Advances in Mathematics of Communications
Volume 8, No. 4, 2014, 389–406
doi:10.3934/amc.2014.8.389

COMPARISON OF SCALAR MULTIPLICATION ON REAL HYPERELLIPTIC CURVES

Michael J. Jacobson, Jr.

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

Monireh Rezai Rad and Renate Scheidler

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

(Communicated by Roger Oyono)

Abstract. Real hyperelliptic curves admit two structures suitable for cryptography — the Jacobian (a finite abelian group) and the infrastructure. Mireles Morales described precisely the relationship between these two structures, and made the assertion that when implemented with balanced divisor arithmetic, the Jacobian generically yields more efficient arithmetic than the infrastructure for cryptographic applications. We confirm that this assertion holds for genus two curves, through rigorous analysis and the first detailed numerical performance comparisons, showing that cryptographic key agreement can be performed in the Jacobian without any extra operations beyond those required for basic scalar multiplication. We also present a modified version of Mireles Morales' map that more clearly reveals the algorithmic relationship between the two structures.

1. Introduction

In 1976, Diffie and Hellman [3] introduced their celebrated key agreement protocol. While they originally described their scheme in the context of finite fields, other suitable finite abelian groups have since been successfully employed. In particular, the Jacobian of a hyperelliptic curve over a finite field was first proposed for this purpose by Koblitz [12], spawning a great deal of work on the subject; see [2] for a partial survey.

The majority of work on hyperelliptic curve cryptosystems has been confined to so-called imaginary hyperelliptic curves, those originally proposed in [12]. However, two other models are available, including the more general real model, which often arises naturally in constructive methods for generating cryptographically suitable curves. In addition to the Jacobian, the real model admits a second structure called the infrastructure. Although not a group, it was nevertheless shown by Scheidler, Stein, and Williams [15] that it could also be used for cryptographic purposes. First attempts to describe arithmetic in the Jacobian and the infrastructure of a

2010 *Mathematics Subject Classification:* Primary: 11G20, 11R11, 11R29, 11Y40; Secondary: 11Y16, 14G50.

Key words and phrases: Real hyperelliptic curve, Jacobian, balanced divisor, infrastructure, scalar multiplication, cryptographic key agreement.

The first and third authors are supported by NSERC of Canada.

real model required extra adjustment steps that were not required in the imaginary model. Consequently, the real model was deemed to be less desirable for cryptographic applications due to its slower performance in practice.

Recent work on arithmetic in the real model has begun to close this performance gap. Galbraith et al. [8] introduced the notion of balanced divisors for arithmetic in the Jacobian, which heuristically removed almost all adjustment steps. Jacobson et al. [9] described improvements to scalar multiplication in the infrastructure. In the variable base case, where the base divisor is an input parameter, a technique was introduced in which all adjustment steps were eliminated heuristically at the cost of a small precomputation. In the fixed base case, a modified algorithm was described that requires no adjustment steps and in which divisor additions were replaced by the faster "baby step" operation.

Although both the Jacobian and the infrastructure can be used for cryptographic applications, it was originally not clear exactly how they were related, or which one offered faster performance in practice. In unpublished work, Mireles Morales [14] described the relationship between the infrastructure and a particular cyclic subgroup of the Jacobian. He showed explicitly that these two structures are equivalent in the sense that any computation in one structure can be reduced to an analogous computation in the other. Furthermore, he made the claim that when coupled with the balanced divisor arithmetic from [8], performing the desired computations in the Jacobian should always be more efficient than in the infrastructure. However, he did not take into the account the improved infrastructure arithmetic from [9] in his analysis.

In this paper, we investigate the assertion by Mireles Morales, considering state-of-the-art algorithms for arithmetic in both settings, including the results of [9]. We confirm that scalar multiplication in the Jacobian using balanced divisor representatives is slightly faster than the corresponding operations in the infrastructure. We describe how to perform both types of scalar multiplication in the Jacobian using the same performance improvements from the infrastructure. We formally analyze the cost of the resulting algorithms as compared with those in the infrastructure, and provide numerical experiments showing that key agreement using the Jacobian is slightly faster in genus two. We also present a modified version of Mireles Morales' map that clarifies the algorithmic relationship between the Jacobian and the infrastructure.

We stress that the main contribution of this paper is the analytical comparison between scalar multiplication in the Jacobian and the infrastructure using the latest algorithms, answering the question of which setting is faster based on operation counts. The numerical results we provide are meant only to support these results and give a relative performance comparison with the imaginary case. We use a common hardware and software platform in order to provide a fair comparison. Divisor arithmetic is done using the affine representation in all cases, because projective formulas in the real case are still under development and have not yet been published. For these reasons, a highly-optimized dedicated software implementation using, for example, special types of defining equations and coordinate systems, is beyond the scope of this paper.

This paper is organized as follows. We describe real hyperelliptic curves and the Jacobian in Section 2, as well as algorithms for computing in the Jacobian using balanced divisors as presented in [8]. In Section 3, we review the infrastructure of a real hyperelliptic curve and the main arithmetic operations in that context.

Mireles Morales' map between infrastructure and Jacobian [14] and our modified map are presented in Section 4. The improved infrastructure arithmetic of [9] takes advantage of certain heuristics involving "holes" (i.e. missing images) in this map; these holes and their properties are the subject of Section 5. We describe how results of Fontein [6] justify the heuristic assumptions used in the arithmetic from [9], and how these can be applied analogously in the Jacobian setting. The resulting scalar multiplication algorithms in both settings, using fixed and variable base divisors, are discussed in Section 6; this includes a comparative analysis. Numerical data supporting our analysis is presented in Section 7, followed by conclusions and suggestions for future work in Section 8.

2. Hyperelliptic curves and balanced divisors

Much of the literature on hyperelliptic curve cryptography considers *imaginary* hyperelliptic curves which have one rational point at infinity. Here, every degree zero divisor class contains a unique reduced representative, and the resulting efficient divisor arithmetic has been investigated extensively. In contrast, *real* models have two rational infinite points, so any degree zero divisor class generally contains a large number of reduced divisors. Thus, additional conditions are required in order to establish a unique representation for each degree zero divisor class. In [8], Galbraith et al. introduced a representation of elements in the Jacobian of a real hyperelliptic curve which is *balanced* at the two points at infinity. In this section, we briefly review the main definitions and operations for real hyperelliptic curves using balanced divisors. For more details, we refer the reader to [4], [5], and [8].

Throughout, let $k = \mathbb{F}_q$ be a finite field of prime power order q, $k[x]$ the ring of polynomials in x over k, and $k(x)$ the field of rational functions in x over k.

Definition 2.1. A *hyperelliptic curve C of genus g (defined) over k* is an affine curve that is absolutely irreducible, smooth, and given by an equation of the form

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

where $f, h \in k[x]$ satisfy one of the following two conditions:

1. $\deg(f) = 2g + 1$, f is monic, and $h = 0$ if q is odd whereas $\deg(h) \leq g$ if q is even. In this case, C is said to be *imaginary*;
2. $\deg(f) = 2g + 2$, f is monic and $h = 0$ if q is odd, whereas f has leading coefficient $e^2 + e$ for some $e \in k^*$ and h is monic of degree $g + 1$ if q is even. In this case, C is said to be *real*.[1]

The coordinate ring of C is $k[C] = k[x, Y]/(Y^2 + h(x)Y - f(x)) = k[x, y]$, and its function field is $k(C) = k(x, y)$. The roots of the curve equation C are y and $-(y + h(x)) \in k[C]$, and the *hyperelliptic involution* on C sends one root to the other.

When C is imaginary, it has a unique k-rational point at infinity, denoted by ∞, whereas if C is real, there are two k-rational infinite points on C, denoted by ∞^+ and ∞^-, respectively. For real models, we denote by ν_{∞^+} and ν_{∞^-} the two corresponding discrete valuations on $k(C)$; it is straightforward to see that $v_{\infty^+}(y) = v_{\infty^-}(y) = -(g+1)$. We fix an embedding of $k(C)$ into the field $k((x^{-1}))$ of Puiseux series in x^{-1} over k, the completion of $k(x)$ with respect to both ∞^+

[1] If q is even, the variable transformation $y \to y + ex^{g+1}$ produces an isomorphic curve of the form $y^2 + h(x)y = F(x)$ with $\deg(F) \leq 2g + 1$. Following [8], we will not consider such models here.

and ∞^-. Then the floor function on $k((x^{-1}))$ is well-defined on $k(C)$; in particular, $\lfloor y \rfloor$ and $-\lfloor y \rfloor - h(x)$ are polynomials in $k[x]$ of degree $g+1$. Following [8], we let a_+ and a_- be their respective leading coefficients, so $\{a_+, a_-\} = \{1, -1\}$ if q is odd and $\{a_+, a_-\} = \{e, e+1\}$ (with e as given in Definition 2.1) if q is even. The quantities $\lfloor y \rfloor$, a_+ and a_- are required in several algorithms later on.

For any hyperelliptic curve C over k, we denote by $\mathrm{Div}^0(C)$ its group of degree zero divisors on C defined over k. Henceforth, all divisors are assumed to be defined over k. The hyperelliptic involution on C naturally extends to divisors, and the image of a divisor D under this map is denoted by $\overline{D}$.

We denote by $\mathrm{div}(\alpha)$ the principal divisor of $\alpha \in k(C)^*$. Two divisors $D_1, D_2 \in \mathrm{Div}^0(C)$ are *(linearly) equivalent*, denoted $D_1 \equiv D_2$, if they differ by a principal divisor. The*(degree zero) divisor class group*, or *Jacobian*, of C over k, denoted $Cl^0(C)$, is the group of divisor classes under linear equivalence. The class of a divisor $D \in \mathrm{Div}^0(C)$ is written as $[D]$.

A divisor D is *affine* if it is not supported at infinite points. Any affine semi-reduced divisor D on C is determined by its *Mumford representation* consisting of a pair of polynomials $Q, P \in k[x]$ where 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 write $D = (Q, P)$. D is *reduced* if $\deg(Q) \leq g$. If C is imaginary, then every degree zero divisor class has a unique representative whose affine part is reduced, whereas if C is real, then each such class contains many divisors with reduced affine support. To establish a unique representation of divisor classes for real models, Galbraith et al. in [8] introduced the effective k-rational degree g divisor

$$D_\infty = \left\lceil \frac{g}{2} \right\rceil \infty^+ + \left\lfloor \frac{g}{2} \right\rfloor \infty^-$$

and showed that every element of $Cl^0(C)$ has a unique representative of the form $D - D_\infty$, where

$$D = D' + n\,\infty^+ + (g - n - \deg(D'))\,\infty^-$$

is an effective k-rational divisor of degree g whose affine part D' is reduced. D is the *balanced representative* of its class and is written as $D = (D', n)$ for brevity. Note that the effectiveness of D forces $0 \leq n \leq g - \deg(D')$. We will argue later on that generically, almost all balanced divisors have $n = 0$; see Section 5.

Henceforth, up to and including Section 5, we only consider real hyperelliptic curves. In the sequel, we will require the following balanced representations:

Example 2.2. For any real hyperelliptic curve of genus g, we have the following balanced representations:

a) The balanced representative of the principal divisor class is $((1,0), \lceil g/2 \rceil)$.
b) The balanced representative of $[\infty^+ - \infty^-]$ is $((1,0), \lceil g/2 \rceil + 1)$.
c) The balanced representative of $[\infty^- - \infty^+]$ is $((1,0), \lceil g/2 \rceil - 1)$.

Definition 2.3 (Definition 5 of [8])**.** For any two divisors D_1 and D_2, the integers ω^+ and ω^- are said to be a pair of *counterweights* for D_1 and D_2 if

$$D_1 \equiv D_2 + \omega^+\infty^+ + \omega^-\infty^- \ .$$

The set of all pairs of counterweights for D_1 and D_2 is denoted by $\omega(D_1, D_2)$.

2.1. THE JACOBIAN OPERATION USING BALANCED DIVISORS. In this section, we summarize the group operation on the Jacobian using balanced representatives. The main algorithms for Jacobian arithmetic as introduced in [8] are given below.

Throughout, we assume that C is a real hyperelliptic curve of genus g over $k = \mathbb{F}_q$, (although Algorithm 1 applies equally to imaginary hyperelliptic curves).

Algorithms 1 and 2 are simply Cantor's well-known divisor addition and reduction of balanced divisors, respectively, with the relevant counterweights included in the output.

Algorithm 1 Divisor Addition

Input: Two semi-reduced affine divisors $D_1' = (Q_1, P_1)$, and $D_2' = (Q_2, P_2)$.
Output: $(D', (\omega^+, \omega^-)) = \text{add}(D_1', D_2')$, where $D' = (Q, P)$ is a semi-reduced affine divisor, and $(\omega^+, \omega^-) \in \omega(D_1' + D_2', D')$.

1: Use the extended Euclidean algorithm to find polynomials $s', e_1, e_2 \in \mathbb{F}_q[x]$ such that
$$s' = \gcd(Q_1, Q_2) = e_1 Q_1 + e_2 Q_2$$

2: Use the extended Euclidean algorithm to find polynomials $s, c_1, c_2 \in \mathbb{F}_q[x]$ such that
$$s = \gcd(s', P_1 + P_2 + h) = c_1 s' + c_2(P_1 + P_2 + h)$$

3: Let $s_1 = c_1 e_1$, $s_2 = c_1 e_2$, and $s_3 = c_2$, so that
$$s = s_1 Q_1 + s_2 Q_2 + s_3(P_1 + P_2 + h)$$

4: Set
$$Q = \frac{Q_1 Q_2}{s^2} \text{ and } P \equiv \frac{s_1 Q_1 P_2 + s_2 Q_2 P_1 + s_3(P_1 P_2 + f)}{s} \pmod{Q}$$

5: **return** (Q, P) and $(\deg(s), \deg(s))$

The output of Algorithm 2 is reduced, but need not be in balanced form. Algorithms 3 and 4 accomplish this task. To see that these two algorithms are correct, consider a reduced input divisor $D_0 = ((Q_0, P_0), n_0)$, and let $D_0' = (Q_0, P_0)$ be the affine part of D_0. If $(D_1', (\omega^+, \omega^-))$ is the output of either Algorithm 3 or 4, then $D_0' \equiv D_1' + \omega^+\infty^+ + \omega^-\infty^-$. Thus,

$$\begin{aligned} D_0 &= D_0' + n_0\infty^+ + (g - n_0 - \deg(D_0'))\infty^- - D_\infty \\ &\equiv D_1' + (n_0 + \omega^+)\infty^+ + (g - n_0 - \deg(D_0') + \omega^-)\infty^- - D_\infty \\ &= (D_1', n_1) \ , \end{aligned}$$

where $n_1 = n_0 + \omega^+$. Therefore, if D_0 is not balanced, i.e. n_0 does not satisfy the condition $0 \le n_0 \le g - \deg(Q_0)$, then we can compute the balanced representative of the divisor class $[D_0]$ by applying the appropriate number of successive red_{∞^+} or red_{∞^-} steps to D_0 and updating n_0 in the process. When $n_0 > g - \deg(Q_0)$, then red_{∞^+} (Algorithm 3) must be applied to decrease the value of n_0, whereas when $n_0 < 0$, we use red_{∞^-} (Algorithm 4) to increase the value of n_0.

As pointed out in Remark 2 of [8], Algorithm 3 can be interpreted generically as composition with $\infty^+ - \infty^-$; similarly, Algorithm 4 corresponds to composition by its negative $\infty^- - \infty^+$. Algorithms 6 and 7 support this observation; see Section 5 for further details.

Algorithm 2 Divisor Reduction

Input: A semi-reduced affine divisor $D_0' = (Q_0, P_0)$ of degree $d_0 \geq g+1$.
Output: $(D_1', (\omega^+, \omega^-)) = \text{red}(D_0')$, where $D_1' = (Q_1, P_1)$ is a reduced affine divisor, and $(\omega^+, \omega^-) \in \omega(D_0', D_1')$.

Set $(\omega^+, \omega^-) = (0, 0)$
while $d_0 > g$ **do**
  if $d_0 \geq g+2$ **then**
    Let $Q_1 = \frac{f - P_0 h - P_0^2}{Q_0}$
    Let $P_1 = (-h - P_0) \pmod{Q}_1$
    Set $d_1 = \deg(Q_1)$
    if the leading term of P_0 is $a_+ x^{g+1}$ **then**
      Update $(\omega^+, \omega^-) = (\omega^+ + d_0 - g - 1, \omega^- + g + 1 - d_1)$
    else if the leading term of P_0 is $a_- x^{g+1}$ **then**
      Update $(\omega^+, \omega^-) = (\omega^+ + g + 1 - d_1, \omega^- + d_0 - g - 1)$
    else
      Update $(\omega^+, \omega^-) = (\omega^+ + \frac{d_0 - d_1}{2}, \omega^- + \frac{d_0 - d_1}{2})$
    end if
  else
    Compute the leading coefficient a of $(P_0 + \lfloor y \rfloor)/Q_0$
    Let $P_1 = aQ_0 - P_0 + h$
    Let $Q_1 = \frac{P_1^2 + hP_1 - f}{Q_0}$ made monic
    Set $d_1 = \deg(Q_1)$
    Update $(\omega^+, \omega^-) = (\omega^+ + d_0 - g - 1, \omega^- + g + 1 - d_1)$
  end if
  Set $Q_0 = Q_1$, $P_0 = P_1$, and $d_0 = d_1$;
end while
return (Q_1, P_1), and (ω^+, ω^-).

Algorithm 3 Balancing Step

Input: A reduced affine divisor $D_0' = (Q_0, P_0)$ of degree d_0.
Output: $(D_1', (\omega^+, \omega^-)) = \text{red}_{\infty^+}(D_0')$, where $D_1' = (Q_1, P_1)$ is a reduced affine divisor of degree d_1, and $(\omega^+, \omega^-) \in \omega(D_0', D_1')$.

$P' = \lfloor y \rfloor + ((P_0 - \lfloor y \rfloor) \pmod{Q_0})$
$Q_1 = \frac{P'^2 + hP' - f}{Q_0}$ made monic
$P_1 = -h - P' \pmod{Q_1}$
$d_1 = \deg(Q_1)$
$(\omega^+, \omega^-) = (d_0 - g - 1, g + 1 - d_1)$
return (Q_1, P_1) and (ω^+, ω^-)

Note that when Algorithm 3 is applied to the affine part of a divisor without considering the value of n, it corresponds exactly to the baby step operation in the infrastructure (see Section 3). Similarly, Algorithm 4 corresponds to an inverse (or backward) infrastructure baby step.

For any two balanced divisors D_1 and D_2 on C, the balanced representative of the class of $D_1 + D_2$ is denoted by $D_1 \oplus D_2$, so $[D_1] + [D_2] = [D_1 \oplus D_2]$. Algorithm 5 computes the divisor $D_1 \oplus D_2$, and is essentially Algorithm 4 of [8]. At most $\lceil g/2 \rceil$

Algorithm 4 Inverse Balancing Step

Input: A reduced affine divisor $D'_0 = (Q_0, P_0)$ of degree d_0.
Output: $(D'_1, (\omega^+, \omega^-)) = \mathrm{red}_{\infty^-}(D'_0) =$, where $D'_1 = (Q_1, P_1)$ is a reduced affine divisor of degree d_1, and $(\omega^+, \omega^-) \in \omega(D'_0, D'_1)$.

1: $P' = -\lfloor y \rfloor - h(x) + ((P_0 + \lfloor y \rfloor + h(x)) \pmod{Q_0})$
2: $Q_1 = \frac{P'^2 + hP' - f}{Q_0}$ made monic
3: $P_1 = -h - P' \pmod{Q_1}$
4: $d_1 = \deg(Q_1)$
5: $(\omega^+, \omega^-) = (g + 1 - d_1, d_0 - g - 1)$
6: **return** (Q_1, P_1) and (ω^+, ω^-)

reduction steps (steps 3-7) are required in Algorithm 5 obtain a reduced divisor. Since generically, almost all balanced divisors are of the form $D = (D', 0)$, generally no balancing steps (steps 8-16) are needed.

Algorithm 5 Divisor Class Addition

Input: Two balanced divisors $D_1 = (D'_1, n_1)$ and $D_2 = (D'_2, n_2)$.
Output: The balanced divisor $(D'_3, n_3) = D_1 \oplus D_2$ equivalent to $D_1 + D_2$.

1: Call Algorithm 1 on inputs D'_1 and D'_2 to obtain $(D'_3, (a, b)) = \mathrm{add}(D'_1, D'_2)$
2: Set $\omega^+ = n_1 + n_2 + a$ and $\omega^- = 2g - \deg(D'_1) - \deg(D'_2) - n_1 - n_2 + b$
3: **while** $\deg(D'_3) > g + 1$ **do**
4: Call Algorithm 2 on input D'_3 to obtain $D', (a, b) = \mathrm{red}(D'_3)$
5: Update $(\omega^+, \omega^-) = (\omega^+ + a, \omega^- + b)$
6: Set $D'_3 = D'$
7: **end while**
8: **while** $\omega^+ < \lceil g/2 \rceil$ or $\omega^- < \lfloor g/2 \rfloor$ **do**
9: **if** $\omega^+ > \omega^-$ **then**
10: Call Algorithm 3 on input D'_3 to obtain $(D', (a, b)) = \mathrm{red}_{\infty^+}(D'_3)$
11: **else**
12: Call Algorithm 4 on input D'_3 to obtain $(D', (a, b)) = \mathrm{red}_{\infty^-}(D'_3)$
13: **end if**
14: Update $(\omega^+, \omega^-) = (\omega^+ + a, \omega^- + b)$
15: Set $D'_3 = D'$
16: **end while**
17: $n_3 = \omega^+ - \lceil g/2 \rceil$
18: **return** $D_3 = (D'_3, n_3)$

Of particular interest in cryptographic applications is the special case of addition or subtraction by the class $[\infty^+ - \infty^-]$ as described in Algorithms 6 and 7, respectively. This is used, for example, in round 1 of the Diffie-Hellman protocol if $\infty^+ - \infty^-$ is chosen as public base divisor and scalar multiplication is performed using the non-adjacent form of the scalars. Here, Example 2.2 shows that composition and reduction (steps 1 and 3-7 of Algorithm 5, respectively) are unnecessary, and only one balancing step is needed.

To prove Algorithms 6 and 7 correct, set $d = \deg(D')$ and note that

$$D \pm (\infty^+ - \infty^-) = D' + (n \pm 1)\infty^+ + (g - n - d \mp 1)\infty^- - D_\infty \ .$$

Algorithm 6 Addition by $[\infty^+ - \infty^-]$

Input: A balanced divisor $D = (D', n)$.
Output: The balanced divisor $D \oplus ((1,0), \lceil g/2 \rceil + 1)$.
1: **if** $n = g - \deg(D')$ **then**
2: Call Algorithm 3 on input D' to obtain $(E', (a,b)) = \text{red}_{\infty^+}(D')$
3: **return** $(E', 0)$
4: **else**
5: **return** $(D', n+1)$
6: **end if**

Algorithm 7 Subtraction by $[\infty^+ - \infty^-]$

Input: A balanced divisor $D = (D', n)$
Output: A balanced divisor equivalent to $D \oplus ((1,0), \lceil g/2 \rceil - 1)$
1: **if** $n = 0$ **then**
2: Call Algorithm 4 on input D' to obtain $(E', (a,b)) = \text{red}_{\infty^-}(D')$
3: **return** $(E', g - \deg(E'))$
4: **else**
5: **return** $(D', n-1)$
6: **end if**

If $1 \leq n \leq g - d - 1$, then this is balanced, else the output $(E', (a,b))$ of step 2 of both algorithms satisfies

$$D \pm (\infty^+ - \infty^-) \equiv E' + (n \pm 1 + a)\infty^+ + (g - n - d \mp 1 + b)\infty^- - D_\infty \ .$$

If $n = g - d$, then Algorithm 6 calls Algorithm 3 which produces $a = d - g - 1$, so $n + 1 + a = 0$. If $n = 0$, then Algorithm 7 calls Algorithm 4 which produces $a = g + 1 - \deg(E')$, so $n - 1 + a = g - \deg(E')$.

Our last algorithm (Algorithm 8) is Algorithm 5 in [8] which describes how to compute a balanced representative of the inverse of a divisor class. Note that there are minor errors on lines 4 (m_1 instead of n_1) and 7 (0 instead of n_1) in [8] which are corrected here.

For the correctness of Algorithm 8, observe that the conjugate divisor of (Q_0, P_0) is D' as given in step 4. Moreover, $\overline{D}_\infty = D_\infty$ if g is even, and $\overline{D}_\infty = D_\infty - (\infty^+ - \infty^-)$ if g is odd. Hence

$$\overline{D}_0 = D' + (g - \deg(Q_0) - n_0)\infty^+ + n_0\infty^- - D_\infty$$

if g is even and

$$\overline{D}_0 = D' + (g - \deg(Q_0) - n_0)\infty^+ + n_0\infty^- + (\infty^+ - \infty^-) - D_\infty$$

if g is odd. So no balancing is needed unless g is odd and $n_0 = 0$, in which case a subtraction by $\infty^+ - \infty^-$ (Algorithm 7) produces a balanced divisor.

3. INFRASTRUCTURE

We summarize the main properties of the infrastructure; details can be found in [18] and [16]. As before, let $C : y^2 + h(x)y = f(x)$ be a real hyperelliptic curve of genus g over $k = \mathbb{F}_q$ with coordinate ring $k[C]$. The infinite degree zero divisor $\infty^+ - \infty^-$ plays an important role here. The order R in $Cl^0(C)$ of the class of this divisor is the *regulator* of C. The divisor $R[\infty^+ - \infty^-]$ is principal and is the divisor

Algorithm 8 Divisor Inversion

Input: A balanced divisor divisor $D_0 = ((Q_0, P_0), n_0)$
Output: A balanced divisor $D_1 = ((Q_1, P_1), n_1)$ such that $[D_1] = -[D_0]$

if $Q_0 = 1$ **then**
  return D_0
else
  Let $D' = (Q_0, (-h - P_0 \pmod{Q_0}))$
  if g is even **then**
    return $(D', g - \deg(Q_0) - n_0)$
  else if g is odd and $n_0 > 0$ **then**
    return $(D', g - n_0 - \deg(Q_0) + 1)$
  else
    Call Algorithm 7 on input $(D', g - \deg(Q_0) - n_0)$ to obtain (D_1', n_1)
    return (D_1', n_1)
  end if
end if

of a *fundamental unit* of $k[C]$, i.e. a generator of the infinite cyclic group $k[C]^*/k^*$. The quotient $|Cl^0(C)|/R$, i.e. the index of the cyclic subgroup $G = \langle[\infty^+ - \infty^-]\rangle$ in $Cl^0(C)$, is equal to the ideal class number of $k[C]$. This index is small for most real hyperelliptic curves, and in fact frequently $Cl^0(C) = G$. Since the Hasse-Weil bounds establish $(\sqrt{q} - 1)^{2g} \leq |Cl^0(C)| \leq (\sqrt{q} + 1)^{2g}$, the regulator is generally of magnitude q^g.

Every non-zero $k[C]$-ideal $\mathfrak{a}$ is a $k[x]$-module of rank 2 with a basis of the form $\{SQ, S(P + y)\}$ where $S, Q, P \in k[x]$, and Q divides $P^2 + Ph - f$; write $\mathfrak{a} = [SQ, S(P + y)]$. If we take S and Q to be monic, then Q is unique and P is unique modulo Q. The ideal $\mathfrak{a}$ is *primitive* if $S = 1$ and *reduced* if additionally $\deg(Q) \leq g$. Hence, the primitive $k[C]$-ideals are in one-to-one correspondence with the semi-reduced affine divisors on C by virtue of mapping the $k[C]$-ideal $\mathfrak{a} = [Q, (P + y)]$ to the affine divisor $\mathrm{div}(\mathfrak{a}) = (Q, P)$. Under this mapping, reduced ideals are sent to reduced affine divisors. In fact, this map is simply a restriction of the well-known isomorphism from the group of non-zero fractional $k[C]$-ideals under multiplication onto the group of affine divisors on C defined over k under addition. The *degree* of a $k[C]$-ideal $\mathfrak{a}$ is $\deg(\mathfrak{a}) = \deg(\mathrm{div}(\mathfrak{a}))$, i.e. the degree of the corresponding affine divisor.

A $k[C]$-ideal $\mathfrak{a}$ is *principal* if it consists of all the $k[C]$-multiples of some fixed element $\alpha \in \mathfrak{a}$; write $\mathfrak{a} = (\alpha)$. Every non-zero principal $k[C]$-ideal has a generator α with $-R < \nu_{\infty^+}(\alpha) \leq 0$ that is unique up to k^*-multiples.

Definition 3.1. The *infrastructure* of C is defined to be the set $\mathcal{R}$ of all reduced principal ideals of $k[C]$. For every ideal $\mathfrak{a} = (\alpha) \in \mathcal{R}$ with $-R < \nu_{\infty^+}(\alpha) \leq 0$, the *distance* of $\mathfrak{a}$ is defined to be $\delta(\mathfrak{a}) = -\nu_{\infty^+}(\alpha)$. If $\mathfrak{a}, \mathfrak{b}$ are infrastructure ideals with $\delta(\mathfrak{a}) \geq \delta(\mathfrak{b})$, then the distance from $\mathfrak{b}$ to $\mathfrak{a}$ is $\delta(\mathfrak{a}, \mathfrak{b}) = \delta(\mathfrak{a}) - \delta(\mathfrak{b})$.

Example 3.2. The trivial $k[C]$-ideal $\mathfrak{a} = (1)$ has basis $\{1, y\}$ and is hence an infrastructure ideal of distance zero.

The fact that no two infrastructure ideals have the same distance imposes an ordering on $\mathcal{R}$ by distance:

$$\mathcal{R} = \{\mathfrak{a}_1, \mathfrak{a}_2, ..., \mathfrak{a}_r\}, \quad 0 = \delta_1 < \delta_2 < \cdots < \delta_r < R \ ,$$

where $\mathfrak{a}_1 = (1)$ and $\delta_i = \delta(\mathfrak{a}_i)$ for $1 \leq i \leq r$.

Computing the distance of a given infrastructure ideal is computationally infeasible — this is equivalent to the *Principal Ideal Problem* in $k[C]$ — but there is an efficient way to compute the relative distance of two successive ideals. In [9] and [18], it was shown that

$$\delta_{i+1} = \delta_i + g + 1 - \deg(Q_i) \quad \text{for } 0 \leq i \leq r-1 \ , \tag{3.1}$$

where $\mathfrak{a}_{i+1} = [Q_i, P_i + y]$. It is now easy to see that $\delta_1 = 0$, $\delta_2 = g+1$, $1 \leq \delta_i - \delta_{i-1} \leq g$ for $3 \leq i \leq r$, and $R \leq \delta_r + g$. Moreover, induction yields

$$g + i - 1 \leq \delta_i \leq (i-1)g + 1 \text{ for } 2 \leq i \leq r \ . \tag{3.2}$$

Also $r + g \leq R \leq rg + 1$. Since R is generally exponentially large compared to g, this implies $|\mathcal{R}| = r \approx R \approx q^g$.

The infrastructure supports two main operations. The first operation, named the *baby step*, computes $\mathfrak{a}_{i+1}$ from $\mathfrak{a}_i$, along with the relative distance $\delta_{i+1} - \delta_i$. Formulas for the baby step are given in [18] and [10]. The second operation on $\mathcal{R}$, referred to as the *giant step* and denoted $\otimes$, computes the first reduced ideal $\mathfrak{a} \otimes \mathfrak{b}$ equivalent to the ideal product $\mathfrak{ab}$ when applying reduction. In fact, $\mathcal{R}$ is "almost" an abelian group under $\otimes$, failing associativity only barely. More exactly, if $\mathfrak{a}, \mathfrak{b} \in \mathcal{R}$, then

$$\delta(\mathfrak{a} \otimes \mathfrak{b}) = \delta(\mathfrak{a}) + \delta(\mathfrak{b}) - d \quad \text{with} \quad 0 \leq d \leq 2g \ .$$

Here, the "shortfall" d in distance tends to be very small compared to $\delta(\mathfrak{a})$ and $\delta(\mathfrak{b})$, and is effectively computable as part of the giant step. It is expected to be equal to $\lceil g/2 \rceil$; see (H2) in Section 5. For more details, we refer the reader to [9], [11], and [16].

Since baby steps almost always produce an increase of 1 in distance (see (H1) in Section 5), we see that for almost all integers $N \in [g+1, R-1]$, there exists an ideal $\mathfrak{a} \in \mathcal{R}$ with $\delta(\mathfrak{a}) = N$. However, $\delta_2 = g+1$ implies that there are no infrastructure ideals of distance between 1 and g, and there are generally other integers that do not occur as distance values. In general, there exists a unique ideal $\mathfrak{a}_i \in \mathcal{R}$ with $\delta(\mathfrak{a}_i) \leq N < \delta(\mathfrak{a}_{i+1})$, referred to as the infrastructure ideal *below* N. It can be efficiently computed, along with the "error" $N - \delta(\mathfrak{a}_i)$, using a technique akin to exponentiation.

Remark 3.3. Let $\mathfrak{a}_i = [Q_{i-1}, P_{i-1} + y]$ be an infrastructure ideal, and let $D_i = (Q_{i-1}, P_{i-1})$ be the corresponding reduced affine divisor. Then we can apply Algorithm 6 to D_i to obtain an output $D_{i+1} = ((Q_i, P_i), (\omega^+, \omega^-))$. By Proposition 2 of [8], $D_{i+1} = D_i - \mathrm{div}((y - P_i)/Q_i)$. Thus, the infrastructure ideal $\mathfrak{a}_{i+1} = \mathrm{div}(D_{i+1})$ corresponding to D_{i+1} is the next ideal $\mathfrak{a}_{i+1} = [Q_i, P_i + y]$ in the ordering on $\mathcal{R}$, obtained by applying a baby step to $\mathfrak{a}_i$, and $\mathfrak{a}_i = ((y - P_i)/Q_i)\mathfrak{a}_{i+1}$. Moreover, $\delta(\mathfrak{a}_{i+1}) = -\omega^+ + \delta(\mathfrak{a}_i)$, so $\delta(\mathfrak{a}_{i+1}, \mathfrak{a}_i) = -\omega^+$.

Remark 3.4. Algorithm 3 is exactly the same as the continued fraction algorithm which was described in [18] and [9]. Therefore, $r-1$ successive applications of this algorithm to the trivial divisor $D_1 = (1, 0)$ generates the entire infrastructure.

4. Two maps from $\mathcal{R}$ to $Cl^0(C)$

Mireles Morales in [14] introduced a map from the infrastructure into the degree zero divisor class group of a real hyperelliptic curve C. In this section, we first review the properties of his map. Then we define a second map between the same sets and

show that this new map leads to efficiency improvements in scalar multiplication using balanced divisors.

Mireles Morales' map is defined as follows:

$$\psi : \mathcal{R} \to Cl^0(C), \quad \psi(\mathfrak{a}) = [\mathrm{div}(\mathfrak{a}) - \deg(\mathfrak{a})\infty^-] \ .$$

This is the restriction to the infrastructure of the group homomorphism that sends any non-zero fractional $k[C]$-ideal $\mathfrak{a}$ to the degree zero divisor class $[\mathrm{div}(\mathfrak{a}) - \deg(\mathfrak{a})\infty^-]$.

Mireles Morales proved that $\psi(\mathfrak{a}) = \psi(\mathfrak{b}) + \delta(\mathfrak{b}, \mathfrak{a})[\infty^+ - \infty^-]$ (Proposition 9 of [14]) and hence $\psi(\mathfrak{a}) = \delta(\mathfrak{a})[\infty^+ - \infty^-]$ (Theorem 1 of [14]). Thus, the image of ψ consists precisely of the multiples $[m(\infty^+ - \infty^-)] \in G$ for which m occurs as the distance of some infrastructure ideal.

We wish to relate the infrastructure to the Jacobian in a different manner that highlights the connection to balanced divisors. To that end, we define a second map between these two structures, with the difference that representatives of the classes in the image of this map are all balanced. Specifically, we define

$$\phi : \mathcal{R} \to Cl^0(C), \quad \phi(\mathfrak{a}) = [\mathrm{div}(\mathfrak{a}) + (g - \deg(\mathfrak{a}))\infty^- - D_\infty] \ .$$

Thus, ϕ maps any infrastructure ideal $\mathfrak{a}$ to the class represented by the balanced divisor $(\mathrm{div}(\mathfrak{a}), 0)$. Since balanced representatives are unique, we see that ϕ is injective.

Remark 4.1. For any infrastructure ideal $\mathfrak{a}$, $\phi(\mathfrak{a}) = \psi(\mathfrak{a}) - \lceil g/2 \rceil [\infty^+ - \infty^-]$. In other words, ψ can be interpreted as a shift of ϕ by $\lceil g/2 \rceil$ with respect to the class of $\infty^+ - \infty^-$.

This is essentially the observation made in Remark 5 of [14]. It also implies that the map ψ, as a translation of the injective map ϕ, is itself injective. The following result is an immediate consequence of Remark 4.1 and Theorem 1 of [14].

Proposition 4.2. *For $\mathfrak{a} \in \mathcal{R}$, $\phi(\mathfrak{a}) = (\delta(\mathfrak{a}) - \lceil g/2 \rceil)\ [\infty^+ - \infty^-]$.*

Let B be the set of all classes in $Cl^0(C)$ whose balanced representative is of the form $(D', 0)$, and recall that $G = \langle [\infty^+ - \infty^-] \rangle \subseteq Cl^0(C)$. The next theorem characterizes the image of the map ϕ.

Theorem 4.3. *The image of $\mathcal{R}$ under ϕ is equal to $G \cap B$.*

Proof. The definition of ϕ and Proposition 4.2 immediately yield $\mathrm{Img}(\phi) \subset G \cap B$.

Conversely, let $[D] \in G \cap B$, so there exists $m \in \mathbb{Z}$ such that $[D] = m[\infty^+ - \infty^-]$. Without loss of generality, assume that $0 < m < R$. Let $\mathfrak{a}$ be the infrastructure ideal below $m + \lceil g/2 \rceil$. Then $\delta(\mathfrak{a}) - \lceil g/2 \rceil \leq m < \delta(\mathfrak{b}) - \lceil g/2 \rceil$, where $\mathfrak{b}$ is obtained by applying a baby step to $\mathfrak{a}$. If $\delta(\mathfrak{a}) - \lceil g/2 \rceil = m$, then $[D] \in \mathrm{Img}(\phi)$ and the proof is complete. So suppose to the contrary that $\delta(\mathfrak{a}) - \lceil g/2 \rceil < m < \delta(\mathfrak{b}) - \lceil g/2 \rceil$, and let $n = m + \lceil g/2 \rceil - \delta(\mathfrak{a})$. By Proposition 4.2, $\phi(\mathfrak{a}) = (\delta(\mathfrak{a}) - \lceil g/2 \rceil)[\infty^+ - \infty^-]$, so

$$\begin{aligned} m[\infty^+ - \infty^-] &= [n(\infty^+ - \infty^-) + \phi(\mathfrak{a})] \\ &= [\mathrm{div}(\mathfrak{a}) + n\infty^+ + (g - \deg(\mathfrak{a}) - n)\infty^- - D_\infty] \ . \end{aligned} \tag{4.1}$$

On the other hand, by Remark 3.3 we have $\delta(\mathfrak{b}) - \delta(\mathfrak{a}) = -\omega^+ = g + 1 - \deg(\mathfrak{a})$; thus, $0 < n < g - \deg(\mathfrak{a})$. Therefore, the right hand side of (4.1), which is the balanced representative of $m[\infty^+ - \infty^-]$, is of the form of (D', n) where $n \neq 0$. Thus, $m[\infty^+ - \infty^-] \notin B$ which contradicts our assumption. Hence, $\mathrm{Img}(\phi) = G \cap B$. $\square$

An important consequence of the results above, as already noted in [14], is that the infrastructure discrete logarithm problem (computing $\delta(\mathfrak{a})$ given $\mathfrak{a}$) can be reduced to the discrete logarithm problem in G, and vice versa. Thus, in terms of security, either $\mathcal{R}$ and G can be used for cryptographic purposes. To implement arithmetic in $Cl^0(C)$ using the infrastructure, it is most efficient if one avoids the elements outside of the image of the map ϕ. This raises the obvious question of how frequently these non-images occur.

5. Hole elements

Definition 5.1. The elements in $Cl^0(C) \setminus B$ are called *hole elements* (or *holes* for short). A *hole divisor* is the balanced representative of a hole.

In other words, hole divisors are balanced divisors of the form (D', n) with $n \neq 0$. By Theorem 4.3, the image of ϕ consists precisely of those multiples of $\infty^+ - \infty^-$, i.e. classes in G, that are not holes. Recall that the image of ψ consisted of exactly those scalar multiples of $[\infty^+ - \infty^-]$ whose scalar does not occur as an infrastructure distance. Informally, the map ψ misses distance values in its image, while ϕ misses hole divisors. Note also that by Remark 4.1, $m[\infty^+ - \infty^-]$ is a hole element in G if and only if $m + \lceil g/2 \rceil \pmod R$ does not occur as an infrastructure distance value.

Divisor class addition involving hole divisors generally requires balancing steps after divisor addition and reduction, which incur additional computational cost. Therefore, we are interested in avoiding hole elements in practice. If the number of holes is small, then the chance of avoiding them in our arithmetic is high.

Fontein [6] determined the number of hole elements for the entire collection of infrastructures (arising from all ideal classes) of a global hyperelliptic function field. In essence, he proved that the probability that a divisor class is a hole element is asymptotically equal to $1/q$, and gave an asymptotic error term, for g fixed and $q \to \infty$. Unfortunately, his results cannot be applied to our setting directly, as his count includes all infrastructures, whereas we only consider the principal ideal class.

Recall from Section 3 that the quotient $H = |Cl^0(C)|/R$ is equal to the ideal class number of $k[C]$. Heuristics of Friedman and Washington [7] predict that H is generally small; for most real hyperelliptic curves, it is one (in which case $Cl^0(C) = G$). Since the size of the infrastructure is governed by R, $H = 1$ represents the cryptographically ideal scenario, since $R = |Cl^0(C)|$ is maximal in this situation. Thus, in most cases we expect that the principal infrastructure is in fact the only infrastructure, and Fontein's results would apply. It is an open problem to specialize Fontein's results to the principal infrastructure in the case that H exceeds one.

When the ideal class number is one, applying Fontein's result yields the following.

Remark 5.2. For sufficiently large q and a real hyperelliptic curve C over $k = \mathbb{F}_q$ with ideal class number one, the probability that a divisor class is a hole element is asymptotically equal to $1/q$.

Even if $H > 1$, we expect this probability to be roughly H/q. When $H = 1$, Remark 5.2 implies the following properties, which were stated in [9] as heuristics for the set of infrastructure ideals. Although they are proved through Fontein's results, we label them (H1) and (H2) in keeping with the notation of [9].

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

(H1) $\delta(\mathfrak{a}_{i+1}) - \delta(\mathfrak{a}_i) = 1$ for $2 \leq i \leq |\mathcal{R}|$.

(H2) $\delta(\mathfrak{a} \otimes \mathfrak{b}) = \delta(\mathfrak{a}) + \delta(\mathfrak{b}) - \lceil g/2 \rceil$ for $\mathfrak{a}, \mathfrak{b} \in \mathcal{R} \setminus \{0\}$.

By (3.1), (H1) is equivalent to $\deg(\mathfrak{a}_i) = g$; if $\deg(\mathfrak{a}_i) < g$, then the ideal $\mathfrak{a}_i$, and its corresponding divisor $\mathrm{div}(\mathfrak{a}_i)$, are said to be *degenerate*. (H1) asserts that degenerate ideals (and divisors) are extremely rare for large q. Note also that if $\mathfrak{a}_{i_0}$ is the first degenerate infrastructure ideal in the distance ordering, then for $2 \leq i \leq i_0$, we have $\delta(\mathfrak{a}_i) = g + i - 1$, and hence $\phi(\mathfrak{a}_i) = (\lfloor g/2 \rfloor + i - 1)(\infty^+ \infty^-)$. In particular, $\phi(\mathfrak{a}_i) = [i(\infty^+ - \infty^-)]$ up to the first degenerate ideal when $g = 2, 3$.

(H2) is equivalent to the assumption that reducing the ideal product $\mathfrak{ab}$ to obtain $\mathfrak{a} \otimes \mathfrak{b}$ requires exactly $\lceil g/2 \rceil$ reduction steps. Moreover, (H1) and (H2) imply that there is no need to keep track of relative distances when performing baby steps and giant steps. For specific ideals, relative distances may not be exactly as given in (H1) and (H2). However, in practice, when computing an infrastructure ideal via a succession of baby steps and giant steps — as is the case, for example, for two communicants executing the Diffie-Hellman protocol in the infrastructure — one expects to obtain the same target ideal even if these steps are executed in different sequence and relative distances are not computed, since the same degenerate ideals (i.e. exceptions to (H1) and (H2)) are encountered in the respective computations. Numerical computations of [9] and those found in Section 7 confirm this.

Based on (H1) and (H2), Jacobson et al. obtained improvements to scalar multiplication in the infrastructure in [9]. The map ϕ makes it possible to extend these improvements to G, deriving properties of balanced divisors that are analogous to (H1) and (H2).

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

(H1') $\quad \phi(\mathfrak{a}_{i+1}) - \phi(\mathfrak{a}_i) = [\infty^+ - \infty^-]$ for $2 \leq i \leq |\mathcal{R}|$.
(H2') $\quad \phi(\mathfrak{a} \otimes \mathfrak{b}) = \phi(\mathfrak{a}) + \phi(\mathfrak{b})$ for $\mathfrak{a}, \mathfrak{b} \in \mathcal{R} \setminus \{0\}$.

(H1') follows directly from Proposition 4.2 and shows that the baby step on $\mathcal{R}$ generically corresponds to adding $[\infty^+ - \infty^-]$ in G. To see that the map ϕ is generically additive as asserted by (H2'), note that

$$\begin{aligned}\phi(\mathfrak{a} \otimes \mathfrak{b}) &= (\delta(\mathfrak{a} \otimes \mathfrak{b}) - \lceil g/2 \rceil)[\infty^+ - \infty^-] \\ &= (\delta(\mathfrak{a}) + \delta(\mathfrak{b}) - \lceil g/2 \rceil - \lceil g/2 \rceil)[\infty^+ - \infty^-] \\ &= \phi(\mathfrak{a}) + \phi(\mathfrak{b}).\end{aligned}$$

In contrast, $\psi(\mathfrak{ab}) = \psi(\mathfrak{a}) + \psi(\mathfrak{b}) - \lceil g/2 \rceil [\infty^+ - \infty^-]$, so ψ is generally not additive.

(H1') and (H2') yield several more useful results.

Remark 5.3. Let D_0 be an affine reduced divisor, and $(D_1, (\omega^+, \omega^-)) = \mathrm{red}_{\infty^+}(D_0)$. Then generically, $D_1 \equiv D_0 + (\infty^+ - \infty^-)$.

To see this, recall that in Algorithm 3, $(\omega^+, \omega^-) = (d_0 - g - 1, g + 1 - d_1)$. generically $d_0 = d_1 = g$; thus, $(\omega^+, \omega^-) = (-1, 1)$.

Remark 5.4. The balanced representative of the conjugate of a balanced divisor $D = ((Q, P), 0)$ is generically equal to $\overline{D} = ((Q, -P - h), 0)$ when g is even.

This is clear from Algorithm 8, since generically, $\deg(Q) = g$ and $n_0 = n_1 = 0$. For odd genus, by Algorithm 8, $\overline{D} = ((Q, -P - h), 1)$. Therefore, one balancing step is needed to obtain the balanced representative of $[\overline{D}]$.

Remark 5.5. Generically for two balanced divisors D_1 and D_2, $\lceil g/2 \rceil$ reduction steps, no balancing steps for even genus and one for odd genus are needed to compute the balanced divisor $D_1 \oplus D_2$.

This important observation was already made in [8]. Note that Remark 5.5 implies in particular that in practice the n values of balanced divisor representatives need not be computed, as they will generically be equal to zero. Once again, if Alice and Bob perform the Diffie-Hellman protocol in G using balanced divisor arithmetic without computing any n values and without any balancing (just addition and reduction), they are expected to generate the same shared key divisor class, since they encounter the same hole divisors (i.e. exceptions to Remarks 5.3-5.5) in their respective sequences of reduction steps. Our computations in genus two confirm this; see Section 7.

6. Scalar multiplication on $\mathcal{R}$ and G

In this section, we compare scalar multiplication on the infrastructure $\mathcal{R}$ and the group $G = \langle[\infty^+ - \infty^-]\rangle$ (represented via balanced divisors) on real hyperelliptic curves and on $Cl^0(C)$ (represented via reduced divisors) on imaginary hyperelliptic curves. For the first two cases, we assume the assertions of Section 5 about degenerate infrastructure ideals and hole elements in G, respectively. In particular, we ignore relative distances in $\mathcal{R}$ and n values in G, and perform no "adjustment steps" as described in [9] in the former setting and no balancing in the latter. This implies in particular that the number of Jacobian operations to compute a divisor class $[aD]$, given a scalar a and a base divisor $D = (Q, P)$, is identical for imaginary and real hyperelliptic curves.

We also consider two standard scenarios occurring in discrete logarithm based cryptography. The *fixed base* scenario performs scalar multiplication on a fixed base divisor in the group settings, and generates a reduced principal ideal of a fixed distance in the infrastructure setting. This situation occurs in round 1 of the Diffie-Hellman protocol for example. For imaginary hyperelliptic curves, this base is usually a divisor of the form $P - \infty$ where P is a k-rational point on C. For the group setting on real hyperelliptic curves, we assume that this base divisor is $\infty^+ - \infty^-$ written in balanced form as $((1, 0), \lceil g/2 \rceil + 1)$; the fact that this divisor does not satisfy Remark 5.4 does not matter by our observations at the end of Section 5. For the infrastructure, the fixed base scenario is described in [9]. The *variable base* scenario performs scalar multiplication on an arbitrary divisor, as is the case, for example, in round 2 of the Diffie-Hellman protocol.

Table 1 shows the operation counts for scalar multiplication on the Jacobian of an imaginary hyperelliptic curve ("Imag") as well as the group G ("Real") and the infrastructure ("Infra") of a real hyperelliptic curve. As in [9], we assume a random scalar of some bit length l given in *non-adjacent form*, so we expect that about one third of the signed digits are non-zero. In each setting, we count the number of doubles, adds, baby steps, and the expected number of multiplications in $\mathbb{F}_q$ required in genus two based on the explicit formulas for divisor arithmetic of [13] and [4]. For simplicity, we count squarings and multiplications in $\mathbb{F}_q$ as the same operation. Adds refer to giant steps in the infrastructure and to Jacobian operations in the group G (Algorithm 5) when C is real and in $Cl^0(C)$ when C is imaginary. Doubles are simply adds of two identical divisors. Baby steps have the usual meaning in the infrastructure, and refer to addition or subtraction of $\infty^+ - \infty^-$ (Algorithm 6 or 7) in G when C is real and of a fixed degenerate divisor $P - \infty$ in $Cl^0(C)$ when C is imaginary. For the infrastructure, we use the operation counts of VAR-DIST2 and FIXED-DIST2 as given in Table 1 of [9]. Note that in

the fixed base scenario in G and the imaginary case, the first double can be replaced by a baby step, which is reflected in our counts.

Table 1. Operation counts for scalar multiplication in $Cl^0(C)$ for C imaginary and in G and $\mathcal{R}$ for C real

	Doubles	Adds	Baby Steps	Field Ops ($g = 2$)
Fixed Base, Imag	$l-1$	0	$l/3+1$	$30.66l - 16$
Fixed Base, Real, Even Genus	$l-1$	0	$l/3+1$	$33l - 25$
Fixed Base, Real, Odd Genus	$l-1$	0	$4l/3$	
Fixed Base, Infra	l	0	$l/3$	$33l$
Variable Base, Imag	l	$l/3$	0	$30.66l$
Variable Base, Real, Even Genus	l	$l/3$	0	$40.33l$
Variable Base, Real, Odd Genus	l	$l/3$	$4l/3$	
Variable Base, Infra	l	$l/3$	$\lceil g/2 \rceil$	$40.33l + 6$

Table 1 shows that for even genus, the group operation counts are identical for the group settings on real and imaginary curves; the infrastructure operation count is only very slightly higher. Thus, at the level of just counting baby steps and giant steps, performance of scalar multiplication exhibits essentially equal performance in all three settings under consideration. For odd genus, we observe that the performance for the group setting on real curves is expected to be slower than the other two, due to the single baby step required for balancing after each divisor class addition or double.

For genus two, we expect G to be slightly faster than $\mathcal{R}$ at the level of field operations, and the imaginary case to still be the fastest, due to the higher costs of divisor arithmetic in the real case. We investigate practical performance for genus two in the next section.

7. Numerical results

We implemented the Diffie-Hellman protocol in the Jacobian and infrastructure of genus two real hyperelliptic curves, using the fixed and variable base scalar multiplication algorithms described in the previous section. We employed the explicit formulas from [4] for divisor arithmetic in both cases, in place of the general-purpose formulas given in Subsection 2.1. For comparison purposes, we also implemented Diffie-Hellman in the Jacobian of genus two imaginary hyperelliptic curves, using the explicit formulas from [13]. In the fixed base scenario, we used a base divisor of the form $P - \infty$ with P a k-rational point on the curve; this is the closest analogy to performing baby steps in the real model. We used the affine representation of divisors and applied the standard isomorphic transformations to the defining equations of our curves [13, 4] to minimize the number of non-zero coefficients, thereby maximizing the efficiency of the curve arithmetic.

We used the computer algebra library NTL [17] for finite field and polynomial arithmetic and the GNU `C++` compiler version 4.4.5. The computations described below were performed on an Intel Core i72600 3.4 GHz computer running Linux.

All three protocols were implemented using genus 2 curves defined over $\mathbb{F}_p$ and $\mathbb{F}_{2^n}$. The finite field was chosen so that the size of the infrastructures and Jacobians under consideration were roughly 2^{160}, 2^{224}, 2^{256}, 2^{384}, and 2^{512}. Thus, for $\mathbb{F}_{2^n}$, we used $n \in \{80, 112, 128, 192, 256\}$, and for $\mathbb{F}_p$, we chose a random prime p such that

p^2 had the required bit length. These settings offer 80, 112, 128, 192, and 256 bits of security, respectively, for cryptographic protocols based on the corresponding discrete logarithm problem. NIST [1] currently recommends these five levels of security for key establishment in U.S. Government applications.

For each finite field, we randomly selected 100,000 curves and executed Diffie-Hellman once for each curve. The random scalars used had 160, 224, 256, 384, and 512 bits, respectively, ensuring that the number of bits of security provided corresponds to the five levels recommended by NIST (again, considering only generic attacks). In order to provide a fair comparison between the three algorithms, the same sequence of random exponents was used for each run of the key agreement protocol. As the algorithms in the real model rely on our heuristic assumptions to ensure correctness, we also checked that the resulting key divisors were in fact equal; across all our computations, this was always the case.

Tables 2 and 3 contain the average CPU time in milliseconds for each of the three algorithms. The headings "Imag", "Real" and "Infra" have the same meaning as for Table 1. The times required to generate domain parameters are not included in these timings, as domain parameter generation is a one-time computation. As predicted by our analysis, the algorithms using the Jacobian in the real model slightly out-perform those using the infrastructure. The imaginary model is still the fastest of all, but by no more than approximately 1.7 milliseconds for q even and 1.1 milliseconds for q odd.

TABLE 2. Scalar multiplication and key exchange timings over $\mathbb{F}_p$ (in milliseconds).

Security Level (in bits)	Fixed Base			Variable Base			Total Diffie-Hellman		
	Imag	Real	Infra	Imag	Real	Infra	Imag	Real	Infra
80	1.114	1.162	1.173	1.227	1.374	1.383	2.341	2.536	2.556
112	1.648	1.732	1.749	1.817	2.033	2.022	3.465	3.766	3.771
128	2.288	2.388	2.404	2.525	2.811	2.818	4.813	5.199	5.223
192	4.397	4.610	4.634	4.802	5.353	5.364	9.200	9.963	9.997
256	6.526	6.799	6.813	7.117	7.856	7.864	13.643	14.655	14.677

TABLE 3. Scalar multiplication and key exchange timings over $\mathbb{F}_{2^n}$ (in milliseconds).

Security Level (in bits)	Fixed Base			Variable Base			Total Diffie-Hellman		
	Imag	Real	Infra	Imag	Real	Infra	Imag	Real	Infra
80	3.119	3.351	3.408	3.540	4.079	4.096	6.660	7.430	7.504
112	1.767	1.844	1.854	1.931	2.126	2.143	3.698	3.970	3.996
128	2.055	2.131	2.145	2.231	2.443	2.450	4.286	4.574	4.595
192	4.545	4.781	4.807	4.937	5.459	5.532	9.482	10.240	10.339
256	8.393	8.969	8.996	9.119	10.213	10.223	17.512	19.182	19.219

8. CONCLUSIONS AND FUTURE DIRECTIONS

Our analysis and numerical experiments show that Mireles Morales' claim that the Jacobian of a real hyperelliptic curve is more efficient than the infrastructure

for cryptographic applications is true for even genus curves. According to Table 1, Jacobian arithmetic needs more baby steps than infrastructure when the genus of the curve is odd. It should be possible to mirror and interpret the scalar multiplication algorithms for the Jacobian described here in the infrastructure. However, this will not result in anything more efficient, and it seems more natural to describe the algorithms in the Jacobian.

On the other hand, our analysis suggests that scalar multiplication in the infrastructure may be faster than in the Jacobian of an odd-genus real hyperelliptic curve due to the fact that each Jacobian operation requires generically at least one baby step for balancing. Our current work includes a more careful investigation of this case, especially for genus three. One possible approach to closing the performance gap is to apply the same trick from the infrastructure described in [9] to the Jacobian to reduce the number of baby steps. Another idea for reducing the number of balancing steps required for odd genus, suggested by Galbraith, is to work with divisors of degree $g+1$ instead of fully reducing after each operation. The advantage is that no balancing steps would be required, but this approach would incur the additional cost of performing divisor arithmetic with higher-degree operands. A careful analysis of these approaches will be required to determine which will work best in practice.

The current state-of-the-art is that scalar multiplication on imaginary and real hyperelliptic curves of even genus both require exactly the same number of operations on divisors, with no adjustment or balancing steps required in practice in the real case. The only remaining difference in performance is in the costs of the basic divisor operation. Baby steps in genus two require five fewer field operations than adding the divisor of a point in the imaginary case, but additions and doublings require four more field multiplications in the real case. However, there has been much less work on explicit formulas for divisor arithmetic on real model. It is conceivable that more attention to this setting may result in a sufficient decrease in the number of field multiplications required per operation, so that the real model will achieve the same or better performance compared to the imaginary model, and become an accepted alternative for practical applications.

References

[1] E. Barker, W. Barker, W. Polk and M. Smid, Recommendation for key management - part 1: General (revised), NIST Special Publication 800-57, 2007.

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

[3] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Trans. Inf. Theory*, **22** (1976), 472–492.

[4] S. Erickson, M. J. Jacobson, Jr. and A. Stein, Explicit formulas for real hyperelliptic curves of genus 2 in affine representation, *Adv. Math. Commun.*, **5** (2011), 623–666.

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

[6] F. Fontein, Holes in the infrastructure of global hyperelliptic function fields, preprint, arXiv:0911.4346

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

[8] S. D. Galbraith, M. Harrison and D. J. Mireles Morales, Efficient hyperelliptic curve arithmetic using balanced representation for divisors, in *Algorithmic Number Theory - ANTS 2008 (Berlin)*, Springer, 2008, 342–356.

[9] M. J. Jacobson, Jr., R. Scheidler and A. Stein, Cryptographic protocols on real hyperelliptic curves, *Adv. Math. Commun.*, **1** (2007), 197–221.

[10] M. J. Jacobson, Jr., R. Scheidler and A. Stein, Fast arithmetic on hyperelliptic curves via continued fraction expansions, in *Advances in Coding Theory and Cryptology* (eds. T. Shaska, W.C. Huffman, D. Joyner and V. Ustimenko), World Scientific Publishing, 2007, 201–244.
[11] M. J. Jacobson, Jr., R. Scheidler and A. Stein, Cryptographic aspects of real hyperelliptic curves, *Tatra Mountains Math. Publ.*, **40** (2010), 1–35.
[12] N. Koblitz, Hyperelliptic cryptosystems, *J. Cryptology*, **1** (1989), 139–150.
[13] T. Lange, Formulae for arithmetic on genus 2 hyperelliptic curves, *Appl. Algebra Eng. Commun. Comput.*, **15** (2005), 295–328.
[14] D. J. Mireles Morales, An analysis of the infrastructure in real function fields, Cryptology eprint archive no. 2008/299, 2008.
[15] R. Scheidler, J. A. Buchmann and H. C. Williams, A key exchange protocol using real quadratic fields, *J. Cryptology*, **7** (1994), 171–199.
[16] R. Scheidler, A. Stein and H. C. Williams, Key-exchange in real quadratic congruence function fields, *Des. Codes Crypt.*, **7** (1996), 153–174.
[17] V. Shoup, NTL: A Library for doing Number Theory (version 5.4.2), `http://www.shoup.net`, 2008.
[18] A. Stein, Explicit infrastructure for real quadratic function fields and real hyperelliptic curves, *Glas. Mat. Ser. III*, **44(64)** (2009), 89–126.

Received March 2014; revised September 2014.

E-mail address: jacobs@cpsc.ucalgary.ca
E-mail address: mrezaira@ucalgary.ca
E-mail address: rscheidl@ucalgary.ca