Tatra Mt. Math. Publ. **47** (2010), 31–65
DOI: 10.2478/v10127-010-0030-9

TATRA MOUNTAINS
Mathematical Publications

CRYPTOGRAPHIC ASPECTS OF REAL HYPERELLIPTIC CURVES

Michael J. Jacobson, Jr. — Renate Scheidler — Andreas Stein

ABSTRACT. In this paper, we give an overview of cryptographic applications using real hyperelliptic curves. We review previously proposed cryptographic protocols and discuss the infrastructure of a real hyperelliptic curve, the mathematical structure underlying all these protocols. We then describe recent improvements to infrastructure arithmetic, including explicit formulas for divisor arithmetic in genus 2, and advances in solving the infrastructure discrete logarithm problem, whose presumed intractability is the basis of security for the related cryptographic protocols.

1. Introduction and motivation

In their highly influential 1976 paper [8], Diffie and Hellman introduced public-key cryptography to the research community, and presented the first protocol by which two parties can agree on a common secret by exchanging information across a public channel only. While the original Diffie-Hellman key agreement protocol was introduced in the context of finite fields, it can be generalized to other suitable finite abelian groups. Suitable in this context means that the group must be sufficiently large, its elements should have a compact representation that supports efficient arithmetic, and the underlying Diffie--Hellman problem (and discrete logarithm problem) should be computationally intractable. Prominent examples of such groups include the group of rational points on an elliptic curve over a finite field [32], [28] and the Jacobian of an imaginary hyperelliptic curve of genus 2 (or 3) over a finite field [29]. The use of real hyperelliptic curves for discrete logarithm based cryptography was subsequently proposed in [39].

2010 Mathematics Subject Classification: Primary 94A60; Secondary 14H45, 11Y40, 11Y16.
Keywords: hyperelliptic curve, Jacobian, infrastructure, public-key cryptography, discrete logarithm.
The first and second authors' research is supported by NSERC of Canada.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Generally speaking, a hyperelliptic curve over a finite field $\mathbb{F}_q$ is given by an equation of the form

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

where $h(x)$ and $f(x)$ are polynomials with coefficients in $\mathbb{F}_q$ that satisfy certain conditions. Depending on these conditions, C is a *real* or *imaginary* model; for example, if q is odd, then $h(x) = 0$, and $f(x)$ is monic of even degree if C is real and of odd degree if C is imaginary. Geometrically, the main distinction between the two models is that real models have two points at infinity, while imaginary models have only one (ramified) point at infinity. Algebraically, the function field $\mathbb{F}_q(x, y)$ of a real, respectively, imaginary hyperelliptic curve exhibits a structural behaviour that is very similar to that of a real, respectively, imaginary quadratic number field. While imaginary models have traditionally been more popular in cryptographic applications, real models are more general; every imaginary hyperelliptic curve can be transformed into a real curve over the same base field $\mathbb{F}_q$, while the reverse process may require a significantly larger base field.

Every hyperelliptic curve has an associated group referred to as its *Jacobian* whose cardinality is exponentially large in the size of the curve. Elements of this group are certain equivalence classes of infinite cardinality that are uniquely representable by so-called *reduced* elements whose size is small. The Jacobian of an imaginary model supports efficient arithmetic that is well understood and a discrete logarithm problem that appears to be difficult. Thus, Jacobians of imaginary hyperelliptic curves (of small genus) are highly suitable for cryptographic use, and a large body of literature has been devoted to this subject. While Jacobians of real models can also potentially be employed in cryptography, their arithmetic tends to be somewhat slower than arithmetic on their imaginary cousins. As a result, real hyperelliptic curve cryptography is conducted in a different structure referred to as the *infrastructure* of the curve.

The infrastructure is a large finite subset of the identity class of the Jacobian whose cardinality is in general roughly the same as that of the Jacobian. Elements of the infrastructure have a small representation analogous to that of reduced representatives in the imaginary Jacobian setting, and can be endowed with a *distance* function that imposes an ordering on these elements. The group operation on the Jacobian, generally referred to as a *giant step*, can in fact also be applied to infrastructure elements. Under giant steps, distances are almost additive. As a result, the infrastructure is almost an abelian group under giant steps that only barely fails associativity.

The main difference between the two structures is that the infrastructure supports a second *baby step* operation that has no analogue in the Jacobian. Baby steps move iteratively from one element to the next in the distance-imposed ordering, and are much faster than giant steps. Distance advances effected by both baby steps and giant steps can be ascertained approximately, and in fact,

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

in the cryptographically interesting case when q is large, they can be predicted exactly with very high heuristic probability. This leads to a third operation on the infrastructure that is analogous to scalar multiplication. It computes for any scalar n and any infrastructure element D of (possibly unknown) distance $\delta(D)$ the infrastructure element E of distance $\delta(E) = n\delta(D)$. This in turn makes it possible to efficiently compute for any non-negative integer d the infrastructure element of distance d.

Diffie-Hellman key agreement using the Jacobian is conducted as in any additive abelian group, via reduced representatives. Two parties, Alice and Bob, agree on an imaginary hyperelliptic curve and a public random class in its Jacobian, represented by a reduced element D. Alice generates a random secret scalar a and sends the reduced element D_a in the class of aD to Bob. Similarly, Bob sends the reduced element D_b in the class of bD to Alice, where b is his secret random scalar. The common key is the reduced element D_{ab} in the class of $abD = a(bD) = b(aD)$. The underlying discrete logarithm problem reads as follows: given D and the reduced element in the class of nD, find n. This problem has undergone considerable study and is widely believed to be intractable for hyperelliptic curves of small genus.

In the infrastructure Diffie-Hellman analogue, Alice and Bob agree on a real hyperelliptic curve. They respectively compute and exchange the infrastructure elements D_a of distance a and D_b of distance b, where a and b are their respective scalars. Each party can in turn compute the infrastructure element D_{ab} of distance ab which is the common key. Other discrete logarithm based protocols, such as the public-key cryptosystem and signature scheme due to ElGamal [9], can similarly be adapted to the infrastructure setting. The infrastructure discrete logarithm problem is the problem of finding the distance of a given infrastructure element.

The discrete logarithm problems in the Jacobian and the infrastructure appear to be equally intractable. Algorithms for extracting discrete logarithms in the Jacobian, such as the index calculus method, have been adapted to the infrastructure setting and yield equal asymptotic runtimes. In fact, there is a distance preserving embedding that maps the infrastructure into a very large cyclic subgroup of the Jacobian. Hence, any algorithm for finding discrete logarithms in a cyclic group can immediately be applied to the infrastructure.

Real models of hyperelliptic curves have not been widely investigated for cryptographic applications because their arithmetic was traditionally thought to be more cumbersome and less efficient than Jacobian arithmetic using imaginary models. However, recent advances in this area have shown that real model arithmetic can be competitive to its imaginary counterpart, since it is able to take advantage of certain speed-ups in infrastructure scalar multiplication. Any scalar multiplication algorithm in the Jacobian is comprised of a suitable series of giant

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

steps. The crucial point that makes the infrastructure scenario attractive is that here, a large number of these giant steps can be replaced by baby steps which are much faster.

The goal of this article is to explain the mathematical background of real hyperelliptic curves, describe the realization of their arithmetic as well as their applications to cryptography, and present recent progress on this subject. We will demonstrate that real hyperelliptic curves are readily available for efficient use in cryptography, and explain how to speed up all previously known cryptographic protocols on real hyperelliptic curves so that these protocols are comparable in speed with their imaginary hyperelliptic curve analogues.

This paper is organized as follows. In Section 2, we discuss imaginary and real hyperelliptic curves as well as transformations between the real and the imaginary model. Section 3 provides a review of divisors and the Jacobian of a hyperelliptic curve, and Section 4 describes the infrastructure of a real hyperelliptic curve. Scalar multiplication is the main ingredient of hyperelliptic curve based cryptographic protocols; recent improvements to this operation in the infrastructure are presented in Section 5, and the use of these improved versions in infrastructure based cryptography is illustrated in Section 6. A brief discussion of infrastructure divisor arithmetic is provided in Section 7. We will survey the mathematical security of real hyperelliptic curve based cryptosystems in Section 8. A short history of real hyperelliptic curve cryptography as well as an overview of related relevant work are given in Section 9, followed by final conclusions and further open problems.

2. Hyperelliptic curves

Given the vast body of literature on hyperelliptic curves and their cryptographic applications, it suffices to cite the surveys [31], [37], [23] and the reference work [7] here. Recent advances on real hyperelliptic curve protocols were described in [24].

Throughout this paper, let q be any prime power, $\mathbb{F}_q$ a finite field of q elements, $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$, and $\overline{\mathbb{F}}_q$ an algebraic closure of $\mathbb{F}_q$. Also, let $\mathbb{F}_q[x]$ and $\mathbb{F}_q(x)$ denote the ring of polynomials and the field of rational functions, respectively, over $\mathbb{F}_q$. For any non-zero polynomial $F \in \mathbb{F}_q[x]$, let $\deg(F)$ denote the degree of F.

For our purposes, a *hyperelliptic curve* of *genus* g over $\mathbb{F}_q$ is defined to be 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. One distinguishes two models as follows:

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

- *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, $\deg(h) = g+1$, $\deg(f) \leq 2g+2$, and the coefficient of x^{2g+2} in f is of the form $s^2 + s$ for some $s \in \mathbb{F}_q$.

Of cryptographic interest are the cases of low genus, in particular the cases $g = 1$ (elliptic curves) and $g = 2$. We also note that for all q, a real model as given in (2.1) is always isomorphic over $\mathbb{F}_q$ to a real model of the form $z^2 + h(x)z = F(x)$ with $\deg(F) \leq g$; see [1, Theorem 9.1].

If $\mathbb{E} = \overline{\mathbb{F}}_q$ or $\mathbb{E} = \mathbb{F}_{q^n}$ for some $n \in \mathbb{N}$, then the set of *(finite)* $\mathbb{E}$*-rational* points on C is the set

$$C(\mathbb{E}) = \big\{(a,b) \in \mathbb{E} \times \mathbb{E} \mid b^2 + h(a)b = f(a)\big\}.$$

Hyperelliptic curves also have *infinite points* (or *points at infinity*). Specifically, any imaginary model has one such point, while every real model has two points at infinity. The *hyperelliptic involution* on C sends any finite point $p = (a,b) \in C(\mathbb{E})$ to its *opposite* $\overline{p} = \big(a, -b-h(a)\big) \in C(\mathbb{E})$. It also sends ∞, the point at infinity of an imaginary hyperelliptic curve, to itself and permutes (i.e., swaps) the two infinite points $\infty, \overline{\infty}$ of a real hyperelliptic curve.

The *coordinate ring* of C is the ring $\mathbb{F}_q[C] = \big\{A + By \mid A, B \in \mathbb{F}_q[x]\big\}$; its field of fractions $\mathbb{F}_q(C) = \big\{A + By \mid A, B \in \mathbb{F}_q(x)\big\}$ is the *function field* of C. The hyperelliptic involution on C extends to maps on $\mathbb{F}_q[C]$ and $\mathbb{F}_q(C)$, sending $\alpha = A + By$ to $\overline{\alpha} = A - Bh - By$.

A (finite or infinite) point is said to be *ramified* if it is equal to its own opposite, and *unramified* otherwise. The finite ramified points on C are exactly the points (a,b) with $h(a) + 2b = 0$, i.e., the points $(a,0)$ with $f(a) = 0$ if q is odd, and (a,b) with $h(a) = 0$ and $f(a) = b^2$ if q is even; if C is imaginary, then ∞ is also ramified. Since C is non-singular, we must have $h'(a)b \neq f'(a)$ for any finite ramified point $p = (a,b)$, where for any polynomial $F(x) \in \mathbb{F}_q[x]$, $F'(x)$ denotes the formal derivative of F with respect to x.

Under certain conditions, there are genus and function field preserving birational transformations over $\mathbb{F}_q$ from one model to the other (see also [35] for the case q odd).

Proposition 2.1. *Any imaginary hyperelliptic curve $C : y^2 + h(x)y = f(x)$ of genus g with an unramified $\mathbb{F}_q$-rational finite point $p = (a,b)$ is birationally equivalent to a real hyperelliptic curve $C' : v^2 + H(u)v = F(u)$ of genus g with $F(0) = H(0) = 0$, and $\mathbb{F}_q(C) = \mathbb{F}_q(C')$. Here,*

$$u = \frac{1}{x-a}\,, \quad v = \frac{y}{b(x-a)^{g+1}} = \frac{yu^{g+1}}{b}\,,$$

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

$$F(u) = \frac{u^{2g+2}}{e} f\left(\frac{1}{u} + a\right), \quad H(u) = \frac{u^{g+1}}{e^2} h\left(\frac{1}{u} + a\right),$$

with $e = b$ if q is odd and $e = h(a)$ if q is even.

Any real hyperelliptic curve $C : y^2 + h(x)y = f(x)$ of genus g with a ramified $\mathbb{F}_q$-rational finite point $p = (a, b)$ is birationally equivalent to an imaginary model $C' : v^2 + H(u)v = F(u)$ of genus g, and $\mathbb{F}_q(C) = \mathbb{F}_q(C')$. Here,

$$u = \frac{1}{e(x-a)}, \quad v = \frac{y+b}{(e(x-a))^{g+1}} = (y+b)u^{g+1},$$

$$F(u) = u^{2g+2} f\left(\frac{1}{eu} + a\right) + bG(u) + b^2 u^{2g+2}, \quad G(u) = u^{g+1} h\left(\frac{1}{eu} + a\right),$$

with $e = f'(a) + bh'(a)$.

Proof. Suppose there exists $p = (a, b) \in C(\mathbb{F}_q)$, and write

$$f(x) = \sum_{i=0}^{2g+2} f_i (x-a)^i$$

with $f_i \in \mathbb{F}_q$ for $0 \le i \le 2g+2$, and

$$h(x) = \sum_{i=0}^{g+1} h_i (x-a)^i$$

with $h_i \in \mathbb{F}_q$ (note that $h_i = 0$ for $0 \le i \le g+1$ if q is odd). Setting

$$t = (x-a)^{-1} \quad \text{and} \quad w = y(x-a)^{-g-1} = yt^{g+1},$$

and subsequently multiplying by t^{2g+2}, transforms C into $C'' : w^2 + G(t)w = E(t)$, where $\mathbb{F}_q(C'') = \mathbb{F}_q(C)$ and

$$\begin{aligned} G(t) &= t^{g+1} h(t^{-1} + a) = h_0 t^{g+1} + h_1 t^g + \cdots + h_{g+1}, \\ E(t) &= t^{2g+2} f(t^{-1} + a) = f_0 t^{2g+2} + f_1 t^{2g+1} + \cdots + f_{2g+2}. \end{aligned}$$

Assume first that C is imaginary. Then

$$h_{g+1} = f_{2g+2} = 0,$$

so

$$G(0) = H(0) = 0.$$

Suppose also that $p \neq \overline{p}$. Then $2b \neq h_0$. If q is odd, then $f_0 = b^2 \neq 0$, so $\deg(E) = 2g+2$. Hence, setting $(u, v) = (t, b^{-1}w)$ and dividing by b^2 generates the desired curve C'. If q is even, then $h_0 \neq 0$, so $\deg(G) = g+1$. Setting $(u, v) = (t, h_0^{-1}w)$ and dividing by h_0 produces the curve $C' : v^2 + H(u)v = F(u)$, where $H(u) = G(u)/h_0$ is monic of degree $g+1$, and $F(u) = E(u)/h_0^2$ has coefficient f_0/h_0^2 at t^{2g+2}. Now $b^2 + h(a)b = f(a)$ implies $\big(b/h(a)\big)^2 + b/h(a) = f(a)/h(a)^2 = f_0/h_0^2$, so this coefficient is of the required form.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Next, assume that C is real, with $p = \overline{p}$. Then $2b + h_0 = 0$, so $b = f_0 = 0$ if q is odd, and $h_0 = 0$, $f_0 = b^2$ if q is even. Thus, $\deg(G) \leq g$ if q is even. Set

$$\tilde{w} = w + bt^{g+1} \quad \text{and} \quad \tilde{E}(t) = E(t) + bG(t)t^{g+1} + b^2t^{2g+2};$$

note that $\tilde{w} = w$ and $\tilde{E} = E$ if q is odd. Adding b^2t^{2g+2} to C'', and noting that $2b = 0$, yields the curve $\tilde{C} : \tilde{w}^2 + G(t)\tilde{w} = \tilde{E}(t)$. The coefficient of t^{2g+2} in $\tilde{E}(t)$ is $f_0 + b^2$, which vanishes, and that of t^{2g+1} is $e = f_1 + bh_1$, which is non-zero due to non-singularity. It follows that $\deg(\tilde{E}) = 2g+1$. Setting $(u, v) = (t/e, \tilde{w}/e^{g+1})$ and dividing by e^{2g+2} now yields the desired curve C'. □

Proposition 2.1 shows that real models are much more general than imaginary models. If q is odd and exceeds $2g + 2$, then any imaginary hyperelliptic curve of genus g over $\mathbb{F}_q$ has an unramified point, since $f(x)$ can have at most $2g+2$ roots; for q even, $q > g+1$ is even sufficient. Thus, over sufficiently large base fields, such as cryptographically suitable fields, imaginary curves can always be transformed to their real counterparts of the same genus. However, a transformation in the opposite direction is only possible over the splitting field of $h(x)$ if q is even, and that of $f(x)$ if q is odd. This field extension can have degree over $\mathbb{F}_q$ as large as $g + 1$ if q is even and $2g + 2$ if q is odd. Arithmetic on an imaginary model over this larger base field would be highly inefficient compared to arithmetic on a real model over $\mathbb{F}_q$.

3. Divisors and the Jacobian

We continue to let C be a hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$ as given in (2.1). Let S denote the set of infinite points on C, i.e., $S = \{\infty\}$ if C is imaginary, and $S = \{\infty, \overline{\infty}\}$ if C is real.

A *divisor* D on C is a formal finite sum of (finite and infinite) points p on C; write $D = \sum_p n_p p$ where $n_p \in \mathbb{Z}$ and $n_p = 0$ for all but finitely many p. The *support* of D, denoted by $\mathrm{supp}(D)$, is the set of points p on C for which $n_p \neq 0$, and the *degree* of D is $\deg(D) = \sum_p n_p \in \mathbb{Z}$. A *principal* divisor has the form $D = \sum_p v_p(\alpha)p$ for some $\alpha \in \mathbb{F}_q(C)^*$, where $v_p(\alpha)$ is the order of vanishing of α at p; write $D = \mathrm{div}(\alpha)$. Every principal divisor has degree zero.

A divisor D is *finite* if $\mathrm{supp}(D) \cap S = \emptyset$ and *infinite* if $\mathrm{supp}(D) \subseteq S$. For any divisor $D = \sum_p n_p p$, let $D_S = \sum_{p \notin S} n_p p$ denote its finite portion. If D has degree zero, then there is a unique representation

$$D = \begin{cases} D_S - \deg(D_S)\infty & \text{if } C \text{ is imaginary,} \\ D_S - \deg(D_S)\infty + n_{\overline{\infty}}(\overline{\infty} - \infty) & \text{if } C \text{ is real.} \end{cases} \tag{3.1}$$

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

The hyperelliptic involution on C can be linearly extended to send D_S to $\overline{D}_S = \sum_{p \notin S} n_p \overline{p}$, and hence D to the divisor

$$\overline{D} = \begin{cases} \overline{D}_S - \deg(D_S)\infty & \text{if } C \text{ is imaginary,} \\ \overline{D}_S - \deg(D_S)\infty - (\deg(D_S) + n_{\overline{\infty}})(\overline{\infty} - \infty) & \text{if } C \text{ is real.} \end{cases}$$

We note that $D + \overline{D}$ is a principal divisor.

Recall that the *Frobenius* automorphism π_q is the map on $\overline{\mathbb{F}}_q$ that sends every element in $\overline{\mathbb{F}}_q$ to its qth power. The map π_q can be defined on finite points on C via action on the coordinates of the point, and on infinite points as the identity. This action extends linearly to divisors on C. A divisor D on C is said to be *defined over* $\mathbb{F}_q$ if it is invariant under π_q. Henceforth, we only consider divisors D defined over $\mathbb{F}_q$. Note that D is defined over $\mathbb{F}_q$ if and only if D_S is defined over $\mathbb{F}_q$.

Clearly, the divisors defined over $\mathbb{F}_q$ form an infinite abelian group $\mathcal{D}$ under formal addition, of which the divisors of degree zero defined over $\mathbb{F}_q$ form an infinite subgroup $\mathcal{D}_0$. Moreover, the set of principal divisors $\mathcal{P}$ defined over $\mathbb{F}_q$ is an infinite subgroup of $\mathcal{D}_0$, and $\operatorname{div}(\alpha) + \operatorname{div}(\beta) = \operatorname{div}(\alpha\beta)$ for $\alpha, \beta \in \mathbb{F}_q(C)^*$. The factor group $\mathcal{J} = \mathcal{D}_0/\mathcal{P}$ is the *(degree zero) divisor class group* or *Jacobian* of C; it is a finite abelian group whose order $h = |\mathcal{J}|$ is the *degree zero divisor class number*, or simply the *class number*, of C. The Hasse-Weil bounds (see for example [47, Theorem 5.2.3, p. 198,]) state that $(\sqrt{q} - 1)^{2g} \leq h \leq (\sqrt{q} + 1)^{2g}$, implying $h \approx q^g$. It follows that the class number is exponentially large in the size of the curve, i.e., in $\log(q)$ and g.

The finite divisors defined over $\mathbb{F}_q$ are in one-to-one correspondence with the fractional ideals of the coordinate ring $\mathbb{F}_q[C]$. A finite divisor $D = \sum_{p \notin S} n_p p$ on C defined over $\mathbb{F}_q$ is said to be *effective* if $n_p > 0$ for all $p \in \operatorname{supp}(D)$. These divisors correspond exactly to the integral $\mathbb{F}_q[C]$-ideals. D is *semi-reduced* if it is non-zero, effective, and for all $p \in \operatorname{supp}(D)$, $n_p = 1$ if p is ramified and $\overline{p} \notin \operatorname{supp}(D)$ if p is unramified. In this case, D can be represented by two polynomials $Q, P \in \mathbb{F}_q[x]$ as follows: for every $p \in \operatorname{supp}(D)$, write $p = (a_p, b_p)$ with $a_p, b_p \in \overline{\mathbb{F}}_q$. Set

$$Q(x) = \prod_{p \in \operatorname{supp}(D)} (x - a_p)^{n_p},$$

and let $P(x)$ be any interpolation polynomial defined via $b_p = -P(a_p)$ for all p in $\operatorname{supp}(D)$. Then Q is monic of degree $\deg(D)$ and divides $f + hP - P^2$. The divisor D is uniquely represented by Q and $P \pmod{Q}$, so we can write $D = (Q, P)$; this is generally referred to as the *Mumford representation* of D. The semi-reduced divisors $D = (Q, P)$ correspond to primitive $\mathbb{F}_q[C]$-ideals, as these ideals are exactly the free $\mathbb{F}_q[x]$-modules of rank 2 generated by Q and $P + y$.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Note that if $D = (Q, P)$ is semi-reduced, then $\overline{D} = (Q, -h - P)$ is also semi-reduced. A semi-reduced divisor $D = (Q, P)$, and its corresponding primitive ideal in $\mathbb{F}_q[C]$, are *reduced* if $\deg(Q) \leq g$.

The above notions naturally extend from finite divisors to arbitrary degree zero divisors. A degree zero divisor D is simply called (semi-)reduced if its finite portion D_S is (semi-)reduced. The $\mathbb{F}_q[C]$-ideal corresponding to a reduced divisor is also said to be reduced. Note that if D is reduced, then so is $\overline{D}$.

If C is imaginary, then every divisor class in $\mathcal{J}$ contains a unique reduced divisor. Thus, arithmetic in the Jacobian of C can be performed via reduced representatives; given reduced divisors D, D' on C, one defines $D \oplus D'$ to be the unique reduced divisor in the class of $D+D'$. The divisor $D \oplus D'$ can be computed, for example, using Cantor's algorithm [4], the NUCOMP algorithm [25], or (in the case of small genus) explicit formulas; see [7, Chapter 14]. The first two of these methods apply to curves of arbitrary genus g and require $O(g^2)$ operations in the base field $\mathbb{F}_q$.

Using the operation $\oplus$ on the Jacobian of an imaginary hyperelliptic curve, it is straightforward to conduct discrete logarithm based cryptography in this group. Diffie-Hellman key agreement using imaginary hyperelliptic curves was first presented by Koblitz [29], and a description of the Digital Signature Algorithm (DSA) on imaginary hyperelliptic curves is given in [7, pp. 570f.]. Other discrete logarithm based schemes can be adapted to this setting in a similar manner.

If C is real, then every divisor class contains a unique reduced divisor D such that $n_{\overline{\infty}}$ as given in (3.1) lies in a specified range of length $g + 1 - \deg(D_S)$. For example, [35] proposed the interval $[0, g - \deg(D_S)]$, and more recently, [15] introduced a *balanced* divisor representation where the interval containing $n_{\overline{\infty}}$ is centered around $\lceil \deg(D_S)/2 \rceil$. The term stems from the fact that balanced divisors have essentially equal contributions at the two infinite places.

Once again, these unique representatives allow for Jacobian arithmetic, and hence discrete logarithm based cryptography, on real hyperelliptic curves. On both imaginary and real models, addition of divisor classes can be performed in $O(g^2)$ operations in the base field $\mathbb{F}_q$ [43], [35]. However, the resulting operation in the real setting is potentially slower than in the imaginary case, since the unique reduced divisor in the class of $D + D'$ is obtained by computing $D \oplus D'$ as in the imaginary setting, and subsequently applying "adjustment" baby steps to guarantee an $n_{\overline{\infty}}$-value in the desired range. The Paulus-Rück representation [35] requires up to g such adjustment steps, while the balanced representation of [15] eliminates all but at most one of these baby steps.

Before the discovery of balanced divisors, arithmetic in the Jacobian of a real hyperelliptic curve was rightfully regarded as clearly less efficient than in the

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

imaginary case. As a result, real models utilized instead a different group-like structure referred to as the *(principal) infrastructure* of C.

4. The infrastructure of a real hyperelliptic curve

For this section, we restrict C to be a real hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$. In this context, the infinite degree zero divisor $\overline{\infty} - \infty$, which already appeared in (3.1), plays an important role. The order R of the divisor class of $\overline{\infty} - \infty$ in the Jacobian $\mathcal{J}$ of C is the *regulator* of C. Thus, the divisor $R(\overline{\infty} - \infty)$ is principal, and if we write $R(\overline{\infty} - \infty) = \mathrm{div}(\epsilon)$, then ϵ is a unit in $\mathbb{F}_q[C]$; it is called a *fundamental unit* of C, since every unit in $\mathbb{F}_q[C]$ is of the form $a\epsilon^m$ with $a \in \mathbb{F}_q^*$ and $m \in \mathbb{Z}$. The regulator R divides the class number h of C, and the quotient h/R is exactly the ideal class number of the coordinate ring $\mathbb{F}_q[C]$. Generically, the ideal class number is small, so $R \approx h \approx q^g$. In fact, the ideal class number is very frequently equal to one, in which case $\mathcal{J}$ is cyclic of order R and generated by the class of $\overline{\infty} - \infty$.

There are three ways of describing the *(principal) infrastructure* $\mathcal{R}$ of C. The first is to define $\mathcal{R}$ to be the set of all reduced principal finite divisors, or equivalently, the set of all reduced principal $\mathbb{F}_q[C]$-ideals; this was done in [39]. The second description stems from [25], [24] and specifies $\mathcal{R}$ to consist of all divisors of the form $D = D_S - \deg(D_S)\infty$, where D_S corresponds to a reduced principal $\mathbb{F}_q[C]$-ideal; note that all these divisors have degree zero and no support at $\overline{\infty}$, and they represent distinct divisor classes in $\mathcal{J}$. The third definition of the principal infrastructure, which we will use here, is to simply declare $\mathcal{R}$ to be the collection of all reduced principal divisors D with $0 \geq n_{\overline{\infty}} > -R$, where $n_{\overline{\infty}}$ is given by (3.1). The finite portions of the collection of divisors in $\mathcal{R}$ are identical in each of these descriptions; only the infinite portions differ. Furthermore, in all three cases, every $D \in \mathcal{R}$ is uniquely defined by its finite portion D_S, so henceforth, we will specify infrastructure divisors simply by their Mumford representation $D = (Q, P)$. Since Q, and hence also $P \pmod{Q}$, both have bounded degree, $\mathcal{R}$ is a finite set.

The reason that we chose the third description above is that it allows for the simplest definition of the *distance* of a divisor. Namely, the distance of $D \in \mathcal{R}$ is $\delta(D) = -n_{\overline{\infty}}$. Since no two infrastructure divisors have the same distance, this imposes an ordering on $\mathcal{R}$ by distance:

$$\mathcal{R} = \{D_1, D_2, \ldots, D_r\}, \quad 0 = \delta_1 < \delta_2 < \cdots < \delta_r < R\,,$$

where $D_1 = \mathbf{0}$ is the trivial divisor and $\delta_i = \delta(D_i)$ for $1 \leq i \leq r$. It can be shown that $\delta_2 = g + 1$, $\delta_i \leq \delta_{i-1} + g$ for $3 \leq i \leq r$, and $R \leq \delta_r + g$. It follows

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

inductively that $g + i - 1 \leq \delta_i \leq (i-1)g + 1$ for $2 \leq i \leq r$. This in turn implies $r + g \leq R \leq rg + 1$, so $|\mathcal{R}| = r \approx R \approx q^g$.

The operation $D_i \to D_{i+1}$ that moves iteratively forward through $\mathcal{R}$ is referred to as a *baby step*. A baby step computes the Mumford representation of D_{i+1} from that of D_i as well as the relative distance $\delta_{i+1} - \delta_i$. The baby step operation is cyclic, in the sense that when applied to D_r it yields D_1. From our previous remarks, a baby step results in an advance in distance of at most g (and $g+1$ for the very first baby step), which is very small compared to the total distance range of approximately q^g. This fact gives the baby step its name. Formulas for the baby step are given in, for example, [23], [25]. A baby step can be computed in $O(g)$ operations in $\mathbb{F}_q$ (see [43]). We point out that baby steps are closely related to the continued fraction expansion of a reduced quadratic irrationality in $\mathbb{F}_q(C)$ [25].

Besides the baby step, the set $\mathcal{R}$ supports a second operation, namely the operation $\oplus$ mentioned earlier in the context of imaginary hyperelliptic curves. This operation imposes an interesting structure on $\mathcal{R}$, whence the infrastructure derives its name. In fact, $\mathcal{R}$ is almost an abelian group with identity $D_1 = \mathbf{0}$ under the operation $\oplus$, failing only associativity. Note that the inverse of a divisor $D = (Q, P) \in \mathcal{R}$ different from D_1 is $\overline{D} = (Q, -h - P) \in \mathcal{R}$ of distance $\delta(\overline{D}) = R + \deg(Q) - \delta(D)$. Moreover, if $D, D' \in \mathcal{R}$, then

$$\delta(D \oplus D') = \delta(D) + \delta(D') - d \quad \text{with} \quad 0 \leq d \leq 2g\,. \tag{4.1}$$

Here, the "shortfall" d in distance is effectively computable as part of the giant step. The identity (4.1) implies that $\mathcal{R}$ is almost associative, in the sense that for $D, D', D'' \in \mathcal{R}$, $D \oplus (D' \oplus D'')$ and $(D \oplus D') \oplus D''$ are close (i.e., within $4g$) in distance. The binary operation $(D, D') \to D \oplus D'$ on $\mathcal{R}$ is called a *giant step*. A giant step computes the Mumford representation of $D \oplus D'$, given those of D and D', as well as the shortfall $d = \delta(D) + \delta(D') - \delta(D \oplus D')$ in distance. Thus, for any divisor D, the giant step $(D, D') \to D \oplus D'$ results in a distance leap of almost $\delta(D')$. The giant step derives its name from this fact. Recall that giant steps have quadratic asymptotic running time in the genus in terms of base field operations, while baby steps have linear time. Thus, baby steps are asymptotically much faster that giant steps; in fact, this holds in practice as well, even for very small genus.

Unfortunately, the fact that $\mathcal{R}$ is not associative under giant steps foils a direct adaptation of discrete logarithm based cryptography from the Jacobian to the infrastructure. If two parties were to perform for example Diffie-Hellman key agreement in $\mathcal{R}$ using giant steps, they would arrive at different divisors after execution of the protocol. This leads to the following concept, first introduced in [39], and more appropriately termed in [37]. For any $s \in [0, R-1]$, the infrastructure divisor *below* s is defined to be the unique divisor $D(s) = D_i \in \mathcal{R}$ with $\delta_i \leq s < \delta_{i+1}$ if $s < \delta_r$, and $\delta_r \leq s < R$ if $s \geq \delta_r$.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

It is now not hard to see that $\mathcal{R}$ is in fact an abelian group under the operation $(D_i, D_j) \to E = D(\delta_i + \delta_j)$. E is obtained by first computing the giant step $D_i \oplus D_j$ and $d = \delta_i + \delta_j - \delta(D_i \oplus D_j)$, and subsequently applying at most d "adjustment" baby steps to $D_i \oplus D_j$ until E is reached; again, the relative distance $\delta(E) - \delta_i - \delta_j$ is obtained in the process. Note that this operation is again slower than the group operation on the Jacobian of an imaginary hyperelliptic curve, due to the extra adjustment baby steps required. It is in essence identical to the arithmetic in the Jacobian of a real hyperelliptic curve presented in [35]. Using a technique akin to scalar multiplication, one can now use this operation to compute for any $s \in [0, R-1]$ the divisor $D(s)$ below s. Again, the distance shortfall $\delta\big(D(s)\big) - s$ is obtained when computing $D(s)$, so this process finds $\delta\big(D(s)\big)$ from s.

The original version of Diffie-Hellman key agreement using real hyperelliptic curves given in [39], and its subsequent improvement given in [37], were based on the first infrastructure description provided above. Here, Alice and Bob agree on a finite field $\mathbb{F}_q$ and a real hyperelliptic curve C over $\mathbb{F}_q$ with regulator R. They each generate respective secret values $a, b \in [g+1, R-1]$. Alice computes the infrastructure divisor $D(a)$ below a which also yields its distance $\delta_a = \delta\big(D(a)\big)$. She sends $D(a)$ (in Mumford representation) to Bob, keeps δ_a secret, and discards a. Similarly, Bob computes $D(b)$, thereby obtaining its distance δ_b, sends the Mumford coefficients of $D(b)$ to Alice, keeps δ_b secret, and discards b. From δ_a and $D(b) = D(\delta_b)$, Alice now computes the divisor $K = D(\delta_a \delta_b)$, while Bob uses his information δ_b and $D(a) = D(\delta_a)$ to also compute K.

To ensure the mathematical security of this protocol, the following two problems should be computationally intractable:

- Infrastructure Diffie-Hellman Problem (DHP): given $D(a)$ and $D(b)$, find the divisor in $\mathcal{R}$ below $\delta\big(D(a)\big)\delta\big(D(b)\big)$.
- Infrastructure Discrete Logarithm Problem (DLP): given $D \in \mathcal{R}$, find $\delta(D)$.

It is easy to see that the infrastructure DLP can be used to solve any instance of the infrastructure DHP: given $D(a)$ and $D(b)$, find their respective distances by solving the appropriate instances of the infrastructure DLP, and subsequently obtain the divisor in $\mathcal{R}$ below $\delta\big(D(a)\big)\delta\big(D(b)\big)$.

For clarity, we emphasize here that it is

- computationally easy to find for any given value $\delta \in [0, R-1]$ the divisor $D \in \mathcal{R}$ of distance δ (or at least of distance as close to δ as possible);
- computationally intractable to find for any divisor in $\mathcal{R}$ its distance.

In other words, it is easy to derive divisors from distances, but hard to derive distances from divisors; the latter task is exactly the infrastructure DLP. Solving the infrastructure DLP for a divisor $D \in \mathcal{R}$ is tantamount to finding a generator

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

of the principal $\mathbb{F}_q[C]$-ideal corresponding to D_S. This latter task represents an instance of the *principal ideal problem* which is believed to be computationally difficult.

Not surprisingly, this version of Diffie-Hellman, as well as the other infrastructure based protocols using real models described in [37], proved to be considerably less efficient and much more technically involved than their corresponding imaginary hyperelliptic curve counterparts. Nevertheless, it is possible to perform public key cryptography in the infrastructure, and recent improvements to these cryptographic schemes given in [24] actually make them competitive with their imaginary hyperelliptic curve counterparts. These improvements will be described in Sections 5 and 6 below.

5. Scalar multiplication

In any real or imaginary hyperelliptic curve based cryptographic protocol, some version of scalar multiplication features as the central ingredient. As indicated earlier, a technique much like scalar multiplication is employed when computing the infrastructure divisor $D(s)$ below any value $s \in [0, R-1]$; details of this method are provided in this section. It is therefore desirable to optimize scalar multiplication as much as possible. This can be done at three levels:

- Optimization of the scalar multiplication algorithm itself.
- Optimization of the arithmetic in the underlying structure (Jacobian or infrastructure); this means optimizing arithmetic on Mumford representations of reduced divisors in the imaginary setting and infrastructure divisors in the real setting.
- Optimization of the arithmetic in the underlying base field $\mathbb{F}_q$.

We discuss the first of these optimization levels in this section, and illustrate how to take advantage of these improvements in the context of real hyperelliptic curve based cryptography in the next section. Optimization of divisor arithmetic will be discussed in Section 7. There is an extensive body of literature on efficient arithmetic in finite fields. As improvements at this level are equally applicable to both the imaginary and real setting, we will ignore this third optimization level here and simply refer the reader to [21], [7].

Most of the material in this section can be found in Section 3 of [24]; here, we only provide an overview and refer the reader to the given source for details and proofs. As before, let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$ as given in (2.1). In order to analyze the performance of scalar multiplication in terms of the number of Jacobian or infrastructure operations, we distinguish between the following basic operations in the underlying structure:

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

- *Doubles*: operations of the form $E \oplus E$ for some arbitrary divisor E.
- *Adds*: operations of the form $E \oplus D$ where D is a fixed known divisor and E is an arbitrary divisor.
- *Baby steps*: baby steps in the infrastructure (real model), or adds $E \oplus D$ where D is a *special* divisor, i.e., a divisor of the form $D = p - \infty$ with p an $\mathbb{F}_q$-rational point on C (imaginary model).

As pointed out by Galbraith et al. [15], the baby step described above for imaginary models is the closest analogue to an infrastructure baby step and can also be computed in $O(g)$ field operations.

All known discrete logarithm based cryptographic schemes employ two types of scalar multiplication scenarios:

- *Fixed base*: both parties perform scalar multiplication on the same fixed base divisor, but using different scalars;
- *Variable base*: both parties perform scalar multiplication on different base divisors.

For example, the fixed and variable base scenarios are used in the first and second round of Diffie-Hellman key agreement, respectively. The digital signature algorithm uses the fixed base scenario for signature generation and both the fixed and variable base scenarios in its signature verification procedure. There is a third scenario that considers both fixed bases and fixed scalars, and can be optimized by precomputing addition chains for the scalars. Moreover, the fixed base scenario can also be further optimized using, for example, windowing methods by precomputing certain small powers of the fixed base divisor. However, these topics go beyond the scope of this paper; here, we focus only on the basic fixed and variable base scenarios listed above.

Recall that computing inverse divisors in both the Jacobian and the infrastructure is essentially free. In any structure where this is the case, it is desirable to perform scalar multiplication using the *non-adjacent form* (NAF) of the scalar $n \in \mathbb{N}$. This is the unique representation

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

where $b_0 = 1$, $b_i \in \{-1, 0, 1\}$ for $1 \leq i \leq l$, and no two consecutive digits b_i are non-zero. The NAF of n can easily be computed using, for example, [21, Algorithm 3.30, p. 98]. The length $l + 1$ of the NAF of n is at most one more than the length of the standard binary representation of n. Moreover, in a randomly generated integer $n \in \mathbb{N}$, one third of the NAF digits are expected to be non-zero, so this representation is sparser than the binary representation where half the bits are expected to be one; see [21, p. 98].

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

In the setting of imaginary hyperelliptic curves, both the fixed and the variable base scenario require the computation of the reduced divisor in the class of nD, given a reduced divisor D and an integer n. If precomputations are not used, optimization of the fixed base scenario happens only at the level of Jacobian arithmetic, i.e., on the group operation involving the base divisor; for example, the base could be a special divisor. Scalar multiplication in the Jacobian is done as follows:

```
SCALAR-MULT$(D, n)$    (*Scalar multiplication, imaginary model*)
*Input:* A reduced divisor $D$ and a scalar $n = \sum_{i=0}^{l} b_i 2^{l-i} \in \mathbb{N}$ given in NAF.
*Output:* The reduced divisor in the class of $nD$.
*Algorithm:*
  (1) set $E = D$;
  (2) for $i = 1$ to $l$ do
      (a) replace $E$ by $E \oplus E$;
      (b) if $b_i = 1$, then replace $E$ by $E \oplus D$;
      (c) if $b_i = -1$, then replace $E$ by $E \oplus \overline{D}$;
  (3) output $E$.
```

For a real hyperelliptic curve C of genus g, the variable base scenario corresponds to the *variable distance* scenario that reads as follows: given an infrastructure divisor $D \in \mathcal{R}$ of unknown distance $\delta(D)$ and a "scalar" n, find the divisor in $\mathcal{R}$ below $n\delta(D)$. It involves the same steps as algorithm SCALAR-MULT above, except after each double in step 2 (a), adjustment baby steps have to be performed to find the infrastructure divisor below $2\delta(E)$. Similar adjustment baby steps are needed after steps 2 (b) and (c) in each loop iteration; see algorithm VAR-DIST1 on [24, p. 205] for details. These adjustment baby steps decrease performance in the real model as compared to the imaginary model.

The fixed base scenario corresponds to the *fixed distance* scenario which requires finding the divisor $D(n) \in \mathcal{R}$ below a given scalar n. Here, one applies the variable distance algorithm to the fixed base divisor $D_2 \in \mathcal{R}$ of known distance $\delta_2 = g+1$ and the scalar $s = \lfloor n/(g+1) \rfloor$ given in NAF to compute $D\big(s(g+1)\big)$. Then one applies a further $n - s(g+1) \leq g$ baby steps to this divisor to obtain $D(n)$. So this procedure is also slower than its imaginary hyperelliptic curve counterpart.

In [24], significant improvements to the infrastructure setting over the versions given above were described that make the real model competitive for cryptography relative to its imaginary counterpart. The main idea here is to eliminate the repeated adjustment steps by performing one initial adjustment, along with other improvements that will be detailed below.

Henceforth, let C be a real hyperelliptic curve of genus g and infrastructure $\mathcal{R}$ over a finite field $\mathbb{F}_q$. Denote by D_+ the divisor obtained by applying a baby step

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

to the divisor $D \in \mathcal{R}$. Similarly, D_- is the divisor obtained by applying a baby step backwards from D, i.e., $(D_-)_+ = (D_+)_- = D$. Backward baby steps are just as fast as forward baby steps and use equally simple formulas using the Mumford representation of D (see, for example, [12]).

The key to our improvements is the following heuristics on the average behavior of the distance function:

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

- $\delta(D_+) - \delta(D) = 1$ for all $D \in \mathcal{R} \setminus \{0\}$.
- The quantity d in (4.1) is always equal to $\lceil g/2 \rceil$. That is, for all $D, D' \in \mathcal{R} \setminus \{0\}$, we have $\delta(D \oplus D') = \delta(D) + \delta(D') - \lceil g/2 \rceil$.

These heuristics make it possible to predict with high probability relative distances between infrastructure divisors, thereby eliminating the need to keep track of these relative distances during our computations. The heuristics are supported by plausible theoretical considerations as well as overwhelming numerical evidence, especially for large q; see [24] for a justification. The first statement above is equivalent to the assertion that $\deg(Q) = g$ with probability $1 - O(q^{-1})$ for every non-trivial divisor $D = (Q, P) \in \mathcal{R}$; divisors violating this assertion are usually referred to as *degenerate* and include the special divisors.

We henceforth assume the truth of (H) for all our algorithms. For the variable distance scenario, all previously required adjustment baby steps are eliminated and replaced by one initial adjustment of $d = \lceil g/2 \rceil$ baby steps applied to the input divisor at the beginning. This results in the following algorithm VAR-DIST (referred to as VAR-DIST2 in [24]):

```
VAR-DIST($D, n$)    (Variable distance scenario, real model)
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 $n\delta(D) + \lceil g/2 \rceil$.
Algorithm:
  (1) for $i = 1$ to $\lceil g/2 \rceil - 1$ do
        (a) replace $D$ by $D_+$;
  (2) set $D' = D,\ D'' = D_+,\ E = D_+$;
  (3) for $i = 1$ to $l$ do
        (a) replace $E$ by $E \oplus E$;
        (b) if $b_i = 1$, then replace $E$ by $E \oplus D''$;
        (c) if $b_i = -1$ and $g$ is even, then replace $E$ by $E \oplus \overline{D''}$;
        (d) if $b_i = -1$ and $g$ is odd, then replace $E$ by $E \oplus \overline{D'}$;
  (4) output $E$.
```

Note that the computational effort of this procedure is identical to that of SCALAR-MULT except for the extra $\lceil g/2 \rceil$ baby steps at the beginning.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

The idea for the fixed distance scenario is to compute the divisor in $\mathcal{R}$ whose distance is some known and easily computable function of the input scalar n. Once again, all adjustment steps are eliminated. Moreover, all the adds in steps 2 (b) and (c) of algorithm SCALAR-MULT are replaced by much faster baby steps. The algorithm uses the base divisor $D_{d+3} \in \mathcal{R}$ of distance $\delta_{d+3} = g+d+2$, with $d = \lceil g/2 \rceil$, which needs to be precomputed by applying $d+2$ baby steps to the trivial divisor $D_1 \in \mathcal{R}$. It is stated as algorithm FIXED-DIST below (algorithm FIXED-DIST2 in [24]):

```
FIXED-DIST($n$)    (Fixed distance scenario, real model)
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 $2^l(g+1) + n + d$, with $d = \lceil g/2 \rceil$.
Precomputed: The divisor $D_{d+3}$ of distance $\delta_{d+3} = g + d + 2$.
Algorithm:
  (1) set $E = D_{d+3}$;
  (2) for $i = 1$ to $l$ do
        (a) replace $E$ by $E \oplus E$;
        (b) if $b_i = 1$, then replace $E$ by $E_+$;
        (c) if $b_i = -1$, then replace $E$ by $E_-$;
  (3) output $E$.
```

Proofs of correctness of algorithms VAR-DIST and FIXED-DIST, assuming heuristics (H), can be found in [24, Propositions 3.1 and 3.2], respectively. In Table 1, we compare the expected computational effort, i.e., the number of doubles, adds and baby steps, of all three algorithms presented in this section. For the imaginary setting, we distinguish between scalar multiplication using an arbitrary or a special base divisor; recall that additions involving a special divisor were referred to as baby steps in this context. For the real setting, we assume Heuristics (H). In algorithm FIXED-DIST, we ignore the precomputation of D_{d+3} as this need only be done once; this divisor could be included in the domain parameters of any cryptographic scheme using this procedure.

TABLE 1. Operation counts for scalar multiplication.

Model	Algorithm	Doubles	Adds	Baby Steps
Imaginary	SCALAR-MULT, arbitrary base	l	$l/3$	0
Imaginary	SCALAR-MULT, special base	l	0	$l/3$
Real	VAR-DIST	l	$l/3$	$\lceil g/2 \rceil$
Real	FIXED-DIST	l	0	$l/3$

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

We see that algorithm VAR-DIST requires only $\lceil g/2 \rceil$ more baby steps than SCALAR-MULT involving an arbitrary base divisor; for the cases of interest $g = 2, 3$, this means only one or two extra baby steps, respectively. FIXED-DIST has the same number of operations as SCALAR-MULT involving a special base divisor. Scalar multiplication timings for various genera and base field sizes from [24] show that when using generic divisor arithmetic and an arbitrary base divisor in the imaginary case, the practical performance matches the predictions of Table 1 very well.

6. Infrastructure based cryptographic protocols

The two most well-known hyperelliptic curve based cryptographic protocols are the already mentioned Diffie-Hellman key establishment procedure and the Digital Signature Algorithm (DSA). Here, we will limit our discussion to these two schemes, but mention that other elliptic curve based schemes as described in [21, Sections 4.4–4.6, pp. 183–196] can be similarly adapted to work in the Jacobian of a hyperelliptic curve. Any such protocol can then be converted to a corresponding infrastructure based scheme by replacing scalar multiplications involving an arbitrary divisor by calls to VAR-DIST, and scalar multiplications involving a fixed special divisor (baby steps) by calls to FIXED-DIST.

Much of this discussion can be found in [24, Section 4]. The domain parameters for infrastructure based Diffie-Hellman include a prime power q, a real hyperelliptic curve C over $\mathbb{F}_q$ of genus g as described in (2.1), the regulator R of C, and the divisor D_{d+3} of distance $g + d + 2$. Here, and throughout this section, we put $d = \lceil g/2 \rceil$ and assume that heuristics (H) apply to C. Note that in the imaginary setting, D_{d+3} is replaced by a (possibly special) reduced base divisor D, and R by the order of the cyclic group generated by D, so the domain parameters are very similar in both settings. The protocol executes as follows:

Infrastructure based Diffie-Hellman key agreement
Domain parameters: q, C, R, D_{d+3}.
Round 1:

(1) Alice generates random $a \in [g + 1, R - 1]$ of NAF length $l + 1$, computes $D_A = \text{FIXED-DIST}(a)$ and sends D_A to Bob;

(2) Bob generates random $b \in [g + 1, R - 1]$ of NAF length $l + 1$, computes $D_B = \text{FIXED-DIST}(b)$, and sends D_B to Alice.

Round 2:

(1) Alice computes $K = \text{VAR-DIST}\big(D_B, 2^l(g + 1) + a + d \pmod{R}\big)$;

(2) Bob computes $K = \text{VAR-DIST}\big(D_A, 2^l(g + 1) + b + d \pmod{R}\big)$.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Assuming heuristics (H), Alice and Bob arrive at a common divisor in $K \in \mathcal{R}$ of distance

$$\delta(K) \equiv \big(2^l(g+1)+a+d\big)\big(2^l(g+1)+b+d\big)+d \pmod{R}.$$

There is a modified version of the above protocol that can be used if R is not known, but has worse running time; details can again be found in [24]. The data in [24] (see in particular Table 6) shows that the new version of key exchange in the infrastructure offers a significant improvement over the previous version, and when using generic divisor arithmetic is slightly faster than key exchange in the imaginary case.

The domain parameters for real hyperelliptic curve based DSA are the same as those for infrastructure based Diffie-Hellman. In addition, all participants agree on a public hash function H that maps messages to integers in $[1, R-1]$ and on a public function that maps infrastructure divisors to integers, for example, by converting the first Mumford coefficient Q to an integer.

A user's private key is an integer $n \in [1, R-1]$ of NAF length $l+1$. The corresponding public key is the divisor $D = D(n) \in \mathcal{R}$ of distance n which is obtained as follows. First, generate the divisor of distance $2^l(g+1)+d$ by applying l successive doubles, starting with the divisor D_{d+2} of distance $g+d+1$; D_{d+2} can be found by applying a backward baby step to D_{d+3}. Next, apply $g-d$ baby steps, starting with the divisor $D\big(2^l(g+1)+d\big)$ thus obtained, to generate the divisor $D^* = D\big(2^l(g+1)+g\big)$ of distance $2^l(g+1)+g$. Finally, compute

$$D(n) = \text{FIXED-DIST}(n) \oplus \overline{D^*}.$$

The total cost of generating the public key is $2l$ doubles, one add, and $l/3+g-d+1$ baby steps. DSA on C is performed as follows:

Infrastructure based digital signature algorithm
Domain parameters: q, C, R, D_{d+3}.
Private key: $n \in [1, R-1]$.
Public key: $D = D(n)$.
Signature Generation: To sign a message m, the signer:

(1) generates random $k \in [1, R-1]$ with $\gcd(k, R) = 1$;

(2) computes $D_k = \text{FIXED-DIST}(k)$ and converts D_k to an integer N;

(3) sets $r \equiv N \pmod{R}$; if $r = 0$, returns to step 1;

(4) computes $s \equiv k^{-1}\big(H(m)+nr\big) \pmod{R}$; if $s = 0$, returns to step 1;

(5) signs m with signature (r, s).

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

(1) checks that $1 \le r, s < R$; if not, rejects the signature;

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

(2) computes $w \equiv s^{-1} \pmod{R}$, $u_1 \equiv H(m)w \pmod{R}$, $u_2 \equiv rw \pmod{R}$;

(3) obtains the signer's public key D;

(4) computes $E_1 = \text{FIXED-DIST}(u_1)$ and $E_2 = \text{VAR-DIST}(D, u_2)$;

(5) computes $E_3 = E_1 \oplus E_2$; if $E_3 = \mathbf{0}$, rejects the signature, else converts E_3 to an integer N;

(6) accepts the signature if $r \equiv N \pmod{R}$ and rejects it otherwise.

If the signature is valid, both D_k and E_3 can be shown to have distance $2^l(g+1) + k + d \pmod{R}$, assuming heuristics (H). Therefore, $E_3 = D_k$, and thus $r \equiv N \pmod{R}$.

The other elliptic curve based protocols described in [21, Sections 4.4–4.6, pp. 183–196] can similarly be adapted to the infrastructure setting. Operation counts for all these schemes (not considering special base divisors in the imaginary setting) can be found in [24, Table 2].

7. Divisor arithmetic

In order to optimize hyperelliptic curve protocols further, it is necessary to make the basic building blocks of scalar multiplication, namely doubles, adds and baby steps, as fast as possible. We first discuss generic algorithms that apply to curves of any genus, and then consider explicit formulas for genus 2 curves which represent the most interesting case for cryptography.

7.1. Cantor's algorithm and NUCOMP

As mentioned in Section 3, the two main generic giant step algorithms are due to Cantor [4] and the NUCOMP method that was first presented in [27] and later discussed in detail in [25]. Given a hyperelliptic curve C of genus g, Cantor's algorithm essentially first adds two reduced divisors on C, obtaining a semi-reduced divisor with Mumford coefficients of degree as large as $2g$, and then reduces the result. NUCOMP improves Cantor's method by performing the reduction step during the addition step. The reduction is in essence performed on intermediate operands occurring during the addition, keeping their degrees well below $2g$. Both algorithms have an asymptotic run time that is quadratic in the genus, but NUCOMP performs significantly faster than Cantor's method for moderate sized genus, and improves as the genus increases.

Tables 2 and 3, taken from [25], contain numerical data comparing basic scalar multiplication using NUCOMP and Cantor's algorithm in real hyperelliptic curves over prime fields and characteristic 2 fields, respectively. These computations were performed on a Pentium IV 2.4 GHz computer running Linux,

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

using the computer algebra library NTL [41] for finite field and polynomial arithmetic and the GNU C++ compiler version 3.4.3. The data in the table is the ratio of the total time using NUCOMP to perform 1000 scalar multiplications using 100-bit multipliers over the time to perform the same operations using Cantor's algorithm. This data shows that NUCOMP does indeed outperform Cantor's algorithm for relatively small genus, and improves as the genus increases.

TABLE 2. Scalar multiplication ratios (NUCOMP / Cantor) over $\mathbb{F}_p$, real.

	$\log_2 p$								
g	2	4	8	16	32	64	128	256	512
2	0.8943	1.1268	1.2192	1.2763	1.0659	1.0987	1.0872	1.0731	1.0835
3	1.0449	1.1497	1.1165	1.1330	1.0503	1.0515	1.0434	1.0376	1.0500
4	1.0745	1.1081	1.0932	1.0784	1.0169	1.0137	1.0150	0.9847	1.0060
5	1.0549	1.0659	1.0300	1.0570	0.9635	0.9771	0.9650	0.9664	0.9787
6	1.0507	1.0124	1.0350	1.0327	0.9444	0.9555	0.9243	0.9540	0.9569
7	0.9705	0.9525	0.9231	0.9209	0.9144	0.9309	0.8950	0.9289	0.9512
8	0.9724	0.9539	0.9426	0.9338	0.9094	0.9195	0.8816	0.9244	0.9254
9	0.9591	0.9179	0.9028	0.9023	0.8726	0.8876	0.8608	0.8913	0.9013
10	0.9105	0.9056	0.8818	0.8877	0.8625	0.8879	0.8642	0.8933	0.8955
11	0.9396	0.9043	0.9159	0.9145	0.8402	0.8596	0.8415	0.8836	0.8862
12	0.9668	0.9341	0.9149	0.9135	0.8356	0.8536	0.8512	0.8832	0.8745
13	0.9581	0.9128	0.8856	0.8942	0.8047	0.7877	0.8201	0.8637	0.8560
14	0.9596	0.9098	0.8782	0.8912	0.8051	0.7874	0.8205	0.8502	0.8471
15	0.9356	0.8696	0.8640	0.8670	0.7789	0.7656	0.8037	0.8425	0.8387
20	0.8065	0.7549	0.7519	0.7638	0.7463	0.7275	0.8110	0.8108	0.8008
25	0.7717	0.7303	0.7215	0.7186	0.6996	0.7124	0.7818	0.7842	0.7723
30	0.7651	0.7337	0.7270	0.7392	0.6873	0.7212	0.7687	0.7749	0.7801

7.2. Explicit formulas

Although the basic formulas for divisor arithmetic, for example Cantor's algorithm, are described in terms of polynomials, the most efficient formulations are achieved using so-called explicit formulas. By breaking down the polynomial operations in terms of sequences of finite field operations, a number of simplifications and improvements can be obtained. A great deal of work has gone into developing efficient explicit formulas for low genus imaginary curves [7]. Formulas exist for genus 2, 3, and 4 using an affine coordinate representation. For genus 2 and 3, inversion-free formulas using projective coordinates also exist. Explicit formulas for real hyperelliptic curves are not yet as well-developed as in the imaginary case. Complete explicit formulas for genus 2 curves in affine

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

TABLE 3. Scalar multiplication ratios (NUCOMP / Cantor) over $\mathbb{F}_{2^n}$, real.

	$\log_2 p$								
g	2	4	8	16	32	64	128	256	512
2	0.9277	1.0896	1.0930	1.0854	1.1152	1.0998	1.0653	1.0787	1.0661
3	0.8948	0.9597	0.9860	0.9622	0.9695	0.9791	0.9640	0.9824	0.9814
4	1.0213	1.0360	1.0375	1.0274	1.0267	1.0252	1.0289	1.0421	1.0440
5	0.9587	0.9672	0.9630	0.9505	0.9372	0.9295	0.9499	0.9516	0.9491
6	0.9989	0.9776	0.9743	0.9838	0.9718	0.9715	0.9689	0.9777	0.9790
7	0.9370	0.9193	0.9025	0.9126	0.8990	0.9041	0.9311	0.9413	0.9424
8	0.9534	0.9439	0.9222	0.9379	0.9365	0.9476	0.9650	0.9785	0.9842
9	0.9008	0.8771	0.8685	0.8928	0.8707	0.8727	0.8891	0.8954	0.8963
10	0.9053	0.8863	0.8854	0.9142	0.9098	0.9115	0.9252	0.9384	0.9491
11	0.8601	0.8518	0.8504	0.8713	0.8624	0.8595	0.8755	0.8838	0.8870
12	0.8878	0.8679	0.8589	0.8705	0.8938	0.9006	0.9154	0.9220	0.9301
13	0.8377	0.8230	0.8171	0.8281	0.8478	0.8476	0.8675	0.8729	0.8770
14	0.8393	0.8258	0.8206	0.8384	0.8659	0.8783	0.8863	0.8957	0.9031
15	0.7970	0.7775	0.7800	0.7942	0.8204	0.8423	0.8548	0.8594	0.8659
20	0.7221	0.7313	0.7312	0.7552	0.7968	0.8291	0.8311	0.8548	0.8638
25	0.6565	0.6298	0.6678	0.6954	0.7907	0.7356	0.7520	0.7756	0.8010
30	0.6576	0.6649	0.6722	0.6936	0.7353	0.7663	0.7823	0.8043	0.8187

representation have only recently been found [11], [12]. Inversion-free formulas for real genus 2 curves are presented in [10].

Table 4 lists the most recent operation counts for explicit formulas in genus 2 imaginary and real hyperelliptic curves assuming an affine model. In the imaginary case, the formulas are due to Lange [30]. In the table, I denotes inversions, S denotes squarings, and M denotes multiplications in the base field; as usual, field additions and subtractions are not counted. The operation count for the addition operation includes the reduction of one field squaring described in [11]. Operation counts for the real case are taken from [11], [12].

TABLE 4. Operation counts for explicit formulas.

Model	Double	Add	Baby Step
Imaginary	1I, 5S, 22M	1I, 3S, 22M	1I, 1S, 10M
Real (q odd)	1I, 4S, 28M	1I, 2S, 26M	1I, 2S, 4M
Real (q even)	1I, 2S, 29M	1I, 1S, 27M	1I, 1S, 5M

In Tables 5 and 6, we list running times, taken from [12], for executing fixed base scalar multiplication (Algorithm SCALAR-MULT with a special divisor

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

for the imaginary case and FIXED-DIST for the real case), variable base scalar multiplication (Algorithm SCALAR-MULT for the imaginary case and the version of VAR-DIST from [24] without the regulator R for the real case), and one iteration of Diffie-Hellman key exchange (i.e., one fixed base and one variable base scalar multiplication) on genus 2 hyperelliptic curves using explicit formulas. Times for the imaginary case and real case (using the infrastructure in the latter) are both included. The finite fields used were of size 2^b where b denotes the specified security level in bits, resulting in the Jacobian and infrastructure each having expected size 2^{2b}. The scalar multipliers used had twice the number of bits as the security level. The programs were run on an Intel Core Duo 2.66 GHz CPU running Linux, using g++ 4.1.2 and NTL [41] for finite field and polynomial arithmetic. The runtimes are given in milliseconds and represent the average taken over 5000 iterations of each algorithm.

TABLE 5. Key exchange timings, q odd.

Security	Imaginary			Real		
(in bits)	Fixed	Var	DH Total	Fixed	Var	DH Total
80	2.137	2.304	4.440	2.307	2.618	4.925
112	3.545	3.942	7.487	3.809	4.469	8.278
128	4.702	5.149	9.851	5.003	5.869	10.872
192	10.526	11.562	22.088	11.192	13.048	24.240
256	15.560	17.077	32.636	16.492	19.168	35.660

TABLE 6. Key exchange timings, q even.

Security	Imaginary			Real		
(in bits)	Fixed	Var	DH Total	Fixed	Var	DH Total
80	4.721	5.331	10.052	5.112	6.139	11.250
112	4.096	4.475	8.571	4.425	5.076	9.500
128	4.814	5.304	10.118	5.138	5.920	11.057
192	11.700	12.942	24.641	12.715	14.721	27.436
256	22.255	24.572	46.827	24.525	28.326	52.851

Note that when explicit formulas are used, the performance of the real model in genus 2 is very close to that of the imaginary model. In our experiments, the real model was always slower, but by at most 6 milliseconds for one application of Diffie-Hellman key exchange. Using the version of VAR-DIST that makes use of the regulator would improve the timings for the real case. In addition, shaving

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

one or two field operations from the explicit formulas in the real case would almost certainly result in the real model being more efficient.

8. Security

As mentioned in Section 4, the security of cryptographic protocols based on the infrastructure of a real hyperelliptic curve is closely related to the infrastructure discrete logarithm problem, namely computing the distance $\delta(D)$ of a given infrastructure divisor D. In every infrastructure-related protocol, this unknown distance plays the role of a private key or allows one to compute the Diffie-Hellman secret, so an adversary who recovers it completely compromises the protocol.

Like the usual discrete logarithm problem in a cyclic group, the infrastructure discrete logarithm problem can be solved generically in square-root time. Using the infrastructure baby step and giant step operations yields a straightforward adaptation of the well-known baby step giant step algorithm. As the distance around the entire infrastructure is equal to the regulator R, this algorithm has complexity $O(\sqrt{R})$ infrastructure operations. We generally expect the regulator to be of size roughly q^g, so in the typical case the complexity is $O(q^{g/2+\epsilon})$ field operations, the same as the discrete logarithm problem in the Jacobian. If the regulator is known, as is usually assumed for cryptographic applications, the Pollard rho algorithm can also be used, yielding an algorithm with expected complexity $O(q^{g/2+\epsilon})$ field operations but with constant storage requirement.

The index-calculus method, which has been used successfully for solving the discrete logarithm problem in the Jacobian of an imaginary hyperelliptic curve, has also been adapted to the infrastructure discrete logarithm problem. As in the imaginary case, this method is most effective when the genus is large compared to $\log q$, yielding subexponential complexity in that case. The first description of an index-calculus algorithm for the infrastructure discrete logarithm problem, due to Müller, Stein, and Thiel, was presented in [34]. Their algorithm has complexity

$$O\big(L_{q^{2g+2}}[1.44 + o(1)]\big) \qquad \text{if} \quad g > \log q,$$

where

$$L_N[\beta] = \exp\Big(\beta\sqrt{\log N \log\log N}\,\Big),$$

indicating that high genus hyperelliptic curves should not in general be used for cryptographic purposes.

The authors of [34] did not provide an implementation of their algorithm. The first implementation of any index-calculus algorithm for real hyperelliptic curves was provided recently by Hammell [19], [20]. In these sources, algorithms

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

for solving the infrastructure discrete logarithm problem, as well as computing the regulator and the structure of the ideal class group of the corresponding function field, are presented. In addition to the numerical results, asymptotic improvements to the algorithm of [34] are described. By taking advantage of the fast baby step operation in the relation generation process, and using newer linear algebra algorithms with better complexity than those considered in [34], a complexity of $O\big(L_{q^g}[2.45 + o(1)]\big)$ is obtained, assuming that $g > \log q$ and that smooth reduced ideals are distributed evenly amongst equivalence classes. Note that, although the constant 2.45 in the new complexity statement is larger than that from [34], the subexponential function is expressed in terms of q^g as opposed to q^{2g+2}, so the new result is in fact an improvement.

Tables 7 and 8 contain some sample runtimes for computing the regulator R, extracted from [20]. Computing R is a special case of the infrastructure discrete logarithm problem, and gives a good indication of the difficulty of that problem. A simple version of the baby step giant step algorithm, the improved algorithm using baby steps for relation generation, and another version using sieving (adapted from Velichka's work in the imaginary case [48], [26]) for relation generation were all implemented and compared. These computations were performed on a Pentium 4 CPU running at 3.0 GHz with 1 GB of RAM. The software was written in C++, the NTL library [41] was used for finite field and polynomial arithmetic, and IML [6] was used for linear algebra. Note that, as expected, both versions of the index-calculus algorithm out-perform baby step giant step for larger genus. Sieving appears to only be worthwhile in even characteristic, but as the routines from [48], [26] were used, and these were only

Table 7. Runtimes for computing R (in hh:mm:ss), q even.

q	g	$\log R$	BSGS	Baby Walk	Sieving
2^2	5	9	:00	:00	:00
2^2	10	21	:00	:01	:00
2^2	15	31	:06	:02	:02
2^2	20	39	2:27	:29	:25
2^2	25	50	35:47:10	2:59	1:39
2^2	30	61	—	17:56	6:34
2^2	35	71	—	1:24:53	32:35
2^2	40	80	—	14:29:13	7:11:32
2^6	5	31	:03	:02	:13
2^6	10	61	—	3:39	2:24
2^6	15	91	—	18:39:49	17:11:07

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

TABLE 8. Runtimes for computing R (in hh:mm:ss), q odd.

q	g	$\log R$	BSGS	Baby Walk	Sieving
5	5	8	:00	:00	:00
5	10	17	:00	:01	:00
5	15	20	:00	:03	:03
5	20	33	:04	:01	:09
5	25	41	2:41	:07	:27
5	30	51	18:02:51	:31	1:33
5	35	58	—	4:22	12:47
5	40	64	—	22:00	1:43:35
67	5	30	:01	:03	:10
67	10	60	—	2:14	17:07
67	15	85	—	13:31:10	167:34:27

optimized for even characteristic, more work on the odd characteristic case may yield improvements.

To date, these computations represent the only results obtained using index--calculus in the infrastructure of a real hyperelliptic curve. In particular, there has not been any work applying these techniques, or recent methods designed for use in the imaginary case such as [18], to low genus real hyperelliptic curves.

8.1. Recent results

Until recently, it was an open question whether the smoothness of the regulator R has any bearing on the complexity of the infrastructure discrete logarithm problem. It is well-known that the complexity of the discrete logarithm problem in a cyclic group depends on the size of the largest prime divisor of the order due to the Pohlig-Hellman algorithm [36]. However, as the infrastructure is not a group, it is not immediately obvious that similar techniques would work in this setting.

Fontein [13], [14] showed that the Pohlig-Hellman algorithm can indeed be adapted to solve the infrastructure discrete logarithm problem, and that, consequently, only hyperelliptic curves whose regulator has a large prime divisor should be used for cryptographic purposes. In particular, Fontein described a concept called *f-representations*, an explicit method of embedding the elements of the infrastructure into a cyclic group of order R that preserves distances. Thus, any generic algorithm for solving the discrete logarithm problem in a cyclic group, including Pohlig-Hellman, can be used to solve the infrastructure discrete logarithm by transforming it into a cyclic group of order R using f-representations.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

The question of whether Pohlig-Hellman applies to the infrastructure was independently answered by Mireles Morales [33]. As described in Section 4, the cyclic subgroup $\mathcal{G}$ of the Jacobian generated by the class of $\infty - \overline{\infty}$ has order R. Recall that an alternative description of the infrastructure, as used in [24], [25], is

$$\mathcal{R}' = \{D_S - \deg(D_S)\infty \mid D = D_S - \deg(D_S)\infty - \delta(D)(\overline{\infty} - \infty) \in \mathcal{R}\}.$$

Consider the class in $\mathcal{G}$ of any divisor of the form $\delta(D)(\infty - \overline{\infty})$, with $D \in \mathcal{R}$. Then the unique divisor representing this class in the Jacobian, in the sense of [35], is $D_S - \deg(D_S)\infty \in \mathcal{R}'$. This yields a one-to-one correspondence between divisors in $\mathcal{R}'$ (or in $\mathcal{R}$), and most elements of $\mathcal{G}$, namely those classes represented by a divisor of the form $n(\infty - \overline{\infty})$ where n is the distance of some infrastructure divisor in $\mathcal{R}$. In the case of genus 1, i.e., elliptic curves, this correspondence was studied by Stein [42] and it was shown that the infrastructure discrete logarithm problem and elliptic curve discrete logarithm problem are in fact equivalent. The result of [33] generalizes this result to arbitrary genus by describing an explicit embedding of the infrastructure into the subgroup $\mathcal{G}$ of the Jacobian. Thus, baby steps $D \to D_+$ in the infrastructure correspond to addition by $b(\infty - \overline{\infty})$ in the Jacobian, where $b = \delta(D_+) - \delta(D)$, and distances correspond to discrete logarithms with respect to the base divisor $\infty - \overline{\infty}$.

Although Mireles Morales' result is similar to Fontein's results applied to the specific case of real hyperelliptic curves, it has the advantage that the group into which the infrastructure is embedded is a subgroup of a well-studied algebraic geometric object (the Jacobian). Although the structure obtained using f-representations is essentially the same, it is likely not as amenable to computation. For example, arithmetic in the Jacobian can be performed efficiently using the balanced divisor representation of [15], whereas some more work would be required to describe efficient arithmetic using f-representations. Fontein's results, on the other hand, are more general in the sense that they also apply to multi-dimensional infrastructures, not just the one-dimensional ones that occur for example in the case of real hyperelliptic curves.

In addition to an independent proof that Pohlig-Hellman applies to the infrastructure discrete logarithm problem, Mireles Morales' result has the useful consequence of enabling problems and applications in the infrastructure to be interpreted in the (perhaps) more familiar setting of a subgroup of the Jacobian. For example, cryptographic key exchange in the infrastructure can be thought of as key exchange in the Jacobian using $\infty - \overline{\infty}$ as a fixed base divisor, and the infrastructure discrete logarithm problem is nothing more than the usual discrete logarithm problem in the subgroup generated by the class of $\infty - \overline{\infty}$. As a result, any algorithm for solving the discrete logarithm problem in the Jacobian of a hyperelliptic curve immediately applies to the infrastructure discrete logarithm problem with the same complexity. For example, the state-of-the-art

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

for solving the discrete logarithm problem on a low genus imaginary hyperelliptic curve, due to Gaudry et al. [18], is $O(q^{2-2/g+\epsilon})$. The fact that divisor arithmetic in the real case has the same asymptotic complexity as the imaginary case implies that this complexity holds for the real setting as well. Thus, thanks to the work of Fontein and Mireles Morales, the infrastructure discrete logarithm problem can also be solved with the same complexity, and furthermore, the same security considerations as in the imaginary case apply.

9. History and related work

As pointed out in Section 1, the first use of real hyperelliptic curves in cryptography dates back to 1996, when Scheidler, Stein, and Williams [39] introduced a real hyperelliptic curve based Diffie-Hellman type protocol. They formulated the infrastructure as the set of reduced principal ideals in the maximal order of a real quadratic function field, or equivalently, the coordinate ring of a real hyperelliptic curve. The notion of infrastructure was first introduced and so named by Shanks [40] in the context of real quadratic number fields, and the number field version of this concept had already undergone investigation for cryptographic applications in 1994 [38]. To our knowledge, the infrastructure, in real quadratic number fields or real hyperelliptic curves, is the only non-group structure that supports efficient and secure discrete logarithm based cryptography. Compared to its quadratic number field analogue, the infrastructure of a real hyperelliptic curve has simpler, cleaner arithmetic. It also appears to represent a cryptographically more secure setting if the curve has small genus, since the number field infrastructure discrete logarithm problem always has at most subexponential complexity [3].

A slight improvement to the original infrastructure key agreement protocol [39] as well as infrastructure versions of ElGamal's 1985 public key cryptosystem and signature scheme [9] were presented in [37]. As mentioned earlier, all these schemes proved to be considerably less efficient and much more technically involved than the corresponding protocols in the imaginary setting. While not explicitly stated in [39], [37], these protocols employed the infrastructure as a group, whose operation assigns two infrastructure divisors D, D' of respective distances δ, δ' the infrastructure divisor below $\delta+\delta'$. This operation is slower than Jacobian arithmetic due to extra adjustment baby steps required, as explained in Section 4. It was also subsequently recognized that this operation in essence amounts to a version of arithmetic via unique representatives in the Jacobian of a real hyperelliptic curve as described in [35].

Many of these technical difficulties were resolved in [24]. In addition, this source as well as [25] provided the first description of the infrastructure in terms

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

of divisors on a real hyperelliptic curve, rather than ideals in the coordinate ring. This divisor theoretic framework matches the traditional description of the Jacobian much more closely. As explained in Section 5, by assuming Heuristics (H), the authors of [24] were able to effect a number of modifications of previous infrastructure based protocols which led to significant improvements in efficiency. The key ideas were to eliminate all the extra adjustment baby steps, and replace some of the giant steps by much more efficient baby steps in the scalar multiplication routines. These ideas can be employed in all known discrete logarithm based protocols for key agreement, digital signatures, and encryption.

Operation counts as well as numerical data revealed that the new arithmetic in the infrastructure of the real model is comparable in efficiency to Jacobian arithmetic for imaginary models. These observations are also supported when comparing the explicit formulas for imaginary hyperelliptic curves of genus 2 given in [30] to those for real hyperelliptic curves of genus 2 in [11], [12]. A careful analysis reveals that the latter require only marginally more finite field multiplications than the former. However, the baby step operation in its explicit form is significantly more efficient than divisor addition in either setting, and as a result, the cryptographic protocols in the real setting perform almost as well as those in the imaginary case when a special base divisor is used. In addition, even though the formulas are not as fast as those in the imaginary case, they are certainly more efficient than using generic algorithms. Thus, using the formulas of [11], [12] will significantly speed up other computations in the divisor class group or infrastructure of a real hyperelliptic curve, for example, computing the regulator or the class number (see, e.g., [45], [44], [46]).

As already mentioned in Section 3, an interesting new model for Jacobian arithmetic on real hyperelliptic curves was recently introduced by Galbraith, Harrison, and Mireles Morales in [15]. Their balanced divisor representation eliminated almost all additional adjustment baby steps required in the representation from [35] in a similar manner to the infrastructure improvements of [24]. If the curve has odd genus, at most one adjustment step is required; for even genus none is needed, making arithmetic in the Jacobian of a real hyperelliptic curve almost as efficient as in the imaginary case. Although a similar representation was suggested by Cassels and Flynn [5] for genus 2 curves, this generalization to arbitrary genus is relatively new and has to date only been applied to an investigation of pairing computations in genus 2 [16]. Moreover, the balanced divisor situation still needs efficient algorithms for divisor addition and reduction. In particular, for hyperelliptic curves of genus 2, it will be necessary to use the explicit formulas of [11], [12].

Additional comments on the results of Fontein and Mireles Morales connecting the infrastructure to the Jacobian in the setting of real hyperelliptic curves are warranted; these results were already mentioned in Section 8. In [33],

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Mireles Morales showed how to embed the infrastructure into the subgroup of the Jacobian generated by $\infty - \overline{\infty}$. In terms of security, this result shows that the infrastructure discrete logarithm problem reduces to the discrete logarithm problem in a subgroup of the Jacobian. In terms of implementation, one obtains cryptographic protocols based on divisor arithmetic only, and the advantageous situation of replacing giant steps with baby steps for scalar multiplication can be realized by using $\infty - \overline{\infty}$ as the base. When combined with balanced divisor arithmetic in the Jacobian, this results in cryptographic protocols that are very similar to the improved infrastructure-based protocols of [24]. Although a thorough comparison has not yet been done, both settings will likely be comparable in efficiency.

Finally, the infrastructure setting was generalized completely by Fontein in [13], [14]. In addition to independently showing how to embed the infrastructure of a real hyperelliptic curve into a group, Fontein interprets the infrastructures of essentially any function field in a very broad context using Arakelov theory and very reasonable basic assumptions. The author presents the concepts of distance, baby step, giant step, and periodicity in global fields. He discusses the special case of one-dimensional infrastructures in detail and thus identifies the settings described in [39], [25], [16], [33], [35] as special cases. His observations on arithmetic lead to very general cryptographic protocols. The choice of the base divisor and the situation at infinity is somewhat variable, leaving plenty of room for improvement in any specific situation. It would be interesting to see whether such a broader point of view yields an interpretation of the infrastructure that leads to more efficient applications.

10. Conclusions

We saw that every imaginary hyperelliptic curve can be transformed into a real model, while the reverse process is frequently not possible over the same base field. Besides representing a far more general class of curves, there are further reasons why real models are of interest. Firstly, hyperelliptic curves for use in cryptography can be generated by special methods such as complex multiplication. Very often, the output of these methods is a real hyperelliptic curve; see, for example, the construction of pairing-friendly hyperelliptic curves in [17]. It is therefore desirable to have efficient arithmetic on these curves available. Secondly, real hyperelliptic curves have gained popularity due to their potential use in pairing-based cryptography [16]. Finally, note that Edwards models of elliptic curves [2] are real models. This suggests that the arithmetic on these curves warrants further investigation and that their hyperelliptic analogues should be investigated further.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

Cryptography based on real hyperelliptic curves has come a long way since it was initially proposed. Recent advances have shown that cryptography using the infrastructure of a real hyperelliptic curve is almost as efficient as using the Jacobian of an imaginary hyperelliptic curve. The idea of balanced divisors will likely have a similar effect on cryptosystems using the Jacobian in the real case. The results of Fontein and Mireles Morales show that real hyperelliptic curves, when using the infrastructure or the Jacobian, are as secure as their imaginary counterparts. Nevertheless, these advances are relatively recent and there is still work to do in the real case.

In terms of security, numerical work on the infrastructure discrete logarithm problem currently lags behind the discrete logarithm problem in the imaginary case. To date, the only work on the subject is the investigation of index calculus in high genus due to Hammell [19], [20], and as mentioned above, no computational work has as yet been done for small genus. It would be interesting to see how well the state-of-the-art algorithms in the low genus imaginary case, especially [18], work in the real case. In addition, methods successfully employed in the high genus case, including sieving and baby step based relation generation, should be tried in low genus (both real and imaginary). More work on sieving itself, especially in odd characteristic, may help improve these algorithms further. Finally, the connection between the imaginary hyperelliptic curve discrete logarithm problem and the real model, both using the Jacobian and infrastructure, should be explored for $g > 1$. If computing discrete logarithms in one of these models turns out to be faster, for example by exploiting baby steps in the real case, such relationships would allow the faster model to be used in both cases.

There is also more that can be done to improve the efficiency of cryptographic protocols using the real model. In particular, implementations for genus 2 imaginary curves make use of explicit formulas such as those given in [30]. Analogous explicit formulas have now been derived for real hyperelliptic curves of genus 2 [11]. These formulas yield slightly less efficient performance than their imaginary counterparts, but not nearly as much work has gone into them; research on this subject has only recently begun and is ongoing. It is possible that further investigation, possibly using alternative algorithms such as NUCOMP, will yield improvements, and savings of only a few operations should make the protocols in the real case as fast as or faster than their imaginary counterparts. Explicit formulas for genus 3 real hyperelliptic curves have yet to be developed. Many additional improvements to the imaginary case, for example, special forms and alternative models of the curve equation, and improved exponentiation methods using precomputations [7] and fast tripling formulas [22], could be generalized to the real case using either the Jacobian or the infrastructure.

As already pointed out in Section 2, every real hyperelliptic curve over a base field of even characteristic is isomorphic to one over the same base field where

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

the right hand side $f(x)$ in (2.1) has degree at most g. This observation is due to van der Poorten and Scheidler; details can be found in [1]. This alternative model in the real case warrants particular attention. An analogue of the Edwards model in genus 1 may also yield efficient arithmetic for genus 2 real hyperelliptic curves.

It is an open question as to whether the infrastructure or the Jacobian is the more efficient setting for cryptographic protocols. Thanks to the work of Mireles Morales, Paulus-Rück, and others, both can be used interchangeably. The balanced divisor representation of elements in the Jacobian [15] has a similar effect on the improvements to scalar multiplication in the infrastructure described in Section 5, in that both ideas eliminate almost all the extra adjustment steps required in other representations. Using $\infty - \overline{\infty}$ as a base divisor in the Jacobian is akin to the fixed-base scalar multiplication in the infrastructure using a doubling and baby step based strategy. Thus, both settings should provide comparable efficiency, but a thorough comparison of both settings has yet to be published.

The advances in efficient Jacobian arithmetic in the real model open up additional possibilities for cryptographic applications beyond what is possible in the infrastructure. For example, a preliminary investigation of pairing-based cryptography in the Jacobian of a genus 2 real hyperelliptic curve is described in [16]. The conclusion is that the real model is not as efficient as the imaginary model, even when using balanced divisor arithmetic combined with explicit formulas. However, it is possible that further investigation, including adapting some of the successful methods from the infrastructure setting, will yield improvements.

One last intriguing possibility is the idea of a Koblitz curve analogue for the real model. Scalar multiplication using a combination of an efficient Frobenius operation and baby steps could be exceptionally fast. Unfortunately, this idea does not work in the infrastructure, as considering a curve over an extension of the original base field does not enlarge the infrastructure. However, such an idea might turn out to be fruitful when considering the Jacobian of a real model instead, and further research is required.

Acknowledgements. The authors are grateful to an anonymous referee's helpful suggestions for improvement of this article.

REFERENCES

[1] AVANZI, R.—JACOBSON, M. J., JR.—SCHEIDLER, R.: *Efficient reduction of large divisors on hyperelliptic curves*, Adv. Math. Communications **4** (2010), 261–279.

[2] BERNSTEIN, D. J.—LANGE, T.: *Faster addition and doubling on elliptic curves*, in: Advances in Cryptology—ASIACRYPT '07, Kuching, Malaysia, 2007 (K. Kurosawa, ed.), Lecture Notes in Comput. Sci., Vol. 4833, Springer, Berlin, 2008, pp. 29–50.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

[3] BUCHMANN, J.: *A subexponential algorithm for the determination of class groups and regulators of algebraic number fields*, in: Séminaire de Théorie des Nombres, Paris, 1988–89, Prog. Math., Vol. 91, 1990, pp. 27–41.

[4] CANTOR, D. G.: *Computing the Jacobian of a hyperelliptic curve*, Math. Comp. **48** (1987), 95–101.

[5] CASSELS, J. W. S.—FLYNN, E. V.: *Prolegomena to a Middlebrow Arithmetic of Curves of Genus 2*, in: London Math. Soc. Lecture Note Ser., Vol. 230, Cambridge Univ. Press, Cambridge, 1996.

[6] CHEN, Z.—STORJOHANN, A.—FLETCHER, C.: *IML: Integer Matrix Library (version 1.0.2)*, `http://www.cs.uwaterloo.ca/ z4chen/iml.html, 2007.`

[7] COHEN, H.—FREY, G.—AVANZI, R.—DOCHE, C.—LANGE, T.—NGUYEN, K.——VERCAUTEREN, F.: *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, in: Discrete Math. Appl., Chapman & Hall/CRC, Boca Raton, FL, 2006.

[8] DIFFIE, W.—HELLMAN, M. E.: *New directions in cryptography*, IEEE Trans. Inform. Theory **22** (1976), 644–654.

[9] ELGAMAL, V.: *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory **IT-31** (1985), 469–472.

[10] ERICKSON, S.—HO, T.—ZEMEDKUN, S.: *Explicit projective formulas for real hyperelliptic curves of genus* 2, 2009 (in preparation).

[11] ERICKSON, S.—JACOBSON, M. J., JR.—SHANG, N.—SHEN, S.—STEIN, A.: *Explicit formulas for real hyperelliptic curves of genus* 2 *in affine representation*, in: Proceedings of First International Workshop—WAIFI '07, Madrid, 2007 (C. Carlet et al., eds.), Lecture Notes in Comput. Sci., Vol. 4547, Springer, Berlin, 2007, pp. 202–218.

[12] ERICKSON, S.—JACOBSON, M. J. JR.—STEIN, A.: *Explicit formulas for real hyperelliptic curves of genus* 2 *in affine representation*, 2009 (in preparation).

[13] FONTEIN, F.: *Groups from cyclic infrastructures and Pohlig-Hellman in certain infrastructures*, Adv. Math. Commun. **2** (2008), 293–307.

[14] ———, *The Infrastructure of a Global Field and Baby Step-Giant Step Algorithms.* Ph.D. Thesis, University of Zürich, Zürich, Switzerland, 2008.

[15] GALBRAITH, S. D.—HARRISON, M.—MIRELES MORALES, D. J.: *Efficient hyperelliptic curve arithmetic using balanced representation for divisors*, in: Algorithmic Number Theory—ANTS '08, Banff, Canada, 2008 (A. van der Poorten, ed.), Lecture Notes in Comput. Sci., Vol. 5011, Springer, Berlin, 2008, pp. 342–356.

[16] GALBRAITH, S. D.—LIN, X.—MIRELES MORALES, D. J.: *Pairings on hyperelliptic curves with a real model*, in: Pairing-Based Cryptography—Pairing '08 (S. Galbraith, ed.) Egham, UK, 2008, Lecture Notes in Comput. Sci., Vol. 5209, Springer, Berlin, 2008, pp. 265–281.

[17] GALBRAITH, S. D.—PUJOLAS, J.—RITZENTHALER, C.—SMITH, B.: *Distortion maps for supersingular genus two curves*, J. Math. Cryptology **3** (2009), 1–18.

[18] GAUDRY, P.—THOMÉ, E.—THÉRIAULT, N.—DIEM, C.: *A double large prime variation for small genus hyperelliptic index calculus*, Math. Comp. **76** (2007), 475–492.

[19] HAMMELL, J. F.: *Index Calculus in the Infrastructure of Real Quadratic Function Fields.* Master's Thesis, University of Calgary, Canada, 2008.

[20] HAMMELL, J. F.—JACOBSON, M. J., JR.: *Index-calculus algorithms in real quadratic function fields*, 2009 (in preparation).

[21] HANKERSON, D.—MENEZES, A.—VANSTONE, S.: *Guide to Elliptic Curve Cryptography*. Springer, New York, 2004.

[22] IMBERT, L.—JACOBSON, M. J., JR.—SCHMIDT, A.: *Fast ideal cubing in imaginary quadratic number and function fields*, Adv. Math. Communications **4** (2010), 237–260.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

[23] JACOBSON, M. J., JR.—MENEZES, A. J.—STEIN, A.: *Hyperelliptic curves and cryptography*, in: Selected papers from the International Conference on Number Theory, Banff, AB, Canada, 2003 (A. van der Poorten, et. al., eds.), Fields Inst. Commun., Vol. 41, Amer. Math. Soc., Providence, RI, 2004, pp. 255–282.

[24] JACOBSON, M. J., JR.—SCHEIDLER, R.—STEIN, A.: *Cryptographic protocols on real and imaginary hyperelliptic curves*, Adv. Math. Commun. **1** (2007), 197–221.

[25] ______, *Fast arithmetic on hyperelliptic curves via continued fraction expansions*, in: Advances in Coding Theory and Cryptology (T. Shaska, T. et al., eds.), Series on Coding Theory and Cryptology, Vol. 3, World Scientific, Hackensack, NJ, 2007, pp. 200–243.

[26] JACOBSON, M. J., JR.—STEIN, A.—VELICHKA, M. D.: *Computing discrete logarithms on high-genus hyperelliptic curves over even characteristic finite fields* (in preparation), 2009.

[27] JACOBSON, M. J., JR.—VAN DER POORTEN, A. J.: , *Computational aspects of NUCOMP*, in: Algorithmic Number Theory—ANTS-V, Sydney, Australia, 2002 (C. Fieker et al., eds.), Lecture Notes in Comput. Sci., Vol. 2369, Springer, Berlin, 2002, pp. 120–133.

[28] KOBLITZ, N.: *Elliptic curve cryptosystems*, Math. Comp. **48** (1987), 203–209.

[29] ______, *Hyperelliptic cryptosystems*, J. Cryptology **1** (1989), 139–150.

[30] LANGE, T.: *Formulae for arithmetic on genus 2 hyperelliptic curves*, Appl. Algebra Engrg. Comm. Comput. **15** (2005), 295–328.

[31] MENEZES, A. J.—WU, Y.-H.—ZUCCHERATO, R. J.: *An elementary introduction to hyperelliptic curves*, in: Algebraic Aspects of Cryptography, Algorithms Comput. Math., Vol. 3, Springer, Berlin, 1998, pp. 155–178.

[32] MILLER, V.: *Use of elliptic curves in cryptography*, in: Advances in Cryptology——CRYPTO '85, Santa Barbara, California, 1985, Lecture Notes in Comput. Sci., Vol. 218, Springer, Berlin, 1986, pp. 417–426.

[33] MIRELES MORALES, D. J.: *An analysis of the infrastructure in real function fields*, Eprint archive no. 2008/299, 2008.

[34] MÜLLER,V.—STEIN, A.—THIEL, C.: *Computing discrete logarithms in real quadratic congruence function fields of large genus*, Math. Comp. **68** (1999), 807–822.

[35] PAULUS, S.—RÜCK, H.-G.: *Real and imaginary quadratic representations of hyperelliptic function fields*, Math. Comp. **68** (1999), 1233–1241.

[36] POHLIG, S. C.—HELLMAN, M. E.: *An improved algorithm for computing logarithms over $GF(p)$ and it's cryptographic significance*, IEEE Trans. Inf. Theory **24** (1978), 106–110.

[37] SCHEIDLER, R.: *Cryptography in quadratic function fields*, Des. Codes Cryptogr. **22** (2001), 239–264.

[38] SCHEIDLER, R.—BUCHMANN, J. A.—WILLIAMS, H. C.: *A key exchange protocol using real quadratic fields*, J. Cryptology **7** (1994), 171–199.

[39] SCHEIDLER, R.—STEIN, A.—WILLIAMS, H. C.: *Key-exchange in real quadratic congruence function fields*, Des. Codes Cryptogr. **7** (1996), 153–174.

[40] SHANKS, D.: *The infrastructure of a real quadratic field and its applications*, in: Proceedings of Number Theory Conf., Univ. Colorado, Boulder, Colorado, 1972, pp. 217–224.

[41] SHOUP, V.: *NTL: A Library for doing Number Theory (version 5.4.2)*, 2008, `http://www.shoup.net`.

[42] STEIN, A.: *Equivalences between elliptic curves and real quadratic congruence function fields*, J. Théorie Nombr. Bordeaux **9** (1997), 75–95.

[43] ______, *Sharp upper bounds for arithmetics in hyperelliptic function fields*, J. Ramanujan Math. Soc. **16** (2001), 1–86.

[44] STEIN, A.—TESKE, E.: *Explicit bounds and heuristics on class numbers in hyperelliptic function fields*, Math. Comp. **71** (2002), 837–861.

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM

[45] ———, *The parallelized Pollard kangaroo method in real quadratic function fields*, Math. Comp. **71** (2002), 793–814.

[46] ———, *Optimized baby-step giant-step methods in hyperelliptic function fields*, J. Ramanujan Math. Soc. **20** (2005), 1–32.

[47] STICHTENOTH, H.: *Algebraic Function Fields and Codes* (2nd ed.), Springer, Berlin, 2009.

[48] VELICHKA, M. D.: *Improvements to Index Calculus Algorithms for Solving the Hyperelliptic Curve Discrete Logarithm Problem Over Characteristic Two Finite Fields.* Master's thesis, University of Calgary, Calgary, Canada, 2008.

Received February 24, 2009

Michael J. Jacobson, Jr.
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta
Canada T2N 1N4

E-mail: jacobs@cpsc.ucalgary.ca

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

E-mail: rscheidl@math.ucalgary.ca

Andreas Stein
Institut für Mathematik
Carl-von-Ossietzky Universität Oldenburg
D-26111 Oldenburg
GERMANY
E-mail: andreas.stein1@uni-oldenburg.de

Authenticated | rscheidl@math.ucalgary.ca author's copy
Download Date | 11/18/12 6:52 PM