same for the real and the imaginary scenario. Columns 4-6 given the number of adds for the imaginary case, the number of adds for the real setting, and the number of baby steps used in the real setting, respectively.

Table 2. Expected number of operations for various cryptographic protocols

Protocol		F and V	Dbls	Adds Imag	Adds Real	Baby Steps Real
STS		F + V	$2l$	$2l/3$	$l/3$	$l/3+d$
MQV		F + 2V	$3l$	$l+1$	$2l/3+1$	$l/3+2d$
DSA	Sig. Gen.	F	l	$l/3$	0	$l/3$
	Sig. Ver.	F + V	$2l$	$2l/3+1$	$l/3+1$	$l/3+d$
IES	Encr.	F + V	$2l$	$2l/3$	$l/3$	$l/3+d$
	Decr.	V	l	$l/3$	$l/3$	d
PSEC	Encr.	F + V	$2l$	$2l/3$	$l/3$	$l/3+d$
	Decr.	F + V	$2l$	$2l/3$	$l/3$	$l/3+d$

We see that in almost all cases, the real scenario uses fewer adds than its imaginary counterpart, at the expense of additional baby steps. The notable exception is IES decryption, where both scenarios use an equal number of adds. As an example, we see that for plain Diffie-Hellman, the number of adds plus doubles required is reduced from $8l/3$ in the imaginary case to $7l/3$ in the real case, i.e. $1/8$ of the required giant steps are replaced by baby steps. Assuming that giant steps in the imaginary and real cases have roughly the same cost and that baby steps have negligible cost, and omitting the distinction between doubles and adds in both scenarios, our version in the real setting would run in $7/8$ the time of that in the imaginary setting, yielding a speed-up of approximately 12%. The most dramatic difference between the two settings occurs for DSA signature generation, where the imaginary setting requires $l/3$ adds, whereas the real case only needs $l/3$ baby steps. In that case, we would expect a speed up of approximately 25%. We again point out that these figures only apply to the most general case where the divisor D included in the domain parameters for an imaginary curve is non-degenerate. If we assume a degenerate divisor and allow procomputation involving the base divisor, we expect the difference in performance between the real and imaginary settings to be less dramatic. A more exact comparison is warranted and will be investigated in the future.

Just like Diffie-Hellman, the STS protocol can be performed without knowledge of the regulator R_x. In this case, the runtime in the real scenario is slightly worse. The analysis of the call to FIXED-DIST2 in Round 1 remains the same as above, assuming an NAF length of $l+1$ for Alice's and Bob's respective scalars. However, the call to VAR-DIST2 in Round 2 is performed on a scalar input of NAF length $l'+1$ where $2^{l'} \approx 2^l(g+1)+2^l+d$ by Proposition 3.2. Neglecting the summand d, we see that $l' = l+c$ where $c \approx \log_2(g+2)$. Thus, under the assumption that R is unknown — which was the case for our implementation described in Section 6 — STS performs slightly worse in the real case, requiring $2l+c$ doubles, $(l+c)/3$ adds, and as before $l/3+d$ baby steps.

5. Security

The security of Diffie-Hellman key exchange using the imaginary model of a hyperelliptic curve has been studied extensively; see [12] for a survey. Being able to solve the discrete logarithm problem (DLP) in the degree zero divisor class group,

i.e. computing the integer n given divisors nD and D, allows one to break the protocol. For sufficiently small genus, the best known algorithm for solving the DLP requires $O(q^{g/2})$ operations. For $g \geq 3$, asymptotically faster algorithms exist [9].

Our version of Diffie-Hellman key exchange using the real model is just a more computationally efficient version of those in [23, 22], so the same well-known security considerations apply (see [23]). In order to break this protocol, an eavesdropper has the following problem: given D_A (or D_B), find either the unknown distance $\delta(D_A)$ (or $\delta(D_B)$), or equivalently, find one of the scalars n_a or n_b. As discussed in [23], this means that the eavesdropper has to solve the infrastructure DLP. This problem is closely related to the DLP in the imaginary setting (see, for example, [12]), and as most of the known algorithms for solving the DLP in the imaginary scenario can be modified to solve the infrastructure DLP, the same security considerations as in the imaginary setting apply to the real setting.

As in the imaginary case, one should compute the size of the key space to be used for the real protocols, i.e., the regulator. This is required for some protocols such as the DSA analogue, because scalar multipliers must be reduced modulo the regulator. Although it is not known how to take advantage of a smooth regulator to solve the infrastructure DLP via some analogue of the Pohlig-Hellman algorithm, such an algorithm can be obtained by transforming the problem to an instance of the DLP in a subgroup of order R_x of the Jacobian of the curve. Thus, as with the order of the base element in the imaginary case, the regulator should be prime or at least have a large prime divisor.

Computing the regulator for curves and finite fields of cryptographic size is an open problem. One possibility is to first compute an imaginary curve and transform it to a related real curve. The regulator of the resulting real curve will be a divisor of the order of the Jacobian of the curve, so by using, for example, imaginary curves of special forms that admit easy computation of the order of the Jacobian, one can obtain real curves whose regulators are known.

As pointed out earlier, heuristics and numerical experiments indicate that degenerate divisors occur very rarely for q sufficiently large. In [2], it was shown how a differential power analysis attack on elliptic curves originally due to Goubin [10] can make hyperelliptic curve protocols potentially susceptible to such an attack, due to the occasional encounter of a degenerate divisor during execution of the protocol. It is currently not known how to take advantage of this situation in the protocols in the real model.

The imaginary versions of the other protocols, DSA, Station-to-Station, MQV, IES, and PSEC, are all provably secure against adaptive chosen ciphertext or chosen message attacks in the random oracle model assuming the intractability of the discrete logarithm problem [11]. Our real versions of these protocols are straightforward generalizations in which the usual discrete logarithm problem is replaced by the infrastructure discrete logarithm problem. Although we do not have security proofs for our protocols, it seems reasonable that the same or similar security proofs should hold, since the infrastructure DLP and the DLP in the Jacobian of a hyperelliptic curve are closely related (see [20]).

6. Implementation and Numerical Results

In order to test the efficiency of our protocols, we implemented NAF-based scalar multiplication in the Jacobian of an imaginary hyperelliptic curve (Algorithm SCALAR-MULT) and the two new scalar multiplication algorithms for the

infrastructure of a real hyperelliptic curve (Algorithm FIXED-DIST2 and Algorithm VAR-DIST2). For comparison purposes, we also implemented Algorithms FIXED-DIST1 and VAR-DIST1, which are based on previous work [23]. We used the computer algebra library NTL [25] for finite field and polynomial arithmetic and the GNU `C++` compiler version 3.4.3. The computations described below were performed on a Pentium IV 2.8 GHz computer running Linux. All five algorithms were implemented using curves defined over prime finite fields $\mathbb{F}_p$ and characteristic 2 finite fields $\mathbb{F}_{2^n}$.

We used the formulation of Cantor's algorithm described in [26] for the giant step operation in both the imaginary and real case for curves over $\mathbb{F}_p$, and the obvious generalizations for curves over $\mathbb{F}_{2^n}$. For reference, the precise formulas for curves over $\mathbb{F}_p$ we used are presented in Appendix A; the formulas for even characteristic are almost identical (see, for example, [16], [12], or [13]). We chose Cantor's algorithm for our experiments because it is the fastest known algorithm that can be used in both the imaginary and real settings. There are much faster algorithms using explicit formulas for the imaginary model [17, 29, 21], but to date these have only begun to be generalized to the real setting [6]. Developing new algorithms for divisor arithmetic is beyond the scope of this paper. Hence, in order to provide a fair comparison illustrating the relative advantages of using the real model as opposed to the imaginary model, Cantor's algorithm was the obvious choice.

We expect that the explicit formulas for the real model will be comparable in speed to their imaginary counterparts, and that the speed ratio between the real and imaginary scenarios will be roughly the same when using explicit formulas instead of Cantor's algorithm. However, it must be emphasized that the goal of our experiments was to demonstrate that the real model is competitive in efficiency and to stimulate work on the explicit formulas that are required to conduct a conclusive and rigorous comparison.

We ran numerous examples of the five scalar multiplication algorithms using curves with genus ranging from 2 to 6 and with the underlying finite field chosen so that the size of the set $\mathcal{R}$, which is approximately q^g where the finite field has q elements, was sufficiently large that the best-known attack on the DLP [9] requires roughly 2^{80}, 2^{112}, 2^{128}, 2^{192}, or 2^{256} operations, corresponding to the five security levels for key establishment in U.S. Government applications recommended by NIST [19]. Table 3 lists the bit-lengths for q required to achieve these security levels. For

Table 3. Required Finite Field Sizes (bits)

Security level	2	3	4	5	6
80	80	60	54	50	48
112	112	84	75	70	68
128	128	96	86	80	77
192	192	144	128	120	116
256	256	192	171	160	154

$g = 2$, the best-known attack uses Pollard-rho to solve the DLP, and has asymptotic complexity $\sqrt{q^2} = q$. For $g \geq 3$, the algorithm in [9] is asymptotically faster, so the estimates in Table 3 were derived from the runtime complexities listed in Table 1 of [9]. As these complexities only hold asymptotically, it is unclear how accurate our estimates are for practical applications — further work is required to extrapolate

expected runtimes for the algorithm in [9] to parameters of cryptographic sizes in order to derive more accurate field sizes. Although the use of curves with genus 3 and larger for cryptographic purposes is questionable, we nevertheless included times for higher genus as they may still be of use for certain cryptographic applications (see for example [21]) as well as other number theoretic problems such as regulator and clas number computation.

For curves defined over $\mathbb{F}_p$, we chose a random prime p of appropriate bit length from Table 3, and for curves over $\mathbb{F}_{2^n}$, the entries in Table 3 are precisely the required values of n. For each genus and finite field, we randomly selected 200 imaginary curves and 200 real curves, and executed Diffie-Hellman key exchange 10 times for each curve. For the real curves, we performed the key exchange protocol using both the original version (Algorithms FIXED-DIST1 and VAR-DIST1) and our improved version (Algorithms FIXED-DIST2 and VAR-DIST2). Thus, we ran 8000 instances of Algorithm SCALAR-MULT (two instances for each participant during each run of the protocol) and 4000 instances each of Algorithm FIXED-DIST1, VAR-DIST1, FIXED-DIST2, and VAR-DIST2 (one instance of each algorithm per participant during each run of the protocol). The random exponents used had either $160, 224, 256, 384$, or 512 bits, ensuring that the number of bits of security provided corresponds to the five levels recommended by NIST. We did not assume that the regulator is known, so the scalar inputs to VAR-DIST2 have $\log_2(g+2)$ additional bits. In order to provide a fair comparison, the same sequence of random exponents was used for each run of the key exchange protocol. We note that although there is a negligible probability that the protocol in the real model fails due to the assumptions of Heuristics H1 and H2, in all cases both parties computed the same key.

Tables 4 and 5 contain the average CPU time in seconds for each of the five scalar multiplication algorithms using curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$, respectively. The times required to generate domain parameters are not included in these timings, as domain parameter generation is a one-time computation. The times obtained bear out our predictions. The new versions of the real scalar multiplication algorithms are faster than their predecessors. Our improved algorithm for fixed-base scalar multiplication in the real case (FIXED-DIST2) is faster than SCALAR-MULT, and VAR-DIST2 is slightly slower.

In Table 6, we give the ratios of the average time spent on divisor arithmetic for Diffie-Hellman key exchange using our improved version in the real model compared to its predecessor, denoted by "New/Old", and the ratios of our improved version as compared to that using the imaginary model, denoted by "New/Imag". As mentioned earlier, we do not assume that the regulators are known for these examples, so the exponents used for VAR-DIST2 are $\log_2(g+2)$ bits longer than those used for the other algorithms.

Our results show that, using generic divisor arithmetic and no precomputation, our improved version of key exchange in the real model requires roughly 60% of the time required without our modifications. In addition, it is competitive with key exchange using the imaginary model in this scenario, and in fact faster in most cases. In addition, the improvements appear to be approaching the predicted (Table 2) values as both the genus and security level increase. Also as predicted, computing signatures (which is essentially algorithm FIXED-DIST2) is especially efficient when using the real model.

Table 4. Scalar multiplication timings over $\mathbb{F}_p$.

Security level (in bits)	g	Running Times (in seconds)				
		S-MULT	F-DIST1	V-DIST1	F-DIST2	V-DIST2
80	2	0.0129	0.0227	0.0227	0.0116	0.0144
	3	0.0182	0.0305	0.0309	0.0161	0.0208
	4	0.0282	0.0476	0.0485	0.0248	0.0320
	5	0.0453	0.0733	0.0747	0.0392	0.0512
	6	0.0629	0.1042	0.1057	0.0538	0.0690
112	2	0.0223	0.0378	0.0384	0.0198	0.0240
	3	0.0331	0.0540	0.0548	0.0292	0.0372
	4	0.0512	0.0845	0.0859	0.0443	0.0569
	5	0.0813	0.1309	0.1334	0.0710	0.0910
	6	0.1239	0.2014	0.2038	0.1052	0.1348
128	2	0.0264	0.0443	0.0449	0.0232	0.0285
	3	0.0397	0.0646	0.0652	0.0348	0.0441
	4	0.0610	0.1000	0.1014	0.0525	0.0671
	5	0.0934	0.1499	0.1513	0.0799	0.1037
	6	0.1344	0.2163	0.2194	0.1140	0.1460
192	2	0.0556	0.0895	0.0901	0.0476	0.0581
	3	0.0818	0.1283	0.1287	0.0713	0.0894
	4	0.1131	0.1825	0.1843	0.0970	0.1222
	5	0.1788	0.2776	0.2786	0.1510	0.1924
	6	0.2561	0.4123	0.4122	0.2139	0.2723
256	2	0.0981	0.1531	0.1546	0.0839	0.1015
	3	0.1364	0.2138	0.2143	0.1152	0.1447
	4	0.2384	0.3574	0.3588	0.1983	0.2514
	5	0.2916	0.4452	0.4459	0.2436	0.3090
	6	0.4209	0.6598	0.6609	0.3488	0.4437

7. Conclusions and Future Work

Our results show that using the real model of a hyperelliptic curve as opposed to the usual imaginary model holds much more promise for practical applications than previously believed. In fact, when using Cantor's algorithm as described in [26] for the giant-step divisor arithmetic, our improvements to the real model yield significant speed-ups over previous work and obtain comparable performance to the imaginary model for curves of genus as small as 2 over finite fields of cryptographically relevant size; this holds for both prime fields and fields of characteristic 2.

As discussed in Section 3.4, the performance improvements clearly depend on how much faster one can perform a baby step as opposed to a giant step in practice and how optimized formulas for giant steps in the real case compare to those in the imaginary case. Thus, it is necessary to investigate more closely the performance of our protocols using more efficient explicit formulas for giant steps on low-genus curves. However, such formulas have only begun to be generalized to the real case recently. To date, only formulas using affine coordinates for genus two curves over

TABLE 5. Scalar multiplication timings over $\mathbb{F}_{2^n}$.

Security level (in bits)	g	Running Times (in seconds)				
		S-MULT	F-DIST1	V-DIST1	F-DIST2	V-DIST2
80	2	0.0137	0.0234	0.0231	0.0114	0.0143
	3	0.0166	0.0277	0.0278	0.0140	0.0183
	4	0.0266	0.0442	0.0444	0.0222	0.0291
	5	0.0437	0.0701	0.0708	0.0364	0.0475
	6	0.0579	0.0944	0.0943	0.0470	0.0613
112	2	0.0256	0.0406	0.0403	0.0202	0.0251
	3	0.0343	0.0542	0.0540	0.0272	0.0360
	4	0.0553	0.0848	0.0848	0.0444	0.0578
	5	0.0844	0.1260	0.1260	0.0669	0.0887
	6	0.1186	0.1790	0.1791	0.0935	0.1236
128	2	0.0303	0.0475	0.0472	0.0237	0.0294
	3	0.0409	0.0643	0.0641	0.0323	0.0428
	4	0.0644	0.0989	0.0985	0.0513	0.0668
	5	0.0994	0.1480	0.1491	0.0794	0.1041
	6	0.1381	0.2086	0.2095	0.1087	0.1420
192	2	0.0704	0.1020	0.1016	0.0535	0.0678
	3	0.1055	0.1505	0.1507	0.0833	0.1070
	4	0.1316	0.1939	0.1934	0.1034	0.1329
	5	0.2048	0.2966	0.2971	0.1614	0.2098
	6	0.2908	0.4247	0.4243	0.2254	0.2937
256	2	0.1325	0.1857	0.1848	0.0992	0.1215
	3	0.1725	0.2359	0.2356	0.1329	0.1687
	4	0.2678	0.3741	0.3736	0.2059	0.2650
	5	0.3686	0.5109	0.5104	0.2865	0.3728
	6	0.5064	0.7104	0.7090	0.3882	0.5066

prime fields have been developed [6]. More work is required in this area before the best arithmetic in imaginary and real curves can be compared.

Our improvements in efficiency come from the use of FIXED-DIST2(n), where we replaced the adds in the usual double and add scalar multiplication algorithm by baby steps. Thus, we expect that protocols in which we can replace scalar multiplication of a fixed divisor with a call to FIXED-DIST2(n) will benefit the most, in terms of efficiency, from this replacement. In particular, signature computation for our version of the digital signature algorithm requires only one call to FIXED-DIST2(n), so we expect especially significant improvements in this case, potentially yielding a speed-up of as much as 25%. Our numerical results clearly support this conclusion.

It remains to be seen how the fixed distance scenario in the real case compares to using a degenerate divisor, rather than a random fixed divisor, in the imaginary setting, as suggested in [15]. For such a divisor D, the degree of D_x is less than the genus, and can be as low as one, i.e. $D = P - \infty$ with P an $\mathbb{F}_q$-rational point on the curve. We point out that the idea of using a degenerate instead of a random divisor in the domain parameters for imaginary hyperelliptic curve protocols can be used

Table 6. Ratios of key exchange runtimes.

Security level (in bits)	g	$\mathbb{F}_p$		$\mathbb{F}_{2^n}$	
		New/Old	New/Imag	New/Old	New/Imag
80	2	0.5734	1.0062	0.5536	0.9390
	3	0.6012	1.0156	0.5826	0.9727
	4	0.5904	1.0053	0.5790	0.9638
	5	0.6112	0.9985	0.5953	0.9599
	6	0.5848	0.9752	0.5742	0.9352
112	2	0.5749	0.9821	0.5598	0.8850
	3	0.6099	1.0023	0.5839	0.9207
	4	0.5939	0.9885	0.6027	0.9243
	5	0.6131	0.9958	0.6173	0.9212
	6	0.5925	0.9688	0.6062	0.9148
128	2	0.5802	0.9810	0.5602	0.8763
	3	0.6075	0.9925	0.5851	0.9178
	4	0.5938	0.9809	0.5986	0.9169
	5	0.6095	0.9824	0.6176	0.9226
	6	0.5968	0.9671	0.5995	0.9074
192	2	0.5887	0.9502	0.5958	0.8614
	3	0.6255	0.9822	0.6319	0.9023
	4	0.5977	0.9689	0.6102	0.8981
	5	0.6174	0.9604	0.6251	0.9062
	6	0.5898	0.9493	0.6115	0.8926
256	2	0.6028	0.9454	0.5956	0.8327
	3	0.6070	0.9526	0.6396	0.8740
	4	0.6279	0.9432	0.6299	0.8794
	5	0.6201	0.9474	0.6456	0.8945
	6	0.6000	0.9413	0.6305	0.8835

for imaginary quadratic number field based cryptography; this scenario amounts to using a base ideal of small norm.

It is possible to improve the fixed-base scalar multiplication algorithms in general at the cost of storing some precomputed values involving the fixed base, for example, using window methods or joint sparse form scalar expansions [11]. These methods effectively reduce the expected density of non-zero values in the corresponding binary expansion of the scalar by precomputing a number of small multiples of the base. Similarly, signature verification can be improved using multi-exponentiation techniques [11]. It should be relatively straightforward to generalize these methods to the real model as well, at which point more extensive comparisons between the imaginary and real versions of these improvements can be obtained.

Our second scalar multiplication primitive, VAR-DIST2(D, n), does not offer any improvements in efficiency to the imaginary case, but by minimizing the number of necessary adjustment baby steps, this algorithm requires only $\lceil g/2 \rceil$ baby steps more, regardless of the size of the scalar. It is an open problem to take advantage of the existence of the faster baby step operation to improve this algorithm further.

Special classes of curves, or curves with certain properties, frequently allow for faster arithmetic than generic curves. For example, Gaudry [8] developed fast scalar

multiplication formulas on the Kummer surface of a genus 2 imaginary hyperelliptic curve. It is of interest to investigate if and how such constructions can be extended to real hyperelliptic curves.

As mentioned in Section 5, there are some open problems related to the security of our protocols. It is not known whether differential power analysis attacks as in [2] can be adapted to exploit situations where our heuristics fail and degenerate divisors arise. Pohlig-Hellman type attacks that exploit the situation where the regulator is smooth have also not been investigated. In fact, as pointed out in [12], although the infrastructure DLP is very closely related to the DLP in the Jacobian, very little work on it was been published. The authors are unaware of any implementations and numerical work on this problem at all. An investigation of methods for constructing real curves of cryptographic size for which we can easily compute the regulator is also required.

Finally, it would be very interesting to see how the ideas of our improvements can be applied to real quadratic number fields. We believe that this is possible to some extent, but this generalization is far from trivial due to the fact that distances in the infrastructure are real numbers as opposed to integers. For an excellent exposition on the infrastructure in real quadratic number fields with an explanation via Arakelov class groups we refer to [24].

Acknowledgements

We would like to thank the referees very much for their valuable comments and suggestions.

Appendix A. Formulas for Divisor Arithmetic

In this appendix, we give the precise formulas for divisor arithmetic used in the implementation described in Section 6. These formulas are for curves defined over odd characteristic finite fields — the analog for the even characteristic case can be derived easily based on the arithmetic descriptions in [12] and [13]. The formulas are essentially taken directly from [26] with the following exceptions:

- We represent the divisor D as $\mathrm{div}(a, b, c)$ with $c = (b^2 - f)/a$, where the curve is given by $y^2 = f$. The formulas we used compute c for free as part of the reduction algorithm, and having it available simplifies the doubling (adding a divisor to itself) and baby step computations.
- We assume Heuristics H1 and H2. This simplifies the giant step formulas in that we can assume that the output of any gcd computations will be 1.

Throughout, $lc(a)$ denotes the leading coefficient of $a \in \mathbb{F}_q[x]$. In the real setting, we assume that s, the polynomial part of $\sqrt{f}$, is precomputed.

REDUCE_IMAG($\mathrm{div}(a, b, c)$)
  // outputs the reduced divisor equivalent to $\mathrm{div}(a, b, c)$
  while ($\deg(a) > g$)
    $a' = c$
    $q = -b/c$, $r = -b \bmod c$
    $c = a + q(b - r)$, $b = r$, $a = a'$
  output $\mathrm{div}(a/lc(a), b, c * lc(a))$

REDUCE_REAL($\mathrm{div}(a, b, -c)$)
  // outputs a reduced divisor equivalent to $\mathrm{div}(a, b, -c)$

while $(\deg(a) > g)$
$q = (s+b)/a$, $r = (s+b) \bmod a$
$b' = s - r$
$a' = c + q(b - b')$
$b = b'$, $c = a$, $a = a'$
output $\mathrm{div}(a/lc(a), b, c * (-lc(a)))$

ADD$(\mathrm{div}(a_1, b_1, c_1), \mathrm{div}(a_2, b_2, c_2))$
// outputs the reduced divisor $\mathrm{div}(a_3, b_3, c_3) = \mathrm{div}(a_1, b_1, c_1) \oplus \mathrm{div}(a_2, b_2, c_2)$
Solve $\gcd(a_2, a_1) = G = Xa_2 + Ya_1$ for $G, X \in \mathbb{F}_q[x]$
$U = X(b_1 - b_2) \bmod a_1$
$a_3 = a_1 a_2$, $b_3 = b_2 + a_2 U$, $c_3 = (b_3^2 - f)/a_3$
$(c_3 = (f - b_3^2)/a_3$ in the real case)
output REDUCE$(\mathrm{div}(a_3, b_3, c_3))$
(using either REDUCE_IMAG or REDUCE_REAL)

DOUBLE$(\mathrm{div}(a, b, c))$
// outputs the reduced divisor $\mathrm{div}(a_2, b_2, c_2) = \mathrm{div}(a, b, c) \oplus \mathrm{div}(a, b, c)$
Solve $\gcd(2b, a) = G = X(2b) + Ya$ for $G, X \in \mathbb{F}_q[x]$
$U = -cX \bmod a_1$
$a_2 = a_1^2$, $b_2 = b + aU$, $c_2 = (b_2^2 - f)/a_2$
$(c_2 = (f - b_2^2)/a_2$ in the real case)
output REDUCE$(\mathrm{div}(a_2, b_2, c_2))$
(using either REDUCE_IMAG or REDUCE_REAL)

BABY$(\mathrm{div}(a, b, c))$
// performs one baby step on $\mathrm{div}(a, b, c)$
$q = (s+b)/a$, $r = (s+b) \bmod a$
$b' = s - r$
$a' = q(b - b') - c$
$b = b'$, $c = a$, $a = a'$
output $\mathrm{div}(a/lc(a), b, c * (-lc(a)))$

References

[1] J. Achter, *The distribution of class groups of function fields*, J. Pure Appl. Algebra, **204** (2006), 316–333.

[2] R. Avanzi, *Countermeasures against differential power analysis for hyperelliptic curve cryptosystems*, in "Cryptographic Hardware and Embedded Systems – CHES 2003," Lect. Notes Comput. Sci., Vol. 2779, Springer-Verlag, (2003), 366–381.

[3] D. G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Math. Comp., **48** (1987), 95–101.

[4] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen and F. Vercouteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography," Chapman & Hall/CRC, Boca Raton, 2006.

[5] A. Enge, *How to distinguish hyperelliptic curves in even characteristic*, in "Public-Key Cryptography and Computational Number Theory," De Gruyter, (2001), 49–58.

[6] S. Erickson, M. J. Jacobson, Jr., N. Shang, S. Shen and A. Stein, *Explicit formulas for real hyperelliptic curves of genus 2 in affine representation*, in "International Workshop on the Arithmetic of Finite Fields – WAIFI 2007," Lect. Notes Comput. Sci., Vol. 4547, Springer-Verlag, (2007), 202–218.

[7] E. Friedman, L. C. Washington, *On the distribution of divisor class groups of curves over a finite field*, in "Théorie des Nombres (Québec, PQ)," de Gruyter, (1989), 227–239.

[8] P. Gaudry, *Fast genus 2 arithmetic based on Theta functions*, Preprint.

[9] P. Gaudry, E. Thomé, N. Thériault and C. Diem, *A double large prime variation for small genus hyperelliptic index calculus*, Math. Comp., **76** (2007), 475–492.
[10] L. Goubin, *A refined power-analysis attack on elliptic curve cryptosystems*, in "Public Key Cryptography – PKC 2003," Lect. Notes Comp. Sci., Vol. 2567, Springer-Verlag, (2002), 199–210.
[11] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography," Springer-Verlag, New York, 2004.
[12] M. J. Jacobson, Jr., A. J. Menezes and A. Stein, *Hyperelliptic curves and cryptography*, in "High Primes and Misdemeanors: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams," Fields Institute Communications, Vol. 41, American Mathematical Society, (2004), 255–282.
[13] M. J. Jacobson, Jr., R. Scheidler and A. Stein, *Fast Arithmetic on Hyperelliptic Curves Via Continued Fraction Expansions*, in "Advances in Coding Theory and Cryptology," Series on Coding Theory and Cryptology, 2. World Scientific Publishing Co. Pte. Ltd., (2007), 201–244.
[14] M. J. Jacobson, Jr. and A. J. van der Poorten, *Computational aspects of NUCOMP*, in "Algorithmic Number Theory - ANTS-V," Lect. Notes Comput. Sci., Vol. 2369, Springer-Verlag, (2002), 120–133.
[15] M. Katagi, I. Kitamura, T. Akishita and T. Takagi, *Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors*, in "Information Security Applications 5th International Workshop - WISA 2004", Lect. Notes Comput. Sci., Vol. 3325, Springer-Verlag, (2005), 345–359.
[16] N. Koblitz, *Hyperelliptic cryptosystems*, J. Cryptology, **1** (1989), 139–150.
[17] T. Lange, *Formulae for arithmetic on genus 2 hyperelliptic curves*, Appl. Algebra Eng. Commun. Comput., **15** (2005), 295–328.
[18] A. J. Menezes, Y. -H. Wu, and R. J. Zuccherato, *An elementary introduction to hyperelliptic curves*, in "Algebraic Aspects of Cryptography," Algorithms and Computation in Mathematics, Vol. 3, Springer Verlag, Berlin, 1998, 155–178.
[19] National Institute of Standards and Technology (NIST), *Recommendation on key establishment schemes*, NIST Special Publication 800-56, January 2003.
[20] S. Paulus, H. -G. Rück, *Real and imaginary quadratic representations of hyperelliptic function fields*, Math. Comp., **68** (1999), 1233–1241.
[21] J. Pelzl, T. Wollinger, and C. Paar, *Low cost security: explicit formulae for genus-4 hyperelliptic curves*, in "Selected Areas in Cryptography — SAC 2003," Lect. Notes Comput. Sci., **3006** (2003), 1–16.
[22] R. Scheidler, *Cryptography in quadratic function fields*, Des. Codes Cryptography, **22** (2001), 239–264.
[23] R. Scheidler, A. Stein and H. C. Williams, *Key-exchange in real quadratic congruence function fields*, Des. Codes Cryptography, **7** (1996), 153–174.
[24] R. Schoof, *Computing Arakelov class groups*, To appear in "Surveys in Algorithmic Number Theory", Mathematical Sciences Research Institute Publications, Cambridge University Press, See `http://www.mat.uniroma2.it/~schoof/infranew2.pdf`
[25] V. Shoup, *NTL: A library for doing number theory*, Software, 2001, See `http://www.shoup.net/ntl`
[26] A. Stein, *Sharp upper bounds for arithmetics in hyperelliptic function fields*, J. Ramanujan Math. Soc., **16** (2001), 1–86.
[27] A. Stein, E. Teske, *Optimized baby step-giant step methods in hyperelliptic function fields*, J. Ramanujan Math. Soc., **20** (2005), 1–32.
[28] A. Stein, H. C. Williams, *Some methods for evaluating the regulator of a real quadratic function field*, Experimental Mathematics, **8** (1999), 119–133.
[29] T. Wollinger, J. Pelzl and C. Paar, *Cantor versus Harley: optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems*, IEEE Trans. Computers, **54** (2005), 861–872.

Received September 2006; revised May 2007.

E-mail address: jacobs@cpsc.ucalgary.ca
E-mail address: rscheidl@math.ucalgary.ca
E-mail address: astein@uwyo.edu