MATHEMATICS OF COMPUTATION
Volume 74, Number 252, Pages 1983–2005
S 0025-5718(05)01758-8
Article electronically published on March 31, 2005

POINT COUNTING ON PICARD CURVES IN LARGE CHARACTERISTIC

MARK BAUER, EDLYN TESKE, AND ANNEGRET WENG

Abstract. We present an algorithm for computing the cardinality of the Jacobian of a random Picard curve over a finite field. If the underlying field is a prime field $\mathbb{F}_p$, the algorithm has complexity $O(\sqrt{p})$.

1. Introduction

Let p be a prime, $p \neq 2, 3$, and let $\mathbb{F}_q$ be a finite field of characteristic p with q elements. A principal problem in arithmetic geometry is to determine the group order of $J_C(\mathbb{F}_q)$, the set of $\mathbb{F}_q$-rational points on the Jacobian of a randomly chosen curve C defined over $\mathbb{F}_q$. In this paper we address this problem in the case where C is a Picard curve.

A Picard curve defined over $\mathbb{F}_q$ is a curve of genus 3 admitting an affine model given by an equation of the form

$$y^3 = f(x),$$

where $f(x) \in \mathbb{F}_q[x]$ is a polynomial of degree 4. If the characteristic p is small, there exists an efficient point counting algorithm using p-adic methods [GG03]. It is also possible to construct special curves defined over large characteristic using complex multiplication [KW]. There is no satisfactory algorithm for point counting on Picard curves in large characteristic. (Pila's generalization of Schoof's algorithm to general abelian varieties [Pil90], albeit polynomial-time, is of little practical use.) On the other hand, efficient arithmetic for Picard curves in any characteristic is available, see, e.g., [Bau04, FO04]. Compared to the arithmetic for hyperelliptic curves of genus three, it is slower just by a factor somewhere between two and three.

Another strong interest in determining the group order of $J_C(\mathbb{F}_q)$ comes from the applications of Picard curves in public-key cryptography: The Jacobian of a Picard curve can be used to implement discrete logarithm-based cryptography such as the Diffie-Hellman key exchange protocol [DH76]. Knowing (the largest prime factor dividing) the group order is necessary to be able to guarantee the desired level of security against the Pohlig-Hellman attack [PH78]. Note that just as with the discrete logarithm problem (DLP) in the group of points on an elliptic curve, for the DLP in the Jacobian of a Picard curve the best currently available algorithms have fully exponential running time. While in the elliptic case, the best algorithm is the parallelized Pollard rho method [Pol78, vOW99], Thériault's index calculus

Received by the editor December 22, 2003 and, in revised form, October 10, 2004.
2000 *Mathematics Subject Classification.* Primary 14H45. Secondary 11Y16.
Key words and phrases. Picard curves.

algorithm [Thé03] for the DLP in the Jacobian of hyperelliptic curves of genus 3 can be extended to Picard curves. This algorithm solves the DLP in time $O(q^{10/7})$, thus asymptotically improving on the $O(q^{3/2})$ running time for the Pollard rho method. But due to the larger O-constants for Thériault's algorithm, its effect on the security of Picard curves over fields of cryptographically relevant sizes (e.g., $q \approx 2^{55}$ for a 165-bit Jacobian) is at most marginal.

Using the Hasse-Weil bound for curves, it is possible to deduce that the group order $\#J_C(\mathbb{F}_q)$ of the Jacobian of a genus 3 curve lies in an interval of size $O(q^{5/2})$. Thus, $\#J_C(\mathbb{F}_q)$ can be computed using the Pollard kangaroo method [Pol78] in $O(q^{5/4})$ operations in the Jacobian. Stein and Teske have given better estimates for the group orders of hyperelliptic genus 3 curves and derived an algorithm of complexity $O(q)$ [ST02a]. They were able to construct a hyperelliptic curve of genus 3 with group order of size 10^{29} [ST02b]. This idea has recently been adapted to Picard curves by Stein and Scheidler [SS]. Their method also has running time $O(q)$, but no experimental results were given. While there also exist algorithms (see, e.g., [Pil90]) that can theoretically compute the zeta function of a curve, and hence the cardinality of its Jacobian, in polynomial time they are of little practical use.

In this paper, we describe a divide-and-conquer approach which takes time and space $O(\sqrt{q})$. It is based on the following idea: Let ζ_3 be a primitive cube root of unity. Then over $\mathbb{Z}[\zeta_3]$, the characteristic polynomial of the Frobenius $F(t)$ splits into a cubic polynomial $g(t)$ and its complex conjugate. Exploiting properties of the Frobenius endomorphism on the Jacobian, we narrow the possible choices for the coefficients of $g(t)$. Using a baby step–giant step type algorithm, we then search the list of remaining candidates by checking whether a random divisor of the Jacobian is annihilated by $g(1)$.

In Section 2, we first deduce some easy facts about Picard curves that in particular enable us to determine the group order of the Jacobian of a Picard curve modulo 2 and 3. Then in Section 3 we focus on the L-polynomial of a Picard curve and show that it has a very special form. From this, in Section 4 we derive our baby step–giant step type algorithm for determining the group order. This algorithm takes $O(\sqrt{q})$ operations in the Jacobian and has to store $O(\sqrt{q})$ divisors. In most cases the output of the algorithm will also enable us to determine the complete L-polynomial of the curve C, which encodes the numbers $\#C(\mathbb{F}_{q^k})$ for all $k \in \mathbb{N}$.

If $q = p^n$ with $n \geq 2$, this algorithm can be improved by first determining the Hasse-Witt matrix of the Picard curve. We describe this approach in Section 5. Running times are summarized in Section 6, which is followed by computational examples in Section 7. We give an example of a Picard curve defined over a field of size $5 \cdot 10^{12}$ which has a 39-digit prime group order. Furthermore, we found a Picard curve defined over $\mathbb{F}_{p^3}$ where p is of size 10^6 and whose group order is divisible by a prime of size 10^{54}. Lastly, we discuss some possible further speed-ups. We conclude in Section 9, and indicate that our new method can also be applied to a special class of hyperelliptic genus 3 curves.

2. Basic facts

For basic definitions on curves and function fields see [Sti93].

Let C be a Picard curve C over $\mathbb{F}_q$ given by the generic equation

$$y^3 = f(x) = x^4 + f_3x^3 + f_2x^2 + f_1x + f_0. \tag{2.1}$$

By completing the square, we can assume that $f_3 = 0$. The zeta function $\zeta(s, C)$ of C can be written as

$$\zeta(s,C) = \frac{L(t,C)}{(1-t)(1-qt)},$$

where $L(t, C)$ is the L-polynomial of the curve. The L-polynomial is of degree $2g$, where $g = 3$ is the genus of the curve, and encodes information on the group order of its Jacobian J_C. More precisely, $\#J_C(\mathbb{F}_q) = L(1, C)$. Let π_q be the Frobenius endomorphism of C and $F_{\pi,C}(t)$ the characteristic polynomial of π_q on the Tate module $T_\ell(J_C) \otimes \mathbb{Q}_\ell$. Then $F_{\pi,C}(t) = t^{2g}L(1/t, C)$. For simplicity, we write $F(t)$ instead of $F_{\pi,C}(t)$ if the reference to the curve is clear. Similarly, we write $L(t)$ instead of $L(t, C)$. We obviously have $\#J_C(\mathbb{F}_q) = F(1)$. Now let $N_r = \#C(\mathbb{F}_{q^r}) - (q^r + 1)$, and we write $L(t) = \sum_{i=0}^{2g} c_i t^i$. Then the coefficients c_i are determined by the recursion $c_0 = 1, c_1 = N_1$,

$$c_i = \frac{1}{i}\left(N_i + \sum_{\substack{j+k=i\\ 1\le j,k\le i-1}} c_k N_j\right), \quad i = 2,\dots,g,$$

and $c_{2g-i} = q^{g-i}c_i$ for $i = 0, \dots, g-1$. Thus for the L-polynomial of our genus 3 curve we only need three coefficients of the L-polynomial or, equivalently, the number of points $\#C(\mathbb{F}_{q^r})$ for $r = 1, 2, 3$. In terms of the N_r we have

$$c_1 = N_1, \quad c_2 = \frac{1}{2}\left(N_2 + N_1^2\right), \quad \text{and} \quad c_3 = \frac{1}{3}\left(N_3 + \frac{3}{2}N_2N_1 + \frac{1}{2}N_1^3\right). \tag{2.2}$$

Lemma 2.1. *Let C be a Picard curve over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Then the L-polynomial splits over $\mathbb{Q}$. More precisely, the group order of the Jacobian is divisible by $q + 1$.*

Proof. Since every element in $\mathbb{F}_q$ is the unique third power of an element in $\mathbb{F}_q$ we have $\#C(\mathbb{F}_q) = q + 1$. Similarly, $\#C(\mathbb{F}_{q^3}) = q^3 + 1$. Thus, $N_1 = N_3 = 0$. Now suppose that $N_2 = a$. Then a is even and by (2.2),

$$L(t) = (qt^2 + 1)(q^2t^4 + (a/2)t^2 - qt^2 + 1).$$

Substituting $t = 1$ gives the desired result. □

In the following we will restrict ourselves to the case $q \equiv 1 \pmod 3$. In this case there is an action of $\mathbb{Z}[\zeta_3]$ on the Jacobian of C given by the automorphism

$$(x, y) \mapsto (x, by), \qquad \text{where } b \neq 1 \in \mathbb{F}_q \text{ and } b^3 = 1,$$

on the curve. A more detailed discussion of the L-polynomial in this case is given in Section 3.

2.1. The $(1-\zeta_3)$-torsion points.

The defining equation of the curve immediately gives us some information on the 3^k-torsion part of the Jacobian. First note that the points on C with vanishing y-coordinate correspond to $(1-\zeta_3)$-torsion points of the Jacobian. Now, $J_C[1-\zeta_3] \simeq (\mathbb{Z}/3\mathbb{Z})^3$, and, if we consider the images under the natural inclusion of the curve into the Jacobian of any three distinct points with vanishing y-coordinate, they will generate the entire $(1-\zeta_3)$-torsion group.

Lemma 2.2. *Let $q \equiv 1 \pmod 3$ and let $y^3 = f(x)$ be the defining equation of the Picard curve.*

(1) *Suppose $f(x)$ splits completely over $\mathbb{F}_q$. Then $\#J_C(\mathbb{F}_q) \equiv 0 \pmod{27}$.*
(2) *Suppose $f(x)$ splits into three factors over $\mathbb{F}_q$. Then $\#J_C(\mathbb{F}_q) \equiv 0 \pmod 9$.*
(3) *Suppose $f(x)$ splits into a factor of degree 3 and a factor of degree 1, or into two factors of degree 2. Then $\#J_C(\mathbb{F}_q) \equiv 0 \pmod 3$.*
(4) *Suppose $f(x)$ is irreducible over $\mathbb{F}_q$. Then $\#J_C(\mathbb{F}_q) \equiv 1 \pmod 3$.*

Proof.

(1) In this case $J_C[1-\zeta_3]$ is defined over $\mathbb{F}_q$. Hence $(\mathbb{Z}/3\mathbb{Z})^3 \le J_C(\mathbb{F}_q)$.
(2) The two $(1-\zeta_3)$-torsion points arising from the roots of $f(x)$ are linearly independent. Hence, $(\mathbb{Z}/3\mathbb{Z})^2 \le J_C(\mathbb{F}_q)$.
(3) In this case, we only know that $J_C(\mathbb{F}_q)$ contains one nontrivial $(1-\zeta_3)$-torsion point.
(4) By Section 3, (3.2), $\#J_C(\mathbb{F}_q)$ is the norm of an element in $\mathbb{Z}[\zeta_3]$. Hence, $\#J_C(\mathbb{F}_q) \equiv 0, 1 \pmod 3$. Furthermore, if $f(x)$ is irreducible there are no $(1-\zeta_3)$-torsion points in $J_C(\mathbb{F}_q)$. Since 3 ramifies in $\mathbb{Z}[\zeta_3]$, i.e., $3 = -\zeta_3^2(1-\zeta_3)^2$, there are also no 3-torsion points in $J_C(\mathbb{F}_q)$. Thus, $\#J_C(\mathbb{F}_q) \equiv 1 \mod 3$. □

2.2. **The 2-torsion points.** As with the 3-torsion, we can derive information about the 2-torsion that is present in the Jacobian. Since over the algebraic closure $J_C[2] \cong (\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z})^3$, it remains to determine which of these elements of order 2 are defined over the ground field.

An element of order 2 in the Jacobian of a Picard curve corresponds to a divisor in one of two specific forms.

The first class of divisors having order 2 comes from bitangents to the curve. These are lines which intersect the Picard curve in two points in the affine plane, each with multiplicity 2. Let $y - ax - b$ represent such a line with P_1 and P_2 being the two points of intersection. The principal divisor associated to $y - ax - b$ is $2P_1 + 2P_2 - 4P_\infty$, so $P_1 + P_2 - 2P_\infty$ is a divisor of order dividing 2. Furthermore, it is not a principal divisor because it is distinguished (see [Bau04]). Thus, it corresponds to an element of exact order 2. From the theory of θ-characteristics, we know there are precisely 28 such bitangents for a Picard curve, one of which intersects only at infinity (see [GH78] for details). This yields 27 distinct points of order 2 in the Jacobian.

The second class of divisors yielding elements of order 2 are divisors of the form $P_1 + P_2 + P_3 - 3P_\infty$, where the P_i are unramified finite points on the curve that are not conjugate under the action of ζ_3. By counting the difference, we conclude there are precisely 36 such divisors. These are the only admissible forms for a divisor of order 2. To prove this, we rely heavily on the isomorphism between the divisor class group and the ideal class group; for details we refer to [Bau04].

Lemma 2.3. *Let $I = [s, s'(u+y), v + wy + y^2]$ (where $s, s', u, v, y \in \mathbb{F}_q[x]$) be a (nontrivial) reduced ideal of $\mathbb{F}_q[C] = \mathbb{F}_q[x,y]/(y^3 - f)$ and such that I^2 is principal. Then the factorization of the ideal I contains no ramified primes or primes which are conjugate under the action ζ_3.*

Proof. Consider the basis for I^2 given by $(d)[S, S'(U+y), V + Wy + y^2]$, where $d = \gcd(f, s')$ and $S, S', U, V, W \in \mathbb{F}_q[x]$ (note that by using the squaring formulas in [Sch01] one can verify $S \neq 1$). This ideal is principal if and only if the ideal given by $I' = [S, S'(U+y), V + Wy + y^2]$ is principal. Let α be the principal generator for this ideal, and write $\alpha = a + by + cy^2$ with $a, b, c \in \mathbb{F}_q[x]$. Since the degrees of α and I' must agree, we have

$$\deg S + \deg S' = \max\{3 \deg a, 3 \deg b + 4, 3 \deg c + 8\}.$$

Since the degree of I' is at most twice the degree of I, we have $c = 0$ and $b \in \mathbb{F}_q$. Since $\alpha \in I'$, if $S' \neq 1$, then $b = 0$, and α is a multiple of S. Thus, $\deg S + \deg S' \geq 3 \deg S$, and hence $\deg S' > \deg S$. However, this is a contradiction since $S' \mid S$. Therefore, $S' = 1$.

Now suppose the factorization of I contains two nonramified prime ideals that are conjugates under the action of ζ_3. Then $s' \neq 1$ and $\gcd(s', f) \neq s'$. Thus, our canonical basis for I' has $S' \neq 1$ (again, see [Sch01]), contradicting the fact that $S' = 1$.

Now assume the factorization of I contains at least one ramified prime, say $\mathfrak{P}$. If $\mathfrak{P}^2$ does not contain I, then $\mathfrak{P}^2 || I^2$, and hence $S' \neq 1$. Any ramified prime must therefore divide I with multiplicity exactly 2, since multiplicity 3 would imply that I is not primitive and hence not reduced. In this case, the degree of I' is bounded by 3, since $\mathfrak{P}^4 = (u)\mathfrak{P}$ for some irreducible polynomial $u \in \mathbb{F}_q[x]$. Consequently, $I' = \langle S \rangle$ where $\deg S = 1$ because the generator α now has degree ≤ 3. Then $I' = \mathfrak{P}$, which implies $I = \mathfrak{P}^2$, which is in fact a (nonprincipal) distinguished ideal. However, I has order dividing 3 since $(\mathfrak{P}^2)^3 = (\mathfrak{P}^3)^2 = \langle u^2 \rangle$. This forces I to be principal, a contradiction. □

Corollary 2.1. *If I is a reduced ideal such that I^2 is principal, then*

$$I = [s, u+y, v+y^2] = \langle s, u+y \rangle \quad \text{with} \deg s \in \{2, 3\}.$$

Proof. The fact that I has this form follows from the previous lemma, while the second equality is just a simple observation. The degree statement about s follows from the fact that if $\deg s = 1$, then I^2 is also distinguished and hence not principal. □

We now develop an explicit criterion for determining whether an ideal has order 2. This will be used to develop an algorithm for completely determining the 2-torsion in the ideal class group (and hence the Jacobian).

Lemma 2.4. *Let $I = \langle s, u+y \rangle$ be a reduced ideal. Then I has order 2 if and only if s^2 divides the norm $\mathrm{N}_{\mathbb{F}_q(C)/\mathbb{F}_q(x)}(u+y) = u^3 + f$. Moreoever, I is a distinguished ideal.*

Proof. Let $I = \langle s, u+y \rangle$ be a reduced ideal. A quick check verifies that $u+y$ is the element of minimal norm in I. Hence $I' = \langle u+y \rangle I^{-1}$ is the unique distinguished ideal in the inverse class of I and $I' = \langle (u^3+f)/s, u+y \rangle$. Therefore, $I\,I' = \langle u+y \rangle$.

If $s^2 \mid (u^3+f)$, then by a simple degree argument we see that $I' = I$ since u^3+f is a constant multiple of s^2. This proves the reverse implication. It also proves that I is distinguished.

Now assume that I is an ideal of order 2. Hence, $I^2 = \langle s^2, U+y \rangle = [s^2, U+y, V+y^2]$ with $\deg U < 2 \deg s$, and this ideal is principally generated by some element α. As in the previous arguments, it is easy to see that $\alpha = a + by$ where $b \in \mathbb{F}_q$. Hence we can write

$$\alpha = b(U+y) + rs^2, \qquad \text{with } k \in \mathbb{F}_q \text{ and } r \in \mathbb{F}_q[x].$$

Also, r must be zero since $\deg U < 2\deg s$. This implies that we can take $\alpha = U + y$. Now $\deg\alpha = \max\{3\deg U, 4\} = 2\deg s$. Therefore $\deg U < \deg s$, and thus $U = u$ since $U + y \in I$ implies $U \equiv u \pmod{s}$ and $\deg u < \deg s$. Therefore $I^2 = \langle s^2, u+y\rangle$, which implies that $s^2 \mid \mathrm{N}_{\mathbb{F}_q(C)/\mathbb{F}_q(x)}(u+y)$. □

Thus, if $s, u \in \mathbb{F}_q[x]$ with $\deg s \in \{2,3\}$ and $\deg u < \deg s$ and such that $s^2 \mid \mathrm{N}_{\mathbb{F}_q(C)/\mathbb{F}_q(x)}(u+y)$, then $I = \langle s, u+y\rangle$ is a distinguished ideal of order 2. We handle the case $\deg s = 2$ first.

Recall the generic equation (2.1) of our curve, with $f_0, f_1, f_2 \in \mathbb{F}_q$ and $f_3 = 0$. Let $s = x^2 + s_1x + s_0$ and $u = u_1x + u_0$ with $s_0, s_1, u_0, u_1 \in \mathbb{F}_q$. By comparing degrees, we see that if $s^2 | (u^3 + f)$, then in fact $s^2 = u^3 + f$. Consequently, by comparing coefficients,

$$\begin{aligned}2s_1 &= u_1^3\ ,\\ 2s_0 + s_1^2 &= f_2 + 3u_0u_1^2\ ,\\ 2s_0s_1 &= f_1 + 3u_0^2u_1\ ,\\ s_0^2 &= f_0 + u_0^3\ .\end{aligned}$$

Solving for the s_i's, we obtain

$$\begin{aligned}s_1 &= u_1^3/2\ ,\\ s_0 &= (-u_1^6/4 + f_2 + 3u_0u_1^2)/2\ ,\\ 0 &= -u_1^9 + 4f_2u_1^3 + 12u_0u_1^5 - 8f_1 - 24u_0^2u_1\ ,\\ 0 &= -8f_2u_1^6 - 24u_0u_1^8 + 16f_2^2 + 144u_0^2u_1^4 + u_1^{12} + 96f_2u_0u_1^2 - 64f_0 - 64u_0^3\ .\end{aligned}$$

Doing a similar calculation for the case $\deg s = 3$, that is, $s = x^3 + s_2x^2 + s_1x + s_0$ and $u = u_2x^2 + u_1x + u_0$ $(s_i, u_i \in \mathbb{F}_q)$, we get

$$\begin{aligned}s_2 &= (3u_1u_2^2)/(2u_2^3)\ ,\\ s_1 &= (3u_1^2u_2/4 + 1 + 3u_0u_2^2)/(2u_2^3)\ ,\\ s_0 &= (-u_1^3/8 - 3u_1/(2u_2) - 3u_0u_1u_2/2)/(2u_2^3)\ ,\\ 0 &= -3u_1^4u_2^2 - 120u_1^2u_2 + 24u_0u_1^2u_2^3 + 16 + 96u_0u_2^2 - 48u_0^2u_2^4 - 64f_2u_2^3\ ,\\ 0 &= -3u_2^2u_1^5 - 40u_2u_1^3 + 24u_2^3u_1^3u_0 - 48u_1 - 96u_2^2u_1u_0 - 48u_2^4u_0^2u_1 - 64f_1u_2^4\ ,\\ 0 &= u_1^6u_2^2 + 24u_1^4u_2 - 24u_0u_1^4u_2^3 + 144u_1^2\\ &\quad -288u_0u_1^2u_2^2 + 144u_0^2u_1^2u_2^4 - 256f_0u_2^5 - 256u_0^3u_2^5\ .\end{aligned}$$

Given explicit values for the f_i's, we can solve for the u_i's using Gröbner bases very efficiently, and then substitute into the appropriate values for the s_i's. For the size of the fields we considered (and fields that were much larger), Magma [Mag] was able to perform these computations in a neglible amount of time. Thus, we can explicitly compute all the elements of order 2 and hence determine the precise number of 2-torsion elements in the Jacobian.

For the case $\deg s = 2$, we have found all of the relevant bitangents. If $\deg s = 3$, we now have a geometric characterization by noting that $\deg u = 2$ for the corresponding u, and hence we are finding parabolas that intersect the curve in three finite points with multiplicity 2.

3. More on L-polynomials of Picard curves

In this section we further investigate the L-polynomial of a Picard curve over a field $\mathbb{F}_q$ with $q \equiv 1 \pmod 3$. We show that the characteristic polynomial of the Frobenius $F(t)$ splits in $\mathbb{Z}[\zeta_3] = \mathbb{Z} + \frac{-1+\sqrt{-3}}{2}\mathbb{Z}$ in two (not necessarily irreducible) factors $g(t)$ and $\overline{g}(t)$, where $\overline{g}(t)$ is obtained from $g(t)$ by applying the complex conjugation to the coefficients. Moreover, for the vast majority of curves over $\mathbb{F}_q = \mathbb{F}_{p^n}$, we have

$$g(t) = t^3 - a_1 t^2 + \overline{a_1}\pi^n t - \pi^n q,$$

where $\pi \in \mathbb{Z}[\zeta_3]$ and such that $\pi\overline{\pi} = p$, and $v_\pi(a_1) = v_{\overline{\pi}}(a_1) = 0$.

Recall that C has an automorphism of order 3 defined over $\mathbb{F}_q$ given by $(x, y) \mapsto (x, by)$ where $b \neq 1 \in \mathbb{F}_q$ is an element such that $b^3 = 1$. It extends to an automorphism of the Jacobian. Hence, $\mathbb{Z}[\zeta_3] \subseteq \operatorname{End}(J_C)$ or more precisely, $\mathbb{Z}[\zeta_3]$ is contained in the center of the endomorphism ring. Now, using [Tat66] we get three possibilities:

(1) J_C is absolutely simple. Here $\operatorname{End}(J_C) \otimes \mathbb{Q}$ is a CM field of degree 6. It is a composite of a totally real field of degree 3 and $\mathbb{Q}[\zeta_3]$.
(2) J_C is over $\mathbb{F}_q$ isogenous to the product of an abelian variety of dimension 2 and an elliptic curve. They are both simple over $\mathbb{F}_q$. Further, we see that the automorphism of order 3 acts on each factor. Hence, $F(t)$ splits into four factors over $\mathbb{Q}[\zeta_3]$: two factors of degree 2 and two linear factors.
(3) J_C is isogenous to the product of three elliptic curves. Arguing as in (2), we see that $F(t)$ splits completely over $\mathbb{Q}[\zeta_3]$.

The first case is the most frequent case. To see this, note that there are $O(q^2)$ isomorphism classes of Picard curves over $\mathbb{F}_q$ since the moduli space has dimension 2. There are at most six isomorphism classes of elliptic curves with complex multiplication (CM) by $\mathbb{Z}[\zeta_3]$. Therefore, the number of isomorphism classes of Picard curves whose Jacobian is isogenous to the product of elliptic curves can be bounded by a constant not depending on q. Moreover, the moduli space of abelian surfaces with CM by $\mathbb{Z}[\zeta_3]$ is one dimensional: Every abelian surface is isogenous to the Jacobian of a hyperelliptic curve C. If J_C has CM by $\mathbb{Z}[\zeta_3]$, then C can be written in the form

$$y^2 = x^6 + ax^3 + b,$$

and $j = b/a^3$ determines the isomorphism class for $a \neq 0$.

From now on we restrict ourselves to the first case. For the other cases, see Remark 3.1.

Let K be the CM field $\operatorname{End}(J_C) \otimes \mathbb{Q}$. The points on the Jacobian form an $\mathcal{O}$-module where $\mathcal{O} \subseteq \mathcal{O}_K$ is an order in K containing the third roots of unity. Let w be an element in $\mathcal{O}_K$ corresponding to the Frobenius endomorphism π_q of C, i.e., $wP = \pi_q(P)$ for all $P \in J_C(\overline{\mathbb{F}_q})$ where the multiplication on the left-hand side is the module multiplication. This property determines w uniquely. We have $F(w) = 0$. We set $w_1 = w$ and denote by $w_2, \ldots, w_6$ the conjugates of w over $\mathbb{Q}$. It is well known that $w_i\overline{w_i} = q$ and $w_i + \overline{w_i}$ is a totally real element (see, e.g., [Wat69]). Since $F(t)$ splits over $\mathbb{Q}(\zeta_3)$, we can write

$$F(t) = (t^3 - a_1 t^2 + a_2 t - a_3)(t^3 - \overline{a_1} t^2 + \overline{a_2} t - \overline{a_3}), \qquad a_i \in \mathbb{Z}[\zeta_3].$$

We put

$$g(t) = t^3 - a_1t^2 + a_2t - a_3\ . \tag{3.1}$$

Then $F(t) = g(t)\overline{g(t)}$ and

$$\#J_C(\mathbb{F}_q) = g(1)\overline{g(1)}\ . \tag{3.2}$$

We derive the following corollary.

Corollary 3.1. *Let C be a Picard curve over $\mathbb{F}_q$. Then $\#J_C(\mathbb{F}_q)$ is the norm of an element in $\mathbb{Z}[\zeta_3]$. In particular, if $\ell \equiv 2 \mod 3$ is prime and $\ell \mid \#J_C(\mathbb{F}_q)$, we already have $\ell^2 \mid \#J_C(\mathbb{F}_q)$.*

We can reorder the elements w_i such that $g(t)$ is the minimal polynomial of $w_i, i = 1, 2, 3$ over $\mathbb{Q}(\zeta_3)$, i.e.,

$$a_1 = w_1 + w_2 + w_3, \quad a_2 = w_1w_2 + w_1w_3 + w_2w_3 \quad \text{and} \quad a_3 = w_1w_2w_3.$$

Lemma 3.1. *Let C be a Picard curve whose Frobenius has the characteristic polynomial $F(t) = g(t)\overline{g(t)}$. Then either $g(1)$ or $\overline{g(1)}$ annihilates the elements in $J_C(\mathbb{F}_q)$.*

Proof. Let ω be the element in $\mathcal{O}_K$ that represents the Frobenius endomorphism as described above. It is clear that $w \cdot D = D$ for all $D \in J_C(\mathbb{F}_q)$ is equivalent to $J_C(\mathbb{F}_q) \subset \ker(w - 1)$. This implies $(w - 1)$ divides either $g(1)$ or $\overline{g(1)}$. □

We now consider the polynomial $g(t)$ in more detail. Using that $q = w_i\overline{w_i}$ for $i = 1, 2, 3$, we have

$$a_1\overline{a_3} = q\overline{a_2}\ . \tag{3.3}$$

By the triangle inequality we find

$$|a_1| = |w_1 + w_2 + w_3| \le |w_1| + |w_2| + |w_3| \le 3\sqrt{q}\ . \tag{3.4}$$

Consequently,

$$\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \le 9q\ . \tag{3.5}$$

Let $q = p^n$ with $q \equiv 1 \pmod 3$. If $p \equiv 2 \pmod 3$, then n is even, so $n = 2n_1$ for some $n_1 \in \mathbb{N}$. Then $a_3 = \epsilon p^{3n_1}$ for some $\epsilon \in \mathbb{Z}[\zeta_3]$, $\epsilon^6 = 1$, and thus, with (3.3), $a_2 = \overline{a_1}\epsilon p^{n_1}$. On the other hand, if $p \equiv 1 \pmod 3$ we have $p = \pi\overline{\pi}$ for some $\pi \in \mathbb{Z}[\zeta_3]$. We can write $q = p^n = \pi^n\overline{\pi}^n$. Then

$$a_3 = \epsilon\pi^{3n-2k}p^k\ , \quad \text{where } k \in \{0, 1, \dots, \lfloor 3n/2 \rfloor\} \text{ and } \epsilon^6 = 1. \tag{3.6}$$

By (3.3) we have $a_1\overline{\epsilon\pi^{3n-2k}}p^k = p^n\overline{a_2}$, and therefore

$$a_2 = \overline{a_1}\epsilon\pi^{3n-2k}p^{k-n}.$$

We distinguish three cases:

(1) $k < n$: Here a_1 is divisible by π^{n-k}. Let a_1' be an integer in $\mathbb{Z}[\zeta_3]$ such that $a_1 = a_1'\pi^{n-k}$. We find that

$$g(t) = t^3 - a_1'\pi^{n-k}t^2 + \overline{a_1'}\epsilon\pi^{2n-k}t - \epsilon\pi^{3n-2k}p^k.$$

Using (3.5), we see that

$$\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \le \frac{9p^n}{p^{(n-k)}} = 9p^k.$$

(2) $k > n$: Here $\overline{a_2}$ is divisible by p. It follows that the coefficient of t^3 in $F(t)$ is divisible by p. We find that

$$g(t) = t^3 - a_1 t^2 + \overline{a_1}\epsilon\pi^{3n-2k}p^{k-n}t - \epsilon\pi^{3n-2k}p^k.$$

(3) $k = n$: Then

$$g(t) = t^3 - a_1 t^2 + \overline{a_1}\epsilon\pi^n t - \epsilon\pi^n q. \tag{3.7}$$

In the first and second case the Picard curve is not ordinary since its p-rank is smaller than 3 (for the relationship between the L-polynomial and the p-rank see, e.g., [Bou00]). These cases are extremely rare, so that from now on we restrict ourselves to the third case.

The main point of our paper is to determine a_1 given a_3, for which we discuss an algorithm in Section 4. Note that we can bound the number of possible a_3 as follows. There are $\lfloor 3n/2 \rfloor + 1$ possibilites for k, and for each k we have 12 possibilities to choose a_3 since there are exactly 12 solutions π to the equation $\pi\overline{\pi} = p$. If π is one solution, then the other 11 solutions are given by $\zeta_3\pi, \zeta_3^2\pi, -\pi, -\zeta_3\pi, -\zeta_3^2\pi, \overline{\pi}, \zeta_3\overline{\pi}, \zeta_3^2, -\overline{\pi}, -\zeta_3\overline{\pi}, -\zeta_3^2\overline{\pi}$.

Remark 3.1. For the sake of completeness, let us consider the case that $F(t)$ splits over $\mathbb{Q}$. (The numbering below corresponds to the cases on page 1989.)

(2) $F(t)$ splits into two factors $f_i(t), i = 1, 2$ of degree $2i$: There are six possibilities for $f_1(t)$. Over $\mathbb{Q}(\zeta_3)$, $f_2(t)$ decomposes as $(t^2 - a_1 t + a_2)(t^2 - \overline{a_1}t + \overline{a_2})$, where $a_2\overline{a_2} = q^2$ and $a_1\overline{a_2} = q\overline{a_1}$. Thus, there is only a constant number of possibilities for the group order of $J_C(\mathbb{F}_q)$.

(3) $F(t)$ splits into three factors $f_i(t)$ of degree 2: Here for each $f_i(t)$ we have precisely six possibilities. Hence, there are at most $\frac{6^3}{3!}$ possibilities for $F(t)$.

We now give a refined version of Corollary 3.1.

Theorem 3.1. *Let C be a Picard curve over $\mathbb{F}_q$ with $q \equiv 1 \pmod 3$ and ℓ a prime such that $\ell \equiv 2 \pmod 3$. Then*

$$J_C(\mathbb{F}_q)[\ell] \cong (\mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z})^i, \quad \textit{with } i \in \{0, 1, 2, 3\}.$$

Proof. Let D be an element in $J_C(\mathbb{F}_q)$ of order ℓ, and let σ be the automorphism of $J_C(\mathbb{F}_q)$ corresponding to ζ_3. Since $\ell \neq 3$, D^σ and D^{σ^2} are distinct elements also of order ℓ. Since $\mathbb{Z}/\ell\mathbb{Z}$ does not have an endomorphism of order 3, we have $D^\sigma \notin \langle D \rangle$ (the subgroup generated by D in the Jacobian). Since $D + D^\sigma + D^{\sigma^2} = 0$, we get $\langle D, D^\sigma \rangle \cong \mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$, and σ induces an endomorphism on this group. Furthermore, if D_1 and D_2 are two distinct elements in $J_C(\mathbb{F}_q)[\ell]$, then the two subgroups generated by D_1, D_2 along with the action of σ must either be identical or their intersection contains only the identity element, since σ is a nontrivial endomorphism of order 3 acting on the intersection. Therefore, the action of σ decomposes $J_C(\mathbb{F}_q)[\ell]$ into a product of disjoint vector spaces of the form $\mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$. Noting that the ℓ-torsion over the algebraic closure of $\mathbb{F}_q$ has rank ≤ 6 yields the desired result. □

4. A baby step–giant step algorithm

We now show how to determine $\#J_C(\mathbb{F}_q) = F(1) = g(1)\overline{g(1)}$ using a time-memory trade-off. To be precise, we determine the coefficients a_1, a_2, a_3 in (3.1) using Lemma 3.1. For simplicity, we describe our algorithm only in the prime field case; it can easily be adapted to the case $q = p^n$ with $n > 1$.

Since $n = k$ in (3.6), we have exactly 12 candidates for a_3, corresponding to the 12 solutions to $p = \pi\overline{\pi}$. By applying our method to all solutions (e.g., on 12 independent machines), we can easily determine the correct choice of a_3. Thus, here we restrict ourselves to the case that $a_3 = \pi p$. Using (3.7) with $n = 1$, we then have $g(1) = 1 - a_1 + \overline{a_1}\pi - \pi p$. Since $a_1 \in \mathbb{Z} + \mathbb{Z}[\zeta_3] \subset \frac{1}{2}(\mathbb{Z} + \mathbb{Z}[\sqrt{-3}])$, we can write $a_1 = b_1 + b_2\sqrt{-3}$, where $b_i = d_i/2$ with $d_1, d_2 \in \mathbb{Z}$. Note that $d_1 \equiv d_2 \pmod 2$. Hence,

$$g(1) = 1 - \pi p - (1 - \pi)b_1 - (1 + \pi)\sqrt{-3}b_2.$$

Now let D be a divisor in $J_C(\mathbb{F}_p)$. By Lemma 3.1, $g(1)D = 0$ or $\overline{g(1)}D = 0$. Without loss of generality assume that $g(1)D = 0$. Then

$$(1 - \pi p)D - (1 + \pi)\sqrt{-3}b_2 D = (1 - \pi)b_1 D,$$

or, using $\sqrt{-3} = 2\zeta_3 + 1$,

$$2(1 - \pi p)D - (1 + \pi)(2\zeta_3 + 1)d_2 D = (1 - \pi)d_1 D. \tag{4.1}$$

Now, $d_1^2 = 4b_1^2 \leq 4|a_1|^2 \leq 36p$, that is, $|d_1| \leq 6\sqrt{p}$, and $d_2^2 = 4b_2^2 \leq 4|a_1|^2/3 \leq 12p$ and therefore $|d_2| \leq 2\sqrt{3p}$. To find d_1 and d_2, we mimick a baby step–giant step approach: For all integer values d_2 with $|d_2| \leq 2\sqrt{3p}$, store the left-hand side of (4.1) in a hash table (the "baby steps"), and then for all $|d_1| \leq 6\sqrt{p}$ check if the right-hand side equals one of the stored elements ("giant steps"). If this is the case for a pair (d_1, d_2), then we put $a_1 = (d_1 + d_2\sqrt{-3})/2$, and we have that $2g(1)\overline{g(1)}$ is a multiple of the order of the divisor D. From this, the exact order can be easily determined, as shown in the pseudo-code below. Note that with the correct solution π, the algorithm is guaranteed to succeed, while it outputs "failure" only if the input value for π was wrong.

Algorithm 4.1. *Algorithm to compute* $\operatorname{ord} D$ *for* $D \in J_C(\mathbb{F}_p)$.
Input: Divisor $D \in J_C(\mathbb{F}_p)$, solution π to $p = \pi\overline{\pi}$
Output: $\operatorname{ord} D$ or "failure".

```
(1) { Precomputation: }
    $A \leftarrow 2(1 - \pi p)D$ and $B \leftarrow (1 + \pi)(2\zeta_3 + 1)D$ and $C \leftarrow (1 - \pi)D$.
(2) $M \leftarrow \lfloor 2\sqrt{3p} \rfloor$.
(3) $r_0 \leftarrow A - MB$ and $\mathcal{R} \leftarrow \{(r_0, M)\}$.
(4) { Baby steps: }
    For $n \leftarrow 1, \dots, 2M$ do the following:
        $r_n \leftarrow r_{n-1} + B$.
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(r_n, M - n)\}$.
(5) $s_0 \leftarrow 0$ and $t_0 \leftarrow s_0$.
    If $(s_0, j) \in \mathcal{R}$ for some $j \in \{-M, \dots, M\}$, then $a_1 \leftarrow j\sqrt{-3}/2$. GOTO Step
    (8).
```

```
(6) { Giant steps: }
    For $n \leftarrow 1, \ldots, \lfloor 6\sqrt{p} \rfloor$ do the following:
    $s_n \leftarrow s_{n-1} + C$.
    If $(s_n, j) \in \mathcal{R}$ for some $j \in \{-M, \ldots, M\}$, then $a_1 \leftarrow (n + j\sqrt{-3})/2$. GOTO Step (8).
    $t_n \leftarrow t_{n-1} - C$.
    If $(t_n, j) \in \mathcal{R}$ for some $j \in \{-M, \ldots, M\}$, then $a_1 \leftarrow (-n + j\sqrt{-3})/2$. GOTO Step (8).
(7) { No match has been found, so $\pi$ was wrong. }
    Return "failure".
(8) $g \leftarrow 1 - a_1 + \overline{a_1}\pi - \pi p$ and $N \leftarrow 2g\overline{g}$. { $N$ is a multiple of $\operatorname{ord} D$. }
(9) { Determine $\operatorname{ord} D$: }
    Factor $N$ into prime factors: $N = \prod_{i=1}^{d} p_i^{e_i}$.
    For $i \leftarrow 1, \ldots, d$ find the smallest number $E_i$ such that $(N/p_i^{E_i})D \neq 0$.
(10) Return $T \leftarrow \prod_{i=1}^{d} p_i^{e_i - E_i + 1}$.
```

It is immediate that for each divisor D, Algorithm 4.1 requires at most $4(3 + \sqrt{3})\sqrt{p} \leq 19\sqrt{p}$ additions in the Jacobian to execute the baby steps and giant steps; the precomputation requires $O(\log p)$ operations. We need to store at most $4\sqrt{3p} + 1$ pairs $(D, j) \in J_C(\mathbb{F}_p) \times \{-M, \ldots, M\}$.

Next we discuss a variant of 4.1 that is designed to address some of the issues that arise in implementing the arithmetic in the Jacobians of Picard curves. In particular, given two divisors D_1 and D_2, the standard addition and reduction formulas compute $D_3 \sim -D_1 - D_2$. Let this addition and reduction be denoted by $\oplus$. To compute $D_1 + D_2$, an additional inversion step is required. Correspondingly, let the standard symbol $+$ represent addition followed by the reduction and inversion. Now, unlike the arithmetic for hyperelliptic curves, inversion is not free. We can avoid the extra inversion step by working in the inverse class half of the time, with respect to a standard double and add algorithm. To be more precise, using $\oplus$, after an odd number of operations, we will be in the inverse, and hence "wrong", class. This merely requires us to maintain a flag that keeps track of which of the two possible classes our current element resides. Moreover, we can exploit that $d_1 \equiv d_2 \pmod 2$ to break the search space up into two pieces that can be processed in parallel. We give the pseudo-code of the new algorithm first, with the explanation to follow. It might be helpful to note that the superscripts "+" and "-" as well as the variable "toggle" are due to the inversion-freeness, while the subscripts "even" and "odd" relate to breaking up the search space.

Algorithm 4.2. *Algorithm to compute* $\operatorname{ord} D$*, avoiding inversions.*
INPUT: Divisor $D \in J_C(\mathbb{F}_p)$, solution π to $p = \pi\overline{\pi}$.
OUTPUT: $\operatorname{ord} D$, or "failure".

```
(1) { Precomputation: }
    $A_{\text{even}} \leftarrow 2(1 - \pi p)D$ and $A_{\text{odd}} \leftarrow 2(1 - \pi p)D - (1 + \pi)(2\zeta_3 + 1)D - (1 - \pi)D$,
    $B^+ \leftarrow -2(1 + \pi)(2\zeta_3 + 1)D$ and $B^- \leftarrow 2(1 + \pi)(2\zeta_3 + 1)D$,
    $C^+ \leftarrow 2(1 - \pi)D$ and $C^- \leftarrow -2(1 - \pi)D$.
(2) $M \leftarrow \lfloor \sqrt{3p} \rfloor + 1$ and *toggle* $\leftarrow 1$.
(3) $r^+_{\text{even},0} \leftarrow A_{\text{even}}$ and $r^-_{\text{even},0} \leftarrow A_{\text{even}}$, and $\mathcal{R}_{\text{even}} \leftarrow \{(r^+_{\text{even},0}, 0)\}$.
    $r^+_{\text{odd},0} \leftarrow A_{\text{odd}}$ and $r^-_{\text{odd},0} \leftarrow A_{\text{odd}}$, and $\mathcal{R}_{\text{odd}} \leftarrow \{(r^+_{\text{odd},0}, 0)\}$.
```

```
 (4) { Baby steps: }
     For $n \leftarrow 1, \dots, M$ do the following:
         If ($toggle = 1$)
             $r^+_{\mathrm{even},n} \leftarrow r^+_{\mathrm{even},n-1} \oplus B^+$,
             $r^-_{\mathrm{even},n} \leftarrow r^-_{\mathrm{even},n-1} \oplus B^-$,
             $r^+_{\mathrm{odd},n} \leftarrow r^+_{\mathrm{odd},n-1} \oplus B^+$,
             $r^-_{\mathrm{odd},n} \leftarrow r^-_{\mathrm{odd},n-1} \oplus B^-$.
         Else
             $r^+_{\mathrm{even},n} \leftarrow r^+_{\mathrm{even},n-1} \oplus B^-$,
             $r^-_{\mathrm{even},n} \leftarrow r^-_{\mathrm{even},n-1} \oplus B^+$,
             $r^+_{\mathrm{odd},n} \leftarrow r^+_{\mathrm{odd},n-1} \oplus B^-$,
             $r^-_{\mathrm{odd},n} \leftarrow r^-_{\mathrm{odd},n-1} \oplus B^+$.
         $\mathcal{R}_{\mathrm{even}} \leftarrow \mathcal{R}_{\mathrm{even}} \cup \{(r^+_{\mathrm{even},n}, n), (r^-_{\mathrm{even},n}, -n)\}$.
         $\mathcal{R}_{\mathrm{odd}} \leftarrow \mathcal{R}_{\mathrm{odd}} \cup \{(r^+_{\mathrm{odd},n}, n), (r^-_{\mathrm{odd},n}, -n)\}$.
         $toggle \leftarrow -toggle$.
 (5) $s_0 \leftarrow 0$.
     If $(s_0, j) \in \mathcal{R}_{\mathrm{even}}$ for some $j \in \{-M, \dots, M\}$, then $a_1 \leftarrow j\sqrt{-3}$. GOTO
     Step (8).
     If $(s_0, j) \in \mathcal{R}_{\mathrm{odd}}$ for some $j \in \{-M, \dots, M\}$, then $a_1 \leftarrow (1+(2j+1)\sqrt{-3})/2$.
     GOTO Step (8).
 (6) { Giant steps: }
     $toggle \leftarrow 1$.
     For $n \leftarrow 1, \dots, \lfloor 3\sqrt{p} \rfloor + 1$ do the following:
         If ($toggle = 1$) then $s_n \leftarrow s_{n-1} \oplus C^+$.
         Else $s_n \leftarrow s_{n-1} \oplus C^-$.
         If $(s_n, j) \in \mathcal{R}_{\mathrm{even}}$ for some $j \in \{-M, \dots, M\}$,
             then $a_1 \leftarrow (-1)^{n+j} n + j\sqrt{-3}$. GOTO Step (8).
         If $(-s_n, j) \in \mathcal{R}_{\mathrm{even}}$ for some $j \in \{-M, \dots, M\}$,
             then $a_1 \leftarrow (-1)^{n+1+j} n + j\sqrt{-3}$. GOTO Step (8).
         If $(s_n, j) \in \mathcal{R}_{\mathrm{odd}}$ for some $j \in \{-M, \dots, M\}$,
             then $a_1 \leftarrow (((-1)^{n+j} 2n + 1) + (2j + 1)\sqrt{-3})/2$. GOTO
     Step (8).
         If $(-s_n, j) \in \mathcal{R}_{\mathrm{odd}}$ for some $j \in \{-M, \dots, M\}$,
             then $a_1 \leftarrow (((-1)^{n+1+j} 2n + 1) + (2j + 1)\sqrt{-3})/2$. GOTO
     Step (8).
         $toggle \leftarrow -toggle$.
 (7) { No match has been found, so $\pi$ was wrong. }
     Return "failure".
 (8) $g \leftarrow 1 - a_1 + \overline{a_1}\pi - \pi p$ and $N \leftarrow 2g\overline{g}$. { $N$ is a multiple of $\operatorname{ord} D$. }
 (9) { Determine $\operatorname{ord} D$: }
     Factor $N$ into prime factors: $N = \prod_{i=1}^{d} p_i^{e_i}$.
     For $i \leftarrow 1, \dots, d$ find the smallest number $E_i$ such that $N/p_i^{E_i} D \neq 0$.
(10) Return $T \leftarrow \prod_{i=1}^{d} p_i^{e_i - E_i + 1}$.
```

We now explain Algorithm 4.2. Let $D \in J_C(\mathbb{F}_p)$. In (4.1), if d_1, d_2 are even, there exist integers k_1, k_2 with $|k_1| \leq 3\sqrt{p}$ and $|k_2| \leq \sqrt{3p}$ and such that $d_i = 2k_i$

$(i = 1, 2)$. Substituting into (4.1) gives

$$2(1-\pi p)D - 2(1+\pi)(2\zeta_3+1)k_2D = 2(1-\pi)k_1D.$$

In terms of A_{even}, B^+ and C^+ (Step (2) of Algorithm 4.2), this means

$$A_{\mathrm{even}} + k_2B^+ = k_1C^+. \tag{4.2}$$

If d_1, d_2 are odd, we instead write $d_i = 2k_i + 1$ $(i = 1, 2)$. Then

$$\begin{aligned}&[2(1-\pi p)D - 2(1+\pi)(2\zeta_3+1)D - 2(1-\pi)D] - (1+\pi)(2\zeta_3+1)\,2k_2\,D\\ &\quad = (1-\pi)\,2k_1\,D,\end{aligned}$$

which in terms of A_{odd}, B^+ and C^+ reads as

$$A_{\mathrm{odd}} + k_2B^+ = k_1C^+.$$

Now assume for some n and j we have $(s_n, j) \in \mathcal{R}_{\mathrm{even}}$ (which happens if d_1, d_2 even). Then

$$(-1)^j(A_{\mathrm{even}} + jB^+) = (-1)^n nC^+.$$

Comparing with (4.2), we see that $k_1 = (-1)^{j+n}n$ and $k_2 = j$. Substituting back in for a_1 yields $a_1 = (-1)^{j+n}n + j\sqrt{-3}$. The additional check if $(-s_n, j) \in \mathcal{R}_{\mathrm{even}}$ is necessary to compensate for the lack of inversions when adding divisors. In fact, if we instead found $(-s_n, j) \in \mathcal{R}_{\mathrm{even}}$, then we have an extra inversion to compensate for. We obtain

$$(-1)^j(A_{\mathrm{even}} + jB^+) = (-1)^{n+1}nC^+,$$

which implies $a_1 = (-1)^{j+n+1}n + j\sqrt{-3}$. A similar argument holds for searching for k_1, k_2 in case the d_i are odd, in which case the match is found within $\mathcal{R}_{\mathrm{odd}}$. In any case, by construction of a_1 and with $g = 1 - a_1 + \overline{a_1}\pi - \pi p$, $2g\overline{g}$ is a multiple of $\operatorname{ord} D$.

Note that Algorithm 4.2 requires essentially the same number of operations in the Jacobian as Algorithm 4.1, but each operation is cheaper since we use "$\oplus$" instead of "$+$".

Remark 4.1.

1. Note that in most cases, Algorithm 4.1 (resp. Algorithm 4.2) recovers not only $g(1)$ but also the coefficients of the polynomial $g(t)$. This gives us the whole L-polynomial of the curve C. The L-polynomial contains more information than the group order of the Jacobian alone. In particular, we find $\#C(\mathbb{F}_{q^k})$ for all $k \in \mathbb{N}$.
2. In Section 8.2 we discuss how to further reduce the running time and space requirements by working with smaller bounds on the norm of a_1.

4.1. **Computing $\#J_C(\mathbb{F}_p)$.**

If

$$\operatorname{ord} D > 12p^{5/2} + 40p^{3/2} + 12\sqrt{p},$$

then $\operatorname{ord} D$ has a unique multiple in the Hasse-Weil interval $[(\sqrt{p}-1)^6, (\sqrt{p}+1)^6]$, which equals $\#J_C(\mathbb{F}_p)$. For cryptographically interesting curves whose Jacobian has a group order almost a prime, this is the most likely case.

If, on the other hand, more than one multiple of $\operatorname{ord} D$ lies in the Hasse-Weil interval, instead of running the same algorithm with another divisor (which is not only expensive but also might not yield the desired result either), there are two ways to proceed.

Let $N = \#J_C(\mathbb{F}_p)$. By Corollary 3.1 if a prime ℓ with $\ell \equiv 2 \pmod 3$ divides N, then also $\ell^2 \mid N$. Hence,

$$\prod_{\ell \mid (\operatorname{ord} D),\ \ell \equiv 2 \pmod 3} \ell^2 \mid N.$$

Now assume ℓ is a prime such that $\ell \mid\mid \operatorname{ord} D$ and $\ell \equiv 1 \pmod 3$. Let $D_1 = (\operatorname{ord} D/\ell)D$. If $\ell \mid\mid N$, then ζ_3 must permute the ℓ-torsion points generated by D_1 among themselves. Now let $\eta \in \mathbb{Z}/l\mathbb{Z}$ be a cube root of unity. If ζ_3 acts cyclicly on D_1, then $\eta D_1 = D_1^{\zeta_3}$ or $\eta D_1 = D_1^{\zeta_3^2}$. If neither holds, we can conclude that $\ell^2 \mid N$.

Thus, by checking the factors of $\operatorname{ord} D$, we may be able to determine a larger divisor of N than $\operatorname{ord} D$ itself, which may have a unique multiple in the Hasse-Weil interval and we are done.

Otherwise, note that Algorithm 4.2 (and 4.1) is designed to find *candidates* for the coefficients of $g(t)$. In particular, if $\operatorname{ord} D$ is very small, there might be different candidates for $g(t)$. Hence, if the output $\operatorname{ord} D$ does not uniquely determine N, instead of terminating we continue the computation from where the algorithm's GOTO command was executed. This produces further successful table lookups in $\mathcal{R}$ that yield further candidates for $g(t)$. On completion of all giant steps, the set of candidates for $g(t)$ contains the correct polynomial. Only the proper value for $g(1)$ annihilates all divisors in the Jacobian. By testing the candidates with randomly chosen divisors, wrong candidates can be eliminated and eventually $\#J_C(\mathbb{F}_p)$ can be determined from the surviving candidate.

5. The Hasse-Witt matrix

Using the Cartier-Manin operator or, more precisely, its concrete realization as the Hasse-Witt matrix, we can deduce information on the L-polynomial modulo p of a curve defined over a field of characteristic p provided that p is not too large.

5.1. The definition and consequences.

If C is a Picard curve, the form of the Hasse-Witt matrix is known explicitly.

Theorem 5.1. [Sar91] *Let $C : y^3 = f(x)$ be a Picard curve defined over the finite field $\mathbb{F}_q$ of characteristic p. Set $F(x,y) = y^3 - f(x)$ and let*

$$F(x,y)^{p-1} = \sum_{i,j} c_{i,j} x^i y^j. \tag{5.1}$$

Then the Hasse-Witt matrix of C is equal to

$$\begin{pmatrix} c_{p-1,p-1} & c_{2p-1,p-1} & c_{p-1,2p-1} \\ c_{p-2,p-1} & c_{2p-2,p-1} & c_{p-2,2p-1} \\ c_{p-1,p-2} & c_{2p-1,p-2} & c_{p-1,2p-2} \end{pmatrix}.$$

Given a matrix $A = (a_{ij})$ with entries in $\mathbb{F}_q$, let $A^{(p)}$ denote the matrix (a_{ij}^p). The next theorem gives a means to compute the L-polynomial modulo p.

Theorem 5.2. [Man65] *Let C be a curve of genus g defined over a finite field $\mathbb{F}_q$ of characteristic p and let A be the Hasse-Witt matrix of C. Set $A_\Phi = AA^{(p)} \cdots A^{(p^n-1)}$. Suppose $F(t)$ is the characteristic polynomial of the Frobenius endomorphism and $\chi_\Phi(t)$ is the characteristic polynomial of A_Φ. Then*

$$F(t) \equiv (-1)^g t^g \chi_\Phi(t) \pmod p.$$

In Section 5.2 we show how to efficiently compute the Hasse-Witt matrix and hence $F(t)$ modulo p in the Picard case. Assume now that we know $F(t)$ modulo p. This information can be used to speed up the algorithms from Section 4 as follows. Recall case (3) on page 1990 where $F(t)$ splits into $g(t)\overline{g}(t)$ over $\mathbb{Z}[\zeta_3]$ with $g(t) = t^3 - a_1t^2 + \overline{a_1}\pi^n t - \pi^n q$ for some $\pi \in \mathbb{Z}[\zeta_3]$, $\pi\overline{\pi} = p$ and $a_1 \in \mathbb{Z}[\zeta_3]$ not divisible by p. We can write

$$F(t) = t^6 - c_1t^5 + c_2t^4 - c_3t^3 + c_2qt^2 - c_1q^2t + q^3 \in \mathbb{Z}[t],$$

where

$$\begin{aligned} c_1 &= \mathrm{Tr}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1), \\ c_2 &= \mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) + (\overline{a_1}\pi^n + a_1\overline{\pi^n}), \\ c_3 &= \overline{a_1}^2\pi^n + a_1^2\overline{\pi^n} \\ &= \mathrm{Tr}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1)(\overline{a_1}\pi^n + a_1\overline{\pi}^n) - \mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1)(\pi^n + \overline{\pi}^n). \end{aligned} \tag{5.2}$$

Using Theorem 5.2, we find $c_i \bmod p$. Now we have 12 possibilities for π^n and six possibilities for $\pi^n + \overline{\pi^n}$. Since we can run through all such possible solutions, we can assume that we know π^n. Then we can solve the system (5.2) for $(\overline{a_1}\pi^n + a_1\overline{\pi}^n) \bmod p$ and $\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \mod p$. This yields d_1, $d_2 \bmod p$ in the representation $a_1 = (d_1 + d_2\sqrt{-3})/2$.

Example. We illustrate the above by an example over the relatively small field $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$, where $p = 1249$ and $\alpha^2 + 28\alpha + 328 = 0$. Consider the curve $C : y^3 = x^4 + f_2x^2 + f_1x + f_0$ with

$$f_2 = 741\alpha + 329, \quad f_1 = 463\alpha + 807 \quad \text{and} \quad f_0 = 381\alpha + 84.$$

Then, modulo p, $F(t) = t^6 + 16t^5 + 60t^4 + 47t^3$. Now choose $\pi^2 \in \mathbb{Z}[\zeta_3] \setminus \mathbb{Z}$ such that $\pi^2\overline{\pi}^2 = p^2$, for example, $\pi^2 = 871 - 560\zeta_3$. From the L-polynomial we can compute d_1 and d_2 modulo p where $a_1 = (d_1 + d_2\sqrt{-3})/2$. Using that $|a_1| \leq 3p$ (see (3.4)), we find a list of candidates for $[a_{11}, a_{12}]$ in the representation $a_1 = a_{11} + a_{12}\zeta_3$. By choosing a random divisor D in the divisor class group of degree zero and trying all possible candidates $[d_1, d_2]$, we find that $a_1 = -548 + 169\zeta_3$. Then $\#J_C(\mathbb{F}_{p^2}) = 3799499968769928913$.

5.2. **Computation.** Now we describe how to efficiently compute the Hasse-Witt matrix of a Picard curve C. If $p \equiv 1 \pmod 3$, the Hasse-Witt matrix has the form

$$\begin{pmatrix} * & * & \\ * & * & \\ & & * \end{pmatrix}.$$

If $p \equiv 2 \pmod 3$, it looks like

$$\begin{pmatrix} & & * \\ & & * \\ * & * & \end{pmatrix};$$

in particular, C does not have maximal p-rank.

For simplicity, let us concentrate on the case $p \equiv 1 \pmod 3$; the case $p \equiv 2 \pmod 3$ works analogously.

The matrix entry $c_{p-1,p-1}$ is the coefficient of $x^{p-1}y^{p-1}$ in (5.1), which is equal to

$$\binom{p-1}{\frac{p-1}{3}} f_{p-1}^{2p-2},$$

where f_{p-1}^{2p-2} denotes the coefficient of x^{p-1} in $f^{2p-2/3}$. (In general, let f_j^i denote the j-th coefficient of $f^{i/3}$.) Furthermore,

$$\begin{aligned} c_{2p-1,p-1} &= \binom{p-1}{\frac{p-1}{3}} f_{2p-1}^{2p-2}, & c_{p-2,p-1} &= \binom{p-1}{\frac{p-1}{3}} f_{p-2}^{2p-2}, \\ c_{2p-2,p-1} &= \binom{p-1}{\frac{p-1}{3}} f_{2p-2}^{2p-2}, & c_{p-1,2p-2} &= \binom{p-1}{\frac{p-1}{3}} f_{p-1}^{p-1}. \end{aligned}$$

Lemma 5.1. *Let p be a prime, $p \equiv 1 \pmod 3$. Then $\binom{p-1}{k} \equiv (-1)^k \pmod p$. In particular, $\binom{p-1}{\frac{p-1}{3}} \equiv 1 \pmod p$.*

Proof. We have

$$\binom{p-1}{k} = \frac{(p-1)(p-2)\cdots(p-k)}{1\cdot 2\cdots k} \equiv \frac{(-1)\cdots(-k)}{1\cdots k} = (-1)^k \pmod p.$$

For the second statement, it suffices to note that since p is odd, $p \equiv 1 \pmod 6$ and thus $(p-1)/3 \equiv 0 \pmod 2$. $\square$

It remains to compute the coefficients of $x^{p-2}, x^{p-1}, x^{2p-2}$, and x^{2p-1} in $h_1(x) = f^{(2p-2)/3}$ and of x^{p-1} in $h_2(x) = f(x)^{(p-1)/3}$. A naive implementation of this computation would take $O(p)$ operations in $\mathbb{F}_q$. Bostan, Gaudry and Schost [BGS04] give a much faster algorithm for computing a fixed coefficient of a polynomial of the form f^h for some $h \in \mathbb{N}$. There, it is used to determine the Hasse-Witt matrix of a hyperelliptic curve, but it can easily be adapted to Picard curves. We briefly summarize the main idea and describe the differences in the Picard curve case.

Let us first consider h_1 (the argument for h_2 is analogous). For simplicity, we write h instead of h_1. The function h satisfies the differential equation

$$fh' - \frac{2(p-1)}{3} f'h = 0.$$

For the k-th coefficient of this identity we obtain

(5.3)
$$\begin{aligned}(k+1)f_0 h_{k+1} &+ \left(k - \frac{2(p-1)}{3}\right) f_1 h_k \\ &+ \left((k-1) - \frac{4(p-1)}{3}\right) f_2 h_{k-1} + \left((k-3) - \frac{8(p-1)}{3}\right) h_{k-3} = 0,\end{aligned}$$

where $f(x) = x^4 + f_2x^2 + f_1x + f_0$. Now let $U_k = [h_{k-3}, \dots, h_k]^t$, and let $A(k)$ be the 4×4 matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\frac{8(p-1)}{3}-(k-4)}{f_0 k} & 0 & \frac{f_2(\frac{4(p-1)}{3}-(k-2))}{f_0 k} & \frac{f_1(\frac{2(p-1)}{3}-(k-1))}{f_0 k} \end{pmatrix}.$$

The initial vector is

$$U_0 = [0, \dots, 0, f_0^{\frac{2(p-1)}{3}}]^t.$$

By (5.3), $U_{k+1} = A(k+1)U_k$. For the Hasse-Witt matrix we need the vectors U_{p-1} and U_{2p-1}. The algorithm from [BGS04] computes $U_{k+1} = A(k+1)\cdots A(1)U_0$ in time $\widetilde{O}(\sqrt{k})$, where $\widetilde{O}(m)$ is the Soft-Oh notation that does not take into account any contributions in terms of $\log m$. Since that algorithm only works for matrices whose entries are polynomials in k (and not rational functions), we have to slightly modify the sequence U_k. So define $V_k = f_0^k k! U_k$. Then $V_{k+1} = B(k+1)V_k$ where $B(k) = f_0 k A(k)$ and V_k is a sequence of matrices whose entries are polynomials in k. Note that

$$V_{p-1} = f_0^{p-1}(p-1)!U_{p-1} = -f_0^{p-1}U_{p-1},$$

by Wilson's lemma. The power f_0^{p-1} can easily be determined using a square-and-multiply technique. Now, since $k! \equiv 0 \pmod p$ for $k \geq p$, the sequence V_k does not make sense in $\mathbb{F}_q = \mathbb{F}_{p^n}$ for $k \geq p$. For computing the $(2p-2)$-th and $(2p-1)$-st coefficients of $f^{(2p-2)/3}$, we lift the problem to characteristic zero, more precisely, to the unramified extension W of $\mathbb{Q}_p$ with residue degree q. Since the p-valuation of $(2p-1)!$ is equal to 1, it is enough to do all computation with precision 2. Since we easily check that $(2p-1)! \equiv p \pmod{p^2}$, we have

$$V_{2p-1} \equiv p f_0^{(2p-1)} U_{p-1} \pmod{p^2}.$$

So we are left with the efficient computation of V_{p-1} and V_{2p-1}, which works just as described in Sections 2 and 3 of [BGS04]. The running time of the algorithm is

$$O(\sqrt{p}\log^3 p \log\log^2 p \log\log\log p)$$

operations in $\mathbb{F}_q$, where it is assumed that the multiplication of two polynomials of degree n takes $O(n\log(n)\log\log(n))$ operations.

6. Running times

We now discuss the running time complexity of our new method. As before, let $q = p^n$. Let $N = \#J_C(\mathbb{F}_q)$. As seen in the previous section, the Hasse-Witt matrix can be computed in time $\widetilde{O}(\sqrt{p})$.

In the prime field case, Algorithm 4.2 requires $O(\sqrt{p}) = O(N^{1/6})$ operations in the Jacobian. No gain is achieved by combining it with the computation of the Hasse-Witt matrix.

If $n = 2$, the most efficient approach is to first compute the Hasse-Witt matrix and determine d_1, d_2 modulo p (where $a_1 = (d_1 + d_2\sqrt{-3})/2$). Then a slight modification of Algorithm 4.2 computes $\operatorname{ord} D$ for any $D \in J_C(\mathbb{F}_q)$ in a constant number of steps. The overall running time is $\widetilde{O}(\sqrt{p})$.

Table 1. Running time complexities for computing $N = \#J_C(\mathbb{F}_q)$. (For $n = 1$ or $n = 4$: number of operations in $\mathbb{F}_q$.)

n	Running time in terms of		
$(q = p^n)$	p	$q = p^n$	$N = \#J_C(\mathbb{F}_q)$
1	$O(\sqrt{p})$	$O(\sqrt{q})$	$O(N^{1/6})$
2	$\widetilde{O}(\sqrt{p})$	$\widetilde{O}(q^{1/4})$	$\widetilde{O}(N^{1/12})$
3	$\widetilde{O}(\sqrt{p})$	$\widetilde{O}(q^{1/6})$	$\widetilde{O}(N^{1/18})$
4	$O(p)$	$O(q^{1/4})$	$O(N^{1/12})$
> 4 ([GG03])	$\widetilde{O}(pn^3)$	$\widetilde{O}(q^{1/n}n^3)$	$\widetilde{O}(N^{1/3n}n^3)$

If $n = 3$, then d_1 and d_2 are bounded by $O(p^{3/2})$. Upon computing the Hasse-Witt matrix and determining d_i modulo p, Algorithm 4.2 takes $O(p^{1/4})$ operations in the Jacobian. Still, the total running time is $\widetilde{O}(\sqrt{p})$.

For $n = 4$, the d_i are of size $O(p^2)$. Then Algorithm 4.2 requires $O(p)$ operations in the Jacobian, which results in an overall running time $\widetilde{O}(p)$.

For $n > 4$, the Gaudry-Gurel extension of Kedlaya's algorithm [GG03] for computing the cardinality of the Jacobian of a curve is definitely faster than our approach. It runs in time $\widetilde{O}(pn^3)$.

We summarize our discussion in Table 1. If $n = 1$ or $n = 4$, the most time-consuming step is the baby step–giant step algorithm. In this case, we give the complexity in terms of operations in $\mathbb{F}_q$ rather than in Soft-Oh notation.

7. Computational results

We implemented Algorithm 4.2 in C++ using NTL. Here we give four examples, three of which are over a finite prime field $\mathbb{F}_p$, and one over a field $\mathbb{F}_{p^3}$ where the algorithm is most efficient (cf. Table 1). In all examples, the Picard curve C is given by the equation $y^3 = x^4 + f_2x^2 + f_1x + f_0$.

(1) We take $p = 123456799903$. Then $p \equiv 1 \pmod 3$, and one solution of $\pi\overline{\pi} = p$ is given by $\pi = -396954 - 271123\zeta_3$ where $\zeta_3 = (-1 + \sqrt{-3})/2$. Since in cryptographic applications we aim for Jacobians with prime or almost-prime group order, we first determine parameters f_2, f_1 and f_0 for C such that $J_C(\mathbb{F}_p)$ has no 2- and 3-torsion points. We use

$$f_2 = 17832302073, \quad f_1 = 31508807039, \quad \text{and} \quad f_0 = 100619648414.$$

We then apply Algorithm 4.2, which finds $a_1 = 186266 - 281004\zeta_3$. We compute

$$\#J_C(\mathbb{F}_p) = 1881666909417341162265915946849651,$$

which is a prime with 34 decimal digits. Once the correct element π was found, this computation took 575.75 seconds on a 2000 Mhz laptop with 512 MB RAM. The total running time is about 12 times as high, given that the algorithm has to be run for each of the 12 possible choices for π. Recall that the latter can be fully parallelized.

(2) Let $p = 251123481769 \equiv 1 \pmod 3$ and C be the Picard curve

$$y^3 = x^4 + 91045798812\,x^2 + 220534121135\,x + 188632117268$$

over $\mathbb{F}_p$. We find $\pi = -356107 + 216933\zeta_3$ and $a_1 = 430420 + 480361\zeta_3$. Then $\#J_C(\mathbb{F}_p) = 15836576914121881401954674414636461$, which has a 32-digit prime factor.

(3) Let $p = 5467598293543$. Given our particular implementation, this is roughly the largest prime for which we can run Algorithm 4.2 entirely in RAM memory. We apply our algorithm to 50 random Picard curves over $\mathbb{F}_p$ with no 2- or 3-torsion. During the first few such computations, we observe that only three different values of π ever occur as correct choices, namely $2494359 + 2142293\zeta_3$, $-2142293 + 352066\zeta_3$, and $-352066 - 2494359\zeta_3$, which differ by primitive cube roots of unity. For each curve, we thus use Algorithm 4.2 with these values of π first (and we are always successful in

finding a_1 that way). For C with parameters

$$f_2 = 2089118456211, \quad f_1 = 5172274005281, \quad \text{and } f_0 = 2251178161215,$$

and with $\pi = 2494359 + 2142293\zeta_3$, we find that $a_1 = 236476 + 1436356\zeta_3$. This yields

$$\#J_C(\mathbb{F}_p) = 163451862786655376815035856445333176273,$$

a 39-digit (127-bit) prime. The associated L-polynomial is

$$\begin{aligned} L(t) =& 1 + 963404\, t + 5023890226968\, t^2 - 7372470654968953727\, t^3 \\ &+ 27468613631917591758867624\, t^4 + 2880060717983503549600019269799 6\, t^5 \\ &+ 163451833986020728373740908484459374007\, t^6. \end{aligned}$$

For each given π, Algorithm 4.2 took just over two hours with a dual 2800 Mhz processor with 1.5 GB of RAM, with only one processor being used for the computation. Because of memory constraints, the odd and even cases were run in serial rather than in parallel.

(4) Now let $p = 1000033 \equiv 1 \pmod 3$ and let $\mathbb{F}_q = \mathbb{F}_{p^3} = \mathbb{F}_p(\alpha)$ where α is a root of the polynomial $t^3 + 594964t^2 + 635999t + 958324$. We choose

$$\begin{aligned} f_2 &= 323409\alpha^2 + 882832\alpha + 836130, \\ f_1 &= 282274\alpha^2 + 299397\alpha + 199524, \\ f_0 &= 88913\alpha^2 + 372958\alpha + 979902, \end{aligned}$$

to achieve that $C : y^3 = x^4 + f_2x^2 + f_1x + f_0$ has a Jacobian without 2- and 3-torsion points. We compute the Hasse-Witt matrix and determine the L-polynomial of C modulo p as

$$L(t) = t^3 + 948468\, t^2 + 893878\, t + 810643.$$

For each choice π with $\pi\overline{\pi} = p$ we then compute d_1 and d_2 modulo p. For $\pi = 1154468017 + 554835313\zeta_3$, we find $d_1 \bmod p = 51565$ and $d_2 \bmod p \in \{23327, 976706\}$. Then a variant of Algorithm 4.2 is run, which considers only these congruence classes of $d_i \bmod p$. We find $a_1 = 985046624 - 1023360\zeta_3$, and determine

$$\#J_C(\mathbb{F}_{p^3}) = 1000297037235511938842496644574071818992774646718689681 = 11^2\ell,$$

where ℓ is a 173-bit prime. Applying our baby step–giant step algorithm took 570.32 seconds on the same machine as for (1), this time using an implementation in Magma [Mag]. Again, the entire computation involves 12 such applications.

8. Miscellaneous

8.1. **Parallelization.** It is possible to parallellize the algorithms of Section 4 by employing a divide-and-conquer approach based on congruences.

Let $\alpha \in \mathbb{Z}[\zeta_3]$ be fixed. We can write $a_1 \equiv \beta \pmod \alpha$ for some residue β modulo α. Then there exists $\gamma \in \mathbb{Z}[\zeta_3]$ such that $a_1 = \beta + \alpha\gamma$. Substituting this into (3.7) we get

$$g(1) = 1 - \beta + \alpha\gamma + \overline{\beta + \alpha\gamma}\pi - \pi p = 1 - \beta + \overline{\beta}\pi - \pi p - \gamma\alpha + \overline{\gamma\alpha}\pi.$$

Assuming we know β, we then have to solve for γ, which is similar to the situation in Section 4 where we solved for a_1. The advantage now is that the search for γ occurs in the range

$$|\gamma| \leq \frac{3\sqrt{p} + |\beta|}{|\alpha|}.$$

Since we may take $|\beta| < |\alpha|$, this results in a speed-up of a single run of the algorithm by a factor of $|\alpha|$, and a decrease in the memory requirements by the same factor. But since β is unknown, this method has to be applied to all $\mathrm{N}(\alpha)$ different congruence classes modulo α. Thus, the overall running time increases by a factor of $|\alpha|$ so that this parallelization is only interesting if there are $\mathrm{N}(\alpha)$ processors available, or if there is not enough memory available to store the $\sqrt{3p}$ baby steps as required by Algorithm 4.2.

8.2. **Norm of a_1.** Recall that in Section 4 we wrote $a_1 = (d_1 + d_2\sqrt{-3})/2$, and used the bound (3.5) $\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \leq 9q$ to bound $|d_1| \leq 6\sqrt{q}$ and $|d_2| \leq 2\sqrt{3q}$. Here we report on experimental results that suggest that for the majority of curves, the norm of a_1 is much smaller. Thus, Algorithms 4.1 and 4.2 search ranges which are unlikely to contain a_1. This offers room for improvement, as we discuss below.

We worked with 62 prime fields $\mathbb{F}_p$ with $p \equiv 1 \pmod 3$ and p ranging from 397 to 500029. For each such field, for 1000 randomly chosen Picard curves with no $(1-\zeta_3)$-torsion, we calculated the norm of a_1, and since $a_1 \neq 0$, we computed

$$k = \lfloor 9p/\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \rfloor.$$

We give a sample of our results in Table 2. Here, columns 2–7 show the number of curves with k-value in the specified range, and the last column shows the median of k. We see that the medians ranged between 12.22 and 13.8. The number of curves

TABLE 2. Sharpness of the bound (3.5) on $\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1)$. Number of curves (out of 1000 random curves over $\mathbb{F}_p$ with irreducible L-polynomials over $\mathbb{Q}$) for which $\lfloor 9p/\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \rfloor = k$.

p	$k = 1-3$	$4-6$	$7-9$	$10-19$	$20-29$	$k \geq 30$	median
20011	98	144	157	280	102	219	12.87
40009	113	144	153	254	117	219	12.31
60013	108	169	119	250	105	249	13.61
80071	111	162	139	270	91	227	12.32
100003	112	149	139	252	89	259	12.57
140053	101	153	121	267	117	241	13.8
180001	113	135	116	295	105	236	13.37
220009	115	130	143	277	88	247	12.95
260011	99	135	130	280	111	245	13.58
300007	110	168	126	255	107	234	12.6
340027	115	152	145	258	103	227	12.26
380041	106	164	136	274	105	215	12.2
420001	123	148	125	254	115	235	12.99
460039	107	140	124	294	100	235	13.07
500029	100	137	139	292	93	239	13.53

TABLE 3. Expected speed-ups S_M when search for a_1 is restricted to $\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \leq 9p/M$. Here, $\mathcal{P}_M$ is the fraction of curves from Table 2 with $k \geq M$.

M	4	5	6	7	8	9	10	11	12	13	14
$\mathcal{P}_M$	0.89	0.84	0.79	0.75	0.7	0.66	0.61	0.57	0.53	0.5	0.47
S_M	1.62	1.65	1.64	1.63	1.57	1.53	1.45	1.38	1.31	1.26	1.21

with $k \leq 3$ seemed to be consistently about 10%, so the extreme values for the norm of a_1 (close to the theoretical bound) seem very rare. Thus, we can decrease the memory requirement and the running time of Algorithm 4.2 by searching for a_1 in a restricted range. The price to pay is that we are no longer guaranteed to find a value for a_1. Our algorithm is no longer deterministic, and we may have to check more than one curve to successfully determine the L-polynomial of a Picard curve. But with a careful choice of the search range for a_1, the expected running time will be shorter than with the deterministic variant, with the added advantage of smaller memory requirements.

To be more precise, consider the following. If we run the deterministic version of the algorithm, we expect, on average, to check 6.5 (out of 12) values of π before we find a_1. If we are to search for a_1 satisfying $\mathrm{N}_{\mathbb{Q}(\zeta_3)/\mathbb{Q}}(a_1) \leq 9p/M$, we expect to check 6.5 values of π before finding a_1 if we have a curve with $k \geq M$, and all 12 values of π otherwise. For each such π, this search is by a factor $\sqrt{M}$ faster than in the deterministic version. With $\mathcal{P}_M$ denoting the probability that a randomly chosen Picard curve has $k \geq M$, we expect to check $6.5+12(1/\mathcal{P}_M-1) = 12/\mathcal{P}_M - 5.5$ values of π. Thus the overall speed-up of this approach is

$$S_M = \frac{6.5}{12/\mathcal{P}_M - 5.5} \cdot \sqrt{M}.$$

Note that this assumes a serial setting, where the 12 values of π are computed consecutively on a single machine. In Table 3, for sample values of M we give the observed probabilities $\mathcal{P}_M$ where the sample space is the set of all 62,000 curves for which we calculated a_1 and k. The last line contains the corresponding expected speed-ups. Using a value for M of about 5 seems optimal, but larger values of M still yield an overall decrease in the running time, and also allow for the computation of larger examples because of the decrease in memory requirements. The observed crossover for the running time of the two variants occurs with $M = 20$.

9. CONCLUSION

We presented an algorithm in time and space $O(\sqrt{q})$ for point counting on Picard curves in large characteristic $\mathbb{F}_q$. It improves on the ordinary baby step–giant step algorithm but is still exponential. Combined with the precomputation of the Hasse-Witt matrix, it is possible to find cryptographically interesting curves over $\mathbb{F}_{p^3}$ in less than 10 minutes on a single laptop. We were also able to count the number of points for a Picard curve defined over a prime field of size $\approx 5 \cdot 10^{12}$, which gave a 39-digit (127-bit) prime group order, the largest example computed so far.

However, computing the cardinality of a cryptographically interesting Jacobian in the prime-field case still seems to be out of reach. For example, for a 156-bit

Jacobian $J_C(\mathbb{F}_p)$ we would have to work with a 52-bit prime p. Our algorithm then would require about $2^{30} \approx 10^9$ operations in $J_C(\mathbb{F}_p)$ for each of the 12 candidates for π, and a hash table with about $2^{29} \approx 5 \cdot 10^8$ entries. Taking 32 bits for the hash value of the divisors and 32 bits for the index, we obtain storage requirements of about 4 GB RAM. (Note that one could reduce the number of "baby steps" by a constant factor, but at the expense of increasing the number of "giant steps" by the same factor.) Very recently, Gaudry and Schost [GS04] designed a space-efficient variant of a baby step–giant step algorithm by Matsuo, Chao and Tsujii [MCT02] that likely can be modified for our application. This should enable us to compute even larger examples, and is future work.

Another possibility is the design of an algorithm that is less generic. For example, one could use a method to find the ℓ-torsion points of the Jacobian of a Picard curve for small ℓ [ELW].

Our algorithm can also be used for hyperelliptic curves of genus 3 with an automorphism of order 4. Let C be a hyperelliptic curve of genus 3 over a field $\mathbb{F}_q$ of characteristic $p \neq 2$ and of the form

$$y^2 = x^7 + ax^5 + bx^3 + cx,$$

where $a, b, c \in \mathbb{F}_q$. If $q \equiv 3 \pmod 4$, the L-polynomial of C splits. If $q \equiv 1 \pmod 4$ we proceed just as in the Picard case, but replacing the field $\mathbb{Q}(\zeta_3)$ by $\mathbb{Q}(i)$.

Acknowledgments

We would like to thank Roger Oyono and Stéphane Flon for providing electronic versions of their explicit formulas [FO04].

References

[Bau04] M. Bauer. The arithmetic of certain cubic function fields. *Mathematics of Computation*, 73:387–413, 2004. MR2034129 (2004k:11179)

[BGS04] A. Bostan, P. Gaudry, and E. Schost. Linear recurrences with polynomial coefficients and computation of the Cartier-Manin operator on hyperelliptic curves. In *Finite Fields and Applications Fq7*, volume 2948 of *Lecture Notes in Computer Science*, pages 40–58. Springer-Verlag, 2004. MR2092621

[Bou00] I. Bouw. The p-rank of curves and covers of curves. In *Courbes semi-stables et groupe fondamental en géomtrie algèbrique (Luminy, 1998)*, number 187 in Progr. Math., pages 267–277, Basel, 2000. MR1768105 (2001j:14042)

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976. MR0437208 (55:10141)

[ELW] K. Eisenträger, K. Lauter, and A. Weng. The ℓ-torsion points on the Jacobian of a Picard curve. In preparation.

[FO04] S. Flon and R. Oyono. Fast arithmetic on Jacobians on Picard curves. In *Public Key Cryptography – PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 55–68. Springer-Verlag, 2004. MR2095638

[GG03] P. Gaudry and N. Gurel. Counting points in medium characteristic using Kedlaya's algorithm. *Experimental Math.* 12:395–402, 2003. MR2043990 (2005b:11084)

[GS04] P. Gaudry and E. Schost. A low-memory parallel version of Matsuo, Chao and Tsujii's algorithm. In *Algorithmic Number Theory Symposium ANTS-VI*, volume 3076 of *Lecture Notes in Computer Science*, pages 208–222. Springer-Verlag, 2004.

[GH78] P. Griffiths and J. Harris. *Principles of algebraic geometry.* Wiley-Interscience, New York, 1978. MR0507725 (80b:14001)

[KW] K. Koike and A. Weng. Construction of CM Picard curves. *Mathematics of Computation*, 74:499–518, 2005. MR2085904

[Mag] Computational Algebra Group. *The Magma Computational Algebra System for Algebra, Number Theory and Geometry.* School of Mathematics and Statistics, University of Sydney. http://magma.maths.usyd.edu.au/magma.

[Man65] Ju. I. Manin. The Hasse-Witt matrix of an algebraic curve. *Trans. Amer. Math. Soc.*, 45:245-246, 1965.

[MCT02] K. Matsuo and J. Chao and S. Tsujii. An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In *Algorithmic Number Theory Symposium ANTS-V*, volume 2369 of *Lecture Notes in Computer Science*, pages 461–474. Springer-Verlag, 2002. MR2041104 (2005a:11089)

[Pil90] J. Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation*, 55(192):745-763, 1990. MR1035941 (91a:11071)

[PH78] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE-Transactions on Information Theory*, 24:106–110, 1978. MR0484737 (58:4617)

[Pol78] J. M. Pollard. Monte Carlo methods for index computation $\pmod p$. *Mathematics of Computation*, 32(143):918–924, 1978. MR0491431 (58:10684)

[Sar91] J. Estrada Sarlabous. On the Jacobian varieties of Picard curves defined over fields of characteristic $p > 0$. *Math. Nachr.*, 152:392–340, 1991. MR1121242 (93g:14033)

[Sch01] R. Scheidler. Ideal arithmetic and infrastructure in purely cubic function fields. *Journal de Théorie des Nombres de Bordeaux*, 13:609–631, 2001. MR1879675 (2002k:11209)

[SS] R. Scheidler and A. Stein. Computing the Class Number of a Cubic Function Field. Talk by A. Stein at the Conference in Number Theory in Honour of Professor H.C. Williams, Banff, May 2003.

[ST02a] A. Stein and E. Teske. Explicit bounds and heuristics on class numbers in hyperelliptic function fields. *Mathematics of Computation*, 71:837–861, 2002. MR1885633 (2002k:11210)

[ST02b] A. Stein and E. Teske. The parallelized Pollard kangaroo method in real quadratic function fields. *Mathematics of Computation*, 71:793–814, 2002. MR1885629 (2002k:11227)

[Sti93] H. Stichtenoth. *Algebraic Function Fields and Codes.* Springer, Berlin, 1993. MR1251961 (94k:14016)

[Tat66] J. Tate. Endomorphisms of abelian varieties over finite fields. *Invent. Math.*, 2:134–144, 1966. MR0206004 (34:5829)

[Thé03] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92. Springer-Verlag, 2003. MR2093253

[vOW99] P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12:1–28, 1999. MR1664774 (99i:94054)

[Wat69] W. C. Waterhouse. Abelian varieties over finite fields. *Ann. Sci. École Norm. Sup. (4)*, 2:521–560, 1969. MR0265369 (42:279)

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

E-mail address: mbauer@math.ucalgary.ca

University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, Canada N2L 3G1

E-mail address: eteske@uwaterloo.ca

Johannes Gutenberg-Universität, Fachbereich Mathematik, Staudingerweg 9, 55128 Mainz, Germany

E-mail address: weng@mathematik.uni-mainz.de