Counting Points on Hyperelliptic Curves over Finite Fields

Pierrick Gaudry[1] and Robert Harley[2]

[1] LIX, École Polytechnique,
91128 Palaiseau Cedex, France
[2] Projet Cristal, INRIA, Domaine de Voluceau - Rocquencourt,
78153 Le Chesnay, France

Abstract. We describe some algorithms for computing the cardinality of hyperelliptic curves and their Jacobians over finite fields. They include several methods for obtaining the result modulo small primes and prime powers, in particular an algorithm *à la* Schoof for genus 2 using Cantor's division polynomials. These are combined with a birthday paradox algorithm to calculate the cardinality. Our methods are practical and we give actual results computed using our current implementation. The Jacobian groups we handle are larger than those previously reported in the literature.

Introduction

In recent years there has been a surge of interest in algorithmic aspects of curves. When presented with any curve, a natural task is to compute the number of points on it with coordinates in some finite field. When the finite field is large this is generally difficult to do.

René Schoof gave a polynomial time algorithm for counting points on elliptic curves i.e., those of genus 1, in his ground-breaking paper [Sch85]. Subsequent improvements by Elkies and Atkin ([Sch95], [Mor95], [Elk98]) lowered the exponent to the point where efficient implementations became possible. After further improvements ([Cou96], [Ler97]) several implementations of the Schoof-Elkies-Atkin algorithm were actually written and very large finite fields can now be handled in practice ([Mor95], [Ver99]).

For higher genus, significant theoretical progress was made by Pila who gave a polynomial time algorithm in [Pil90] (see also [HI98]). However to date these methods have not been developed as extensively as the elliptic case. As a first step towards closing this gap it is fruitful to concentrate on low genus hyperelliptic curves, as these are a natural first generalization of elliptic curves and techniques used in the elliptic case can be adapted. Such techniques include Schoof-like methods and several others which all contribute to a practical algorithm.

We mention two possible applications of the ability to count points on low genus hyperelliptic curves. An early theoretical application was the proof that primality testing is in probabilistic polynomial time, [AH92]. A practical application results from the apparent difficulty of computing discrete logarithms in the Jacobian groups of these curves. In low genus, no sub-exponential algorithms are currently known, except for some very thin sets of examples ([Rüc99], [FR94]) and hence the Jacobian group of a random curve is likely to be suitable for constructing cryptosystems [Kob89]. To build such a cryptosystem, it is first desirable to check that the group order has a large prime factor since otherwise the logarithm could be computed in small subgroups [PH78].

We restrict ourselves to *odd* characteristic for simplicity. We will work with models of odd degree where arithmetic is analogous to that of imaginary quadratic fields. For the even degree alternative, which is similar to real quadratic fields, see the recent paper [ST99] which describes a birthday paradox algorithm optimized using an analogue of Shanks' infrastructure.

Our contribution contains several complementary approaches to the problem of finding the size of Jacobian groups, all of which have been implemented. By combining these approaches we have been able to count larger groups than previously reported in the literature.

The first approach is an efficient birthday paradox algorithm for hyperelliptic curves. We have filled in all the details required for a large-scale distributed implementation, although the basic idea has been known for 20 years. In our implementation we also use an optimized group operation for genus 2, in which we have reduced the number of field operations required.

The time taken grows as a small power of the field size and this algorithm, if used in isolation, would take a prohibitive amount of time to handle large groups such as those of cryptographic size. However our version of it can take advantage of prior information on the result modulo some integer. We elaborate various strategies for collecting as much of this information as possible.

We show that when the characteristic p is not too large, the result modulo p can be obtained surprisingly easily using the Cartier-Manin operator. It provides an elegant and self-contained method based on theoretical material proved in the 1960's.

To go further, we also extend Schoof's algorithm to genus 2 curves using Cantor's division polynomials. On the basis of previous outlines existing in the literature, but not directly implementable, we elaborated a practical algorithm and programmed it in Magma. For the case where the modulus is a power of 2, we are able to bypass computations with division polynomials and use a much faster technique based on formulae for halving in the Jacobian.

The combinations of these techniques has allowed us to count genus 2 groups with as many as 10^{38} elements.

We would particularly like to thank Éric Schost of the GAGE laboratory at École Polytechnique for helpful discussion concerning algebraic systems. Furthermore his assistance in computing Gröbner bases was invaluable and allowed us to compute group orders modulo larger powers of 2 than would otherwise have been possible.

We also thank François Morain for many constructive comments on this paper.

Prerequisites and Notations

We will take a concrete approach, concentrating on arithmetic and algorithmic aspects rather than more abstract geometric ones.

Let g be a positive integer and let $\mathbb{F}_q$ be the finite field of $q = p^n$ elements, where p is an odd prime. For our purposes, a *hyperelliptic curve of genus g* is the set of solutions (x, y) of the equation $y^2 = f(x)$, where $f(x)$ is a monic polynomial of degree $2g+1$ with coefficients in $\mathbb{F}_q$ and with distinct roots[1]. Note that the coordinates may be in the base field $\mathbb{F}_q$ or in an extension field.

When a point $P = (x_P, y_P)$ is on the curve $\mathcal{C}$, its *opposite* is the point $-P = (x_P, -y_P)$. A *divisor*[2] is a formal sum $D = \sum_i P_i$ of points on $\mathcal{C}$. Note that points may be repeated with some multiplicity in the sum. A *semi-reduced* divisor is a divisor with no two points opposite. Such a divisor with k points is said to have *weight k*. A *reduced* divisor is a semi-reduced divisor of weight $k \leq g$.

The *Jacobian*, denoted $\mathbf{J}$, is the set of reduced divisors. An important fact is that one can define an addition operation on reduced divisors which makes $\mathbf{J}$ into a group, whereas this is not possible on the curve itself directly. This group law is denoted by $+$ and will be described in the next section.

A convenient representation of reduced (and semi-reduced) divisors, due to Mumford [Mum84], uses a pair of polynomials $\langle u(x), v(x)\rangle$. Here $u(x) = \prod_i (x - x_i)$ and $v(x)$ interpolates the points P_i respecting multiplicities. More precisely $v = 0$ or $\deg v < \deg u$, and u divides $f - v^2$. We say that a semi-reduced divisor is *defined over* a field $\mathbb{F}$ when the coefficients of u and v are in $\mathbb{F}$ (even though the coordinates x_i and y_i may be in an extension field) and write $\mathbf{J}/\mathbb{F}$ for the set of such divisors.

Most reduced divisors have weight g. The set of those with strictly lower weight is called Θ. A divisor of weight 1 i.e., with a single point $P = (x_P, y_P)$, is represented by $\langle u(x), v(x)\rangle = \langle x - x_P, y_P\rangle$. The unique divisor of weight 0, $\mathcal{O} = \langle u(x), v(x)\rangle = \langle 1, 0\rangle$, is the neutral element of the addition law. Scalar multiplication by an integer l is denoted by:

$$[l]D = D + D + \cdots + D \ . \quad (l \text{ times}) \tag{1}$$

We say that D is an l–torsion divisor whenever $[l]D = \mathcal{O}$. The set of all l–torsion divisors, including those defined over extension fields, is denoted by $\mathbf{J}[l]$.

We concentrate particularly on genus-2 curves and in this case the divisors in $\mathbf{J} \setminus \Theta$ have the form:

$$D = \langle x^2 + u_1 x + u_2, v_0 x + v_1\rangle \ . \tag{2}$$

[1] Strictly speaking, this is the affine part of a smooth projective model. In genus 2 every curve is birationally equivalent to such a curve provided the base field is large enough.

[2] Strictly speaking, these are degree-0 divisors with the multiplicity of the point at infinity left implicit.

1 Group Law in the Jacobian

We will sketch the group law i.e., addition of reduced divisors, using the intuitive 'sum of points' notation and then describe efficient formulae for computing the law in genus 2 using Mumford's representation.

The computation of $D_1 + D_2$ can be viewed, at a high level of abstraction, as the following three steps:

- form a (non-reduced) divisor with all the points of D_1 and D_2,
- semi-reduce it by eliminating all pairs of opposite points,
- reduce it completely.

The third step is the only one that presents any difficulty[3]. When we reach it we have a semi-reduced divisor with at most $2g$ points. If there are g or fewer then no reduction is necessary but if there are more than g we reduce by a higher-genus analogue of the well known chord-and-tangent operation for elliptic curves.

1.1 Reduction Step

Fix $g = 2$ and, for the moment, consider a semi-reduced divisor R with 3 distinct points. The reduction of R is as follows.

Let $y = a(x)$ be the equation of the parabola (or perhaps line) interpolating the three points. The roots of $f - a^2$ are the abscissae of the intersections between the parabola and the curve. This is a quintic polynomial so there are five intersections (including multiplicities). We already know 3 of them, the points of R. Form a divisor S with the other two, and the result of the reduction is $-S$.

In the more frequent case where R has 4 points, choose an interpolating cubic (or lower degree) polynomial $a(x)$ instead. Then $f - a^2$ has degree 5 or 6 and we know 4 intersections. Form S with the others and the result is $-S$.

In cases where some points of R are repeated, the interpolation step is adjusted to ensure tangency to the curve with sufficient multiplicity. Also, in genus $g > 2$ the reduction step may need to be repeated several times.

In practice it would be inefficient to compute the group law this way using the representation of divisors as sums of points, since the individual points may be defined over extension fields. By using Mumford's notation we can work entirely in the field of definition of the divisors.

1.2 Group Law in Mumford's Notation

Cantor gave two forms of the group law using Mumford's notation in [Can87]. One was a direct analogue of Gauss's reduction of binary quadratic forms of

[3] In a more classical treatment the reduction would be described as choosing a representative for an equivalence class of degree 0 divisors modulo linear equivalence, where linearly equivalent divisors are those that differ by a principal divisor.

negative discriminant, the other an asymptotically fast algorithm for high genus making clever use of fast polynomial arithmetic.

We describe an efficient algorithm, carefully optimized to reduce the number of operations required. We find that in genus 2 doubling a divisor or adding two divisors both take 30 multiplication operations and 2 inversions, in general. Note for comparison that optimized elliptic curve operations typically take 3 or 4 multiplications and 1 inversion.

Space limits us to a brief description of the genus 2 doubling operation. Let $D = \langle u, v\rangle$. We cover the cases that may occur, in order of increasing complexity.

Simple case: If $v = 0$ the result is simply $\mathcal{O}$.

Weight 1: Here $u(x) = x - x_P$ and $v(x) = y_P$. The result is $\langle (x - x_P)^2, ax + b\rangle$, where $ax + b$ is the tangent line at P with $a = f'(x_P)/2y_P$ and $b = y_P - ax_P$.

Weight 2: Compute the resultant $r = u_2v_0^2 + v_1^2 - u_1v_0v_1$ of u and v.

> **Resultant 0:** If $r = 0$ then u and v have a root in common i.e., D has a point with ordinate 0. Isolate the other point by $x_P = -u_1 + v_1/v_0$, $y_P = v_1 + v_0x_P$ and return to the weight 1 case above.
>
> **General case:** Consider the fact that v is a square root of f modulo u. We can double the multiplicity of all points in D by using a Newton iteration to compute a square root modulo u^2.
> - Newton iteration: set $U = u^2$, and $V = (v + f/v)/2 \bmod U$,
> - Get 'other' roots: set $U = (f - V^2)/U$,
> - Make U monic,
> - Reduce V modulo U,
>
> The result is $\langle U, -V\rangle$.

Several observations help to optimize calculation with these formulae: after the first step, $V \equiv v \bmod u$; also the division by U in the second step is exact; not all coefficients of the polynomials are really needed; finally some multiplications can be avoided using Karatsuba's algorithm.

The general addition operation is similar to doubling although the Newton iteration is replaced by a little Chinese Remainder calculation and more cases need to be handled. Since the details are somewhat tedious, we give the resulting pseudo-code and sample C code at the following Web site:

`http://cristal.inria.fr/~harley/hyper/`

2 Frobenius Endomorphism

In this section we collect some useful results and quote them without proof. A starting point for the reader interested in pursuing this material is [IR82] and the references therein.

We first describe properties of the q–power Frobenius endomorphism $\phi(x) = x^q$. Note that it has no effect on elements of $\mathbb{F}_q$ but it becomes non-trivial in extension fields. This map extends naturally to points, by transforming their x and y coordinates. It extends further to divisors by acting point-wise.

Crucially, this latter action is equivalent to acting on each coefficient of the u and v polynomials in Mumford's notation. When a divisor is defined over $\mathbb{F}_q$, ϕ may permute its points but it leaves the divisor as a whole invariant.

2.1 Characteristic Polynomial

The ϕ operator acts linearly and has a characteristic polynomial of degree $2g$ with integer coefficients. In genus 2 it is known to have the form:

$$\chi(t) = t^4 - s_1 t^3 + s_2 t^2 - s_1 q t + q^2 \ , \tag{3}$$

so that $\chi(\phi)$ is the identity map on all of $\mathbf{J}$, in other words:

$$\forall P \in \mathbf{J},\ \ \phi^4(P) - [s_1]\phi^3(P) + [s_2]\phi^2(P) - [s_1 q]\phi(P) + [q^2]P = \mathcal{O} \ . \tag{4}$$

The so-called *Riemann hypothesis for curves*, on the roots of their zeta functions, was proved by Weil and implies that the complex roots of χ have absolute value $\sqrt{q}$. Hence, in genus 2 the following bounds apply: $|s_1| \le 4\sqrt{q}$ and $|s_2| \le 6q$.

2.2 Relations Between Frobenius and Cardinalities

The Frobenius is intimately related to the number of points on the curve and the number of divisors in $\mathbf{J}$, over the base field and its extensions.

First of all, knowledge of χ is equivalent to that of $\#\mathcal{C}/\mathbb{F}_{q^i}$ for $1 \le i \le g$. In genus 2 the following formulae relate them:

$$\#\mathcal{C}/\mathbb{F}_q = q - s_1 \quad \text{and} \quad \#\mathcal{C}/\mathbb{F}_{q^2} = q^2 - s_1^2 + 2s_2 \ . \tag{5}$$

Furthermore $\#\mathbf{J}/\mathbb{F}_q$ is completely determined by χ according to the formula $\#\mathbf{J}/\mathbb{F}_q = \chi(1)$. An important consequence is that the group order is constrained to a rather small interval, the Hasse-Weil interval:

$$\left\lceil (\sqrt{q}-1)^{2g} \right\rceil \le \#\mathbf{J}/\mathbb{F}_q \le \left\lfloor (\sqrt{q}+1)^{2g} \right\rfloor \ . \tag{6}$$

In the reverse direction, knowledge of $\#\mathbf{J}/\mathbb{F}_q$ almost determines χ for q large enough. For instance in genus 2, $(\#\mathbf{J}/\mathbb{F}_q) - q^2 - 1 = s_2 - s_1(q+1)$ and the bound on s_2 given above ensures that there are $O(1)$ possibilities.

3 Birthday Paradox Algorithm

To compute the group order $N = \#\mathbf{J}/\mathbb{F}_q$ exactly we search for it in the Hasse-Weil interval which has width w close to $4gq^{g-1/2}$. The first few coefficients s_i of χ can be computed by exhaustively counting points on the curve over $\mathbb{F}_{q_i}$. Doing so for $i \le I$ reduces the search interval to width $w = O(q^{g-(I+1)/2})$ but costs $O(q^I)$ (see [Elk98]). In genus 2 this is not useful and one simply takes $w = 2\lfloor 4(q+1)\sqrt{q} \rfloor$.

3.1 Computing the Order of the Group

Assume for the moment that we know how to compute the order n of a randomly chosen divisor D in $\mathbf{J}/\mathbb{F}_q$ (from now on the term "divisor" always refers to a reduced divisor). Writing e for the group exponent, we have $n \mid e$ and $e \mid N$ and thus N is restricted to at most $\lceil (w+1)/n \rceil$ possibilities. Usually $n \geq w$ and so N is completely determined.

It is possible for n to be smaller, though. In such a case we could try several other randomly chosen divisors, taking n to be the least common multiple of their orders and stopping if $n > w$. After a few tries n will converge to e and if $e > w$ the method terminates.

However in rare cases the exponent itself may be small, $e \leq w$. It is known that $\mathbf{J}/\mathbb{F}_q$ is the product of at most $2g$ cyclic groups and thus $e \geq \sqrt{q} - 1$ and in fact this lower bound can be attained.

It is possible to obtain further information by determining the orders of divisors in the Jacobian group of the quadratic twist curve, but even this may not be sufficient. We do not yet have a completely satisfactory solution for such a rare case, however we mention that the Weil pairing may provide one.

3.2 Computing the Order of One Divisor

To determine the order n of an arbitrary divisor D we find some multiple of n, factor it and search for its smallest factor d such that $[d]D = \mathcal{O}$.

There are certainly multiples of n in the search interval (since the group order is one such) and we can find one of them using a birthday paradox algorithm, in particular a distributed version of Pollard's lambda method [Pol78] with distinguished points. For a similar Pollard rho method see [vOW99].

Since the width of the search interval is w, we expect to determine the multiple after $O(\sqrt{w})$ operations in the Jacobian. By using distinguished points and distributing the computation on M machines, this takes negligible space and $O(\sqrt{w}(\log q)^2/M)$ time[4].

The birthday paradox algorithm is as follows.

- Choose some distinguishing characteristic.
- Choose a hash function h that hashes divisors to the range 0..19, say.
- Pick 20 random step lengths $l_i > 0$ with average roughly $M\sqrt{w}$,
- Precompute the 20 divisors $D_i = [l_i]D$.
- Precompute $E = [c]D$.

Here c is the center of the search interval. In genus 2 $c = q^2 + 6q + 1$. The calculation then consists of many 'chains' of iterations run on M client machines:

- Pick a random $r < w$ and compute $R = [r]D$.
- Pick a random bit b and if it is 1 set R to $R + E$.
- While R is not distinguished, set $r := r + l_{h(R)}$ and $R := R + D_{h(R)}$.

[4] Using classical algorithms for field arithmetic.

– Store the distinguished R on a central server along with r and b.

The distinguishing feature must be chosen to occur with probability significantly less than $\sqrt{w}/M$, say 50 times less. Thus each chain takes about $\sqrt{w}/M/50$ steps and has length about $w/50$.

Note that chains with $b = 0$ visit many pseudo-random divisors in the set $S_1 = \{[r]D \mid 0 \leq r < w\}$ and a few with larger r. Chains with $b = 1$ visit many divisors in $S_2 = \{E + [r]D \mid 0 \leq r < w\}$ and a few with larger r. However the choice of E guarantees that the intersection $I = S_1 \cap S_2$ contains at least $w/2$ divisors.

Now after a total of $O(\sqrt{w})$ steps have been performed, $O(\sqrt{w})$ divisors have been visited and $O(\sqrt{w})$ of them are expected to be in I. Then the birthday paradox guarantees a significant chance that a *useful collision* occurs i.e., that the same divisor R is visited twice with different bits b. Shortly afterwards a useful collision of distinguished points is detected at the server, between R_0 and R_1 say.

Therefore $r_0 \equiv c + r_1$ modulo n and finally $c + r_1 - r_0$ is the desired multiple of n.

3.3 Beyond the Birthday Paradox

To handle larger examples than is possible with the birthday paradox algorithm alone, we precompute the Jacobian order modulo some integer. If N is known modulo m then the search for a multiple of a divisor's order can be restricted to an arithmetic progression modulo m, rather than the entire search interval[5]. In this way the expected number of operations can be reduced by a factor $\sqrt{m}$.

The algorithm outlined above needs to be modified as follows (we can assume that m is much smaller than w since otherwise no birthday paradox algorithm would be required!).

– Increase the frequency of the distinguishing characteristic by a factor $\sqrt{m}$.
– The step lengths must be multiples of m chosen with average length $M\sqrt{wm}$.
– Replace E with $[z]D$ where z is nearest c such that $z \equiv N \bmod m$.

To compute N modulo m with m as large as possible, we will first compute it modulo small primes and prime powers using various techniques explained in the next few sections. Then the Chinese Remainder Theorem gives N modulo their product.

To date this use of local information has speeded up the birthday paradox algorithm by a significant factor in practice. It should be pointed out however that while the birthday paradox algorithm takes exponential time, the Schoof-like algorithm described below takes polynomial time. Hence it can be expected that for future calculations with very large Jacobians, the Schoof part will provide most of the information.

[5] Note that we could also take advantage of partial information that restricted N to several arithmetic progressions modulo m.

4 Cartier-Manin Operator and Hasse-Witt Matrix

We propose a method for calculating the order of the Jacobian modulo the characteristic p of the base field, by using the so-called *Cartier-Manin operator* and its concrete representation as the *Hasse-Witt matrix* (see [Car57]). In the case of hyperelliptic curves, this $g \times g$ matrix can be computed by a method given in [Yui78] which generalizes the computation of the Hasse invariant for elliptic curves. Yui's result is as follows:

Theorem 1. *Let $y^2 = f(x)$ with $\deg f = 2g+1$ be the equation of a genus g hyperelliptic curve. Denote by c_i the coefficient of x^i in the polynomial $f(x)^{(p-1)/2}$. Then the Hasse-Witt matrix is given by*

$$A = (c_{ip-j})_{1 \le i,j \le g} \quad . \tag{7}$$

In [Man65], Manin relates it to the characteristic polynomial of the Frobenius modulo p. For a matrix $A = (a_{ij})$, let $A^{(p)}$ denote the elementwise p–th power i.e., (a_{ij}^p). Then Manin proved the following result:

Theorem 2. *Let $\mathcal{C}$ be a curve of genus g defined over a finite field $\mathbb{F}_{p^n}$. Let A be the Hasse-Witt matrix of $\mathcal{C}$, and let $A_\phi = AA^{(p)} \cdots A^{(p^{n-1})}$. Let $\kappa(t)$ be the characteristic polynomial of the matrix A_ϕ, and $\chi(t)$ the characteristic polynomial of the Frobenius endomorphism. Then*

$$\chi(t) \equiv (-1)^g t^g \kappa(t) \mod p \quad . \tag{8}$$

Now it is straightforward to compute $\chi(t)$ modulo the characteristic p and hence $\#\mathbf{J}/\mathbb{F}_q \bmod p$, provided that p is not too large (say at most 100000). Note that this is a very efficient way to get information on the Jacobian order, particularly when p is moderately large. Such a situation can occur in practice with fields chosen, for implementation reasons, to be of the form $\mathbb{F}_{p^n}$ with p close to a power of 2 such as $p = 2^8 - 5$ or $p = 2^{16} - 15$.

5 Algorithm *à la* Schoof

In this section we describe a polynomial time algorithm *à la* Schoof for computing the cardinality of $\mathbf{J}/\mathbb{F}_q$ in genus 2. This algorithm follows theoretical work of Pila [Pil90] and Kampkötter [Kam91]. We make extensive use of the division polynomials described by Cantor [Can94].

5.1 Hyperelliptic Analogue of Schoof's Algorithm

The hyperelliptic analogue of Schoof's algorithm consists of computing χ modulo some small primes l by working in $\mathbf{J}[l]$. Once this has been done, modulo sufficiently many primes (or prime powers), then χ can be recovered exactly by the Chinese Remainder Theorem. From the bounds on s_i above, it suffices to consider $l = O(\log q)$. In practice we use a few small l, determine χ modulo their product, and use this information to optimize a birthday paradox search as described previously.

Let l be a prime power co-prime with the characteristic. Then the subgroup of l–torsion points has the structure $\mathbf{J}[l] \cong (\mathbb{Z}/l\mathbb{Z})^{2g}$. Moreover, the Frobenius acts linearly on this subgroup and Tate's theorem [Tat66] states that the characteristic polynomial of the induced endomorphism is precisely the characteristic polynomial of the Frobenius endomorphism on $\mathbf{J}$ with its coefficients reduced modulo l. Hence by computing the elements of $\mathbf{J}[l]$ and the Frobenius action on them, we can get the characteristic polynomial modulo l.

The following lemma due to Kampkötter simplifies the problem.

Lemma 1. *If l is an odd prime power, then the set $\mathbf{J} \setminus \Theta$ contains a $\mathbb{Z}/l\mathbb{Z}$–basis of $\mathbf{J}[l]$.*

Thus the Frobenius endomorphism on $\mathbf{J}[l]$ is completely determined by its action on $\mathbf{J}[l] \setminus \Theta$.

Let $D = \langle x^2 + u_1 x + u_2, v_0 x + v_1 \rangle$ be a divisor in $\mathbf{J} \setminus \Theta$, then the condition $[l]D = \mathcal{O}$ can be expressed by a finite set of rational equations in u_1, u_2, v_0, v_1. More precisely, there exists an ideal I_l of the polynomial ring $\mathbb{F}_q[U_1, U_2, V_0, V_1]$ such that D lies in $\mathbf{J}[l] \setminus \Theta$ if and only if $f(u_1, u_2, v_0, v_1) = 0$ for all polynomials f in (a generating set of) the ideal I_l. In [Kam91], Kampkötter gives explicit formulae for multivariate polynomials generating I_l.

From now on, we can represent a generic element of $\mathbf{J}[l] \setminus \Theta$ by the quotient ring $\mathbb{F}_q[U_1, U_2, V_0, V_1]/I_l$. The Frobenius action can be computed for this element and it is possible to find its minimal polynomial by brute force. The characteristic polynomial is then easy to recover (at least in the case where l is a prime) and we are done. This method due to Pila and Kampkötter has polynomial-time complexity, however it involves arithmetic on ideals which requires time-consuming computations of Gröbner bases. In the following we propose another method which avoids the use of ideals.

5.2 Cantor's Division Polynomials

In [Can94], Cantor defined *division polynomials* of hyperelliptic curves, generalizing the elliptic case, and gave an efficient recursion to build them.

These polynomials are closely related to Kampkötter's ideal I_l, but they allow a Schoof-like algorithm to work mostly with one instead of four variables. An approximate interpretation of the phenomenon is that the division polynomials lead to a representation of I_l directly computed in a convenient form (almost a Gröbner basis for a lexicographical order).

Cantor's construction provides 6 sequences of polynomials $d_0^{(l)}, d_1^{(l)}, d_2^{(l)}$ and $e_0^{(l)}, e_1^{(l)}, e_2^{(l)}$ such that for divisors $P = \langle x - x_P, y_P \rangle$ of weight 1 in general position, we get

$$[l]P = \left\langle x^2 + \frac{d_1^{(l)}(x_P)}{d_0^{(l)}(x_P)} x + \frac{d_2^{(l)}(x_P)}{d_0^{(l)}(x_P)}, y_P \left(\frac{e_1^{(l)}(x_P)}{e_0^{(l)}(x_P)} x + \frac{e_2^{(l)}(x_P)}{e_0^{(l)}(x_P)} \right) \right\rangle \quad . \tag{9}$$

The degrees of these division polynomials are

d_0	d_1	d_2	e_0	e_1	e_2
$2l^2-1$	$2l^2-2$	$2l^2-3$	$3l^2-2$	$3l^2-2$	$3l^2-3$

By lemma 1 it is sufficient to consider divisors $D \notin \Theta$. In order to multiply $D = \langle u(x), v(x)\rangle$ by l we express it as a sum of two divisors of weight 1 i.e., we write $D = P_1 + P_2$. These divisors are given by $P_1 = \langle x - x_1, y_1\rangle$ and $P_2 = \langle x - x_2, y_2\rangle$ where x_1 and x_2 are the roots of $u(x)$ and $y_i = v(x_i)$. Clearly $[l]D = [l]P_1 + [l]P_2$.

The divisor D is an l–torsion divisor if and only if $[l]P_1$ and $[l]P_2$ are opposite divisors. This last condition is converted into a condition on the polynomial representations $[l]P_1 = \langle u_{P_1}(x), v_{P_1}(x)\rangle$ and $[l]P_2 = \langle u_{P_2}(x), v_{P_2}(x)\rangle$. Indeed two divisors are opposite if their u polynomials are equal and their v polynomials are opposite. Hence the elements of $\mathbf{J}[l] \setminus \Theta$ are characterized by a set of rational equations in the 4 indeterminates x_1, x_2, y_1, y_2, two of them involving only the two indeterminates x_1 and x_2.

Thus we get an ideal similar to I_l represented in a convenient form: we can eliminate x_2 with the two bivariate equations by computing some resultants, then we have a univariate polynomial in x_1 and for each root x_1 it is not difficult to recover the corresponding values of x_2, y_1 and y_2.

5.3 Details of the Algorithm

Next we explain the computation of the characteristic polynomial modulo a fixed prime power l. Here we will assume that l is odd (the even case discussed in the next section).

Building an Elimination Polynomial for x_1. We first compute Cantor's l–division polynomials. We refer to the original paper [Can94] for the recursion formulae and the proof of the construction. This phase takes negligible time compared to what follows.

The second step is to eliminate x_2 in the two bivariate equations. The system looks like

$$\begin{cases} E_1(x_1, x_2) = d_1(x_1)d_2(x_2) - d_1(x_2)d_2(x_1) = 0 \ , \\ E_2(x_1, x_2) = d_0(x_1)d_2(x_2) - d_0(x_2)d_2(x_1) = 0 \ . \end{cases} \tag{10}$$

The polynomial $(x_1 - x_2)$ is clearly a common factor of E_1 and E_2, and this factor is a parasite: it does not lead to a l–torsion divisor[6]. We throw away this factor and consider the new reduced system, still denoting the two equations by

[6] If there is another common factor of E_1 and E_2, we have to throw it away. This occurs when a non trivial l–torsion divisor is in Θ. The values for the degrees assume that we are in the generic case.

$E_1(x_1,x_2)$ and $E_2(x_1,x_2)$. Then we eliminate x_2 by computing the following resultant

$$R(x_1) = \mathrm{Res}_{x_2}(E_1(x_1,x_2), E_2(x_1,x_2)) = 0 \ . \tag{11}$$

We can then note that $R(x_1)$ is divisible by some high power of $d_2(x_1)$. Indeed, if $d_2(x_1) = 0$ then the expressions E_1 and E_2 have common roots (at the roots of $d_2(x_2)$). The power of d_2 in R is $\delta = 2l^2 - 2$. We assume that the base field is large enough and we specialize the system at many distinct values for x_1. Substituting ξ_i for x_1, the system becomes two *univariate* polynomials in x_2, for which we compute the resultant r_i. With enough pairs (ξ_i, r_i) i.e., one more than a bound on the degree of $\tilde{R}(x_1) = R(x_1)/(d_2(x_1))^\delta$, we can recover $\tilde{R}(x_1)$ by interpolation. Knowing the degrees of d_0, d_1, d_2, it is easy to get

$$\deg \tilde{R}(x_1) = 4l^4 - 10l^2 + 6 \ . \tag{12}$$

Eliminating the Parasites (optional). As previously mentioned there are l^4 divisors of l–torsion and thus the degree of $\tilde{R}(x_1)$ is too high by a factor 4. This means that there are still a lot of parasite factors, due to the fact that we only took conditions on the abscissae x_1, x_2 into account and imposed nothing on the ordinates y_1, y_2. Two strategies can be used: we can decide to live with these parasites and go on to the next step or we can compute another resultant to eliminate them (and get a polynomial of degree $l^4 - 1$). The choice depends on the relative speeds of the resultant computation and the root-finding algorithm.

In order to eliminate the parasites we construct a third equation $E_3(x_1,x_2)$, coming from the fact that the ordinates of $[l]P_1$ and $[l]P_2$ are opposite. We write that the coefficients are opposite,

$$\begin{cases} y_1 \frac{e_1(x_1)}{e_0(x_1)} = -y_2 \frac{e_1(x_2)}{e_0(x_2)} \\ y_1 \frac{e_2(x_1)}{e_0(x_1)} = -y_2 \frac{e_2(x_2)}{e_0(x_2)} \end{cases} , \tag{13}$$

and this system implies that $E_3(x_1,x_2) = e_1(x_1)e_2(x_2) - e_1(x_2)e_2(x_1) = 0$.

Taking the resultant between E_1 and E_3, we get a polynomial $\tilde{S}(x_1)$ of degree $12l^4 - 30l^2 + 18$ whose GCD with $\tilde{R}(x_1)$ is of degree $l^4 - 1$ (in general, a few parasites may remain in rare cases). We still denote this GCD by $\tilde{R}(x_1)$ for convenience.

Recovering the Result Modulo l. To find the result we factor $\tilde{R}(x_1)$ and, for each irreducible factor, we construct an extension of $\mathbb{F}_q$ using this factor to get a root X_1 of $\tilde{R}(x_1)$. Then we substitute this root into E_1 and E_2 and recover the corresponding root X_2. Using the equation of the curve we get the ordinates Y_1 and Y_2, which may be in a quadratic extension. We get the two divisors $P_1 = \langle x - X_1, Y_1\rangle$ and $P_2 = \langle x - X_2, Y_2\rangle$ and check whether $[l](P_1 + P_2) = \mathcal{O}$ or $[l](P_1 - P_2) = \mathcal{O}$. If neither holds, then we started from a parasite solution and we try another factor of $\tilde{R}(x_1)$. In the favorable case we get an l–torsion divisor D with which we check the Frobenius equation. To do so we compute

$$[s_1]\phi^3(D) + [qs_1 \bmod l]\phi(D) \ , \tag{14}$$

for every $s_1 \in [0, l-1]$ and

$$\phi^4(D) + [s_2]\phi^2(D) + [q^2 \bmod l]D \ , \tag{15}$$

for every $s_2 \in [0, l-1]$. We only keep the pairs (s_1, s_2) for which these are equal.

If there is only one pair (s_1, s_2) left, or if there are several pairs all leading to the same value for the cardinality modulo l, then it is not necessary to continue with another factor. Thus it is usually not necessary to have a complete factorization of $\tilde{R}(x_1)$ and the computation is faster if one starts with irreducible factors of smallest degree.

We summarize the above in the following:

Algorithm. Computation of $\#\mathbf{J}/\mathbb{F}_q$ modulo l.

1. Compute $\tilde{R}(x_1)$.
2. Find a factor of $\tilde{R}(x_1)$ of smallest degree.
3. Build P_1 and P_2 with this factor.
4. Check if $P_1 + P_2$ or $P_1 - P_2$ is an l–torsion divisor. If so call it D, else go back to step 2.
5. For each remaining pair (s_1, s_2), check the Frobenius equation for D.
6. Compute the set of possible values of $\#\mathbf{J}/\mathbb{F}_q$ from the remaining values of (s_1, s_2). If there are several values left, go back to step 2. If there is just one, return it.

5.4 Complexity

We evaluate the cost of this algorithm by counting the number of operations in the base field $\mathbb{F}_q$. We neglect all the $\log^\alpha l$ factors, and denote by $M(x)$ the number of field operations required to multiply two polynomials of degree x.

The first step requires $O(l^4)$ resultant computations, each of which can be done in $M(l^2)$ operations, and the interpolation of a degree $O(l^4)$ polynomial which can be done in $M(l^4)$ operations. For the analysis of the remaining steps, we will denote by d the degree of the smallest factor of $\tilde{R}(x_1)$ that allows us to conclude. We assume moreover that the most costly part of the factorization is the distinct degree factorization (which is the case if d is small and if the number of factors of degree d is not too large). Then the cost of finding the factor is $O(d\log(q))M(l^4)$. Thereafter the computation relies on manipulations of polynomials of degree d and the complexity is $O(l + \log(q))M(d)$, where l reflects the l possible values of s_1 and of s_2 and $\log(q)$ reflects the Frobenius computations. Hence the (heuristic) overall cost for the algorithm is

$$O(l^4)M(l^2) + O(d \log q)M(l^4) + O(l^2 + \log q)M(d) \tag{16}$$

operations in the base field.

Now we would like to obtain a complexity for the whole Schoof-like algorithm. For that we will keep only the primes l for which $d = O(l)$; this should occur heuristically with a fixed probability (this is an analogue of 'Elkies primes' for elliptic curves). Then we have to use a set of $O(\log q)$ primes l, each of them satisfying $l = O(\log q)$. Moreover we will assume fast polynomial arithmetic and thus $M(x) = O(x)$ (ignoring logarithmic factors). Hence the cost of the algorithm is heuristically $O(\log^7 q)$ operations in $\mathbb{F}_q$. Each operation can be performed in $O(\log^2 q)$ bit operations using classical arithmetic and we get that the complexity of the Schoof-like algorithm is $O(\log^9 q)$.

Remark. This analysis is heuristic, but one could obtain a rigorous proof that the algorithm runs in polynomial time. The algorithm could also be made deterministic by avoiding polynomial factorizations. However in both cases the exponent would be higher than 9.

6 Lifting the 2-Power Torsion Divisors

In this section, we will show how to obtain some information on the $\#\mathbf{J}/\mathbb{F}_q$ modulo small powers of 2. Factoring f gives some information immediately. To go further we iterate a method for 'halving' divisors in the Jacobian. This quickly leads to divisors defined over large extensions, so that the run-time grows exponentially. In practice we can use this technique to obtain partial information modulo 256, say.

The divisors of order 1 or 2 are precisely the $D = \langle u(x), 0\rangle$ for which $u(x)$ divides $f(x)$ and is of degree at most g. When f has n irreducible factors, then it has 2^n factors altogether. Exactly half of them have degree at most g, since f is square-free of degree $2g+1$. Hence the number of such divisors is 2^{n-1}, and $2^{n-1} \mid \#\mathbf{J}/\mathbb{F}_q$. Furthermore, when f is irreducible then the 2–part is trivial and $\#\mathbf{J}/\mathbb{F}_q$ is odd.

6.1 Halving in the Jacobian

Let $D = \langle u(x), v(x)\rangle$ be a divisor different from $\mathcal{O}$. We would like to find a divisor Δ such that $[2]\Delta = D$. Note that there are 2^{2g} solutions, any two of which differ by a 2–torsion divisor. In general, Δ is defined over an extension of the field of definition of D.

Writing $\Delta = \langle \tilde{u}(x), \tilde{v}(x)\rangle$, we derive a rational expression for the divisor $[2]\Delta$ using the formulae of section 1. Then equating this expression with D, we get a set of $2g$ polynomial equations in $2g$ indeterminates $\tilde{u}_i$ and $\tilde{v}_i$ with $2g$ parameters u_i and v_i. There are g^2 such systems corresponding to the different possible weights of D and Δ.

We consider the most frequent case where D and Δ are both of weight g. The corresponding system has at most 2^{2g} solutions and these can be obtained by constructing a Gröbner basis for a lexicographical order, factoring the last polynomial in the basis and propagating the solution to the other polynomials. All this can be done in time polynomial in $\log q$ provided that the divisor D we are dealing with is defined over an extension of bounded degree of $\mathbb{F}_q$.

In order to speed up the computations in the case where D is defined over a large extension, we can avoid repeated Gröbner-basis computations and instead compute a single *generic* Gröbner basis for the system, where the coefficients of D are parameters. As the halving is algebraic over $\mathbb{F}_q$ (because the curve is defined over $\mathbb{F}_q$), the generic basis is also defined over $\mathbb{F}_q$. After this computation we can halve any divisor D, even when defined over a large extension, by plugging its coefficients into the generic basis to get the specialized one.

We are indebted to Eric Schost who kindly performed the construction of this generic Gröbner basis for the curves we studied [Sch]. For his construction, he made use of the `Kronecker` package [Lec99] written by Grégoire Lecerf. This package behaves very well on these types of problem (lifting from specialized systems to generic ones), and it is likely that we would not have been able to do this lifting by using classical algorithms for Gröbner-basis computations.

Example. Let $\mathcal{C}$ be defined by

$$y^2 = x^5 + 1597\,x^4 + 1041\,x^3 + 5503\,x^2 + 6101\,x + 1887 \ , \tag{17}$$

over the finite field $\mathbb{F}_p$ with $p = 10^{17} + 3$. We will search for all *rational* 2–power torsion divisors i.e., those defined over $\mathbb{F}_p$. Two irreducible factors of $f(x)$ have degree at most 2, they are

$$f_1 = x + 28555025517563816 \quad \text{and} \quad f_2 = x + 74658844563359755 \ ,$$

Thus there are three rational divisors of order two: $P_1 = \langle f_1, 0\rangle$, $P_2 = \langle f_2, 0\rangle$ and $P_1 + P_2$. The halving method applied to P_1 finds four rational divisors of order 4. They are $\langle u, v\rangle$ and $\langle u, -v\rangle$ where:

$$\begin{aligned} u &= x^2 + 1571353025997967\,x + 12198441063534328 \\ v &= 32227723250469108\,x + 68133247565452990 \end{aligned}$$

and:

$$\begin{aligned} u &= x^2 + 70887725815800572\,x + 94321182398888258 \\ v &= 42016761890161508\,x + 3182371156137467 \ . \end{aligned}$$

There are 16 solutions altogether but the others are in extension fields (the Gröbner bases are too large to include them here!) Applying the method to P_2 and to $P_1 + P_2$ finds no further rational 4–torsion divisors. By continuing in the same manner one finds 8 divisors of order 8, 16 of order 16, 32 of order 32 and no more. Thus the 2–part of the rational Jacobian is of the form $(\mathbb{Z}/2) \times (\mathbb{Z}/32)$ and hence $\#\mathbf{J}/\mathbb{F}_p \equiv 64 \bmod 128$.

This type of exhaustive search in the base field determines the exact power of 2 dividing $\#\mathbf{J}/\mathbb{F}_p$. In the next section we show how to find information modulo larger powers of 2.

6.2 Algorithm for Computing $\#\mathbf{J}/\mathbb{F}_q \bmod 2^k$

Next we go into extension fields to find some 2^k–torsion divisors and we substitute them into χ, the characteristic equation of the Frobenius endomorphism, to determine values of its coefficients modulo 2^k and hence the value of $\#\mathbf{J}/\mathbb{F}_q$ mod 2^k, for increasing k.

Algorithm (for $g = 2$).

1. Factor f to find a 2–torsion divisor. Halve it to get a 4–torsion divisor D_4.
2. Find the pair (s_1, s_2) mod 4 for which $\chi(D_4) = \mathcal{O}$. Set k to 2.
3. Compute the generic Gröbner basis for halving (weight 2) divisors in the given Jacobian.
4. Build a 2^{k+1}–torsion divisor $D_{2^{k+1}}$ by substituting the coefficients of D_{2^k} in the system, computing a root of the eliminating polynomial in an extension of minimal degree, and propagating it throughout the system.
5. For each pair (s_1, s_2) mod 2^{k+1} compatible with the previously found pair modulo 2^k, plug $D_{2^{k+1}}$ into χ and find the pair for which $\chi(D_{2^{k+1}}) = \mathcal{O}$.
6. Set $k = k + 1$, and go back to Step 4.

Note that this is an idealized description of the algorithm. In fact there will frequently be several pairs (s_1, s_2) remaining after checking the Frobenius equation for one 2^k–torsion divisor. We can eliminate false candidates by checking with other 2^k–torsion divisors. It can be costly to eliminate all of them when the required divisors are in large extensions; an alternative strategy is to continue and expect the false candidates to be eliminated later using 2^{k+1}–torsion divisors.

In this algorithm, we could skip step 3 and compute specific Gröbner bases at each time in step 4. However, the generic Gröbner basis is more efficient and allows one to perform one or two extra iterations for the same run-time.

7 Combining these Algorithms — Practical Results

We have implemented all these algorithms and tested their performance for real computation. Some of them were written in the C programming language, and others were implemented in the Magma computer algebra system [BC97].

7.1 Prime Field

In the case where the curve is defined over a prime field $\mathbb{F}_p$, where p is a large prime, we use all the methods described in previous sections except for Cartier-Manin. We give some data for a 'random' curve for which we computed the cardinality of the Jacobian. Let the curve $\mathcal{C}$ be defined by

$$\begin{aligned} y^2 = x^5 &+ 3141592653589793238\ x^4 + 4626433832795028841\ x^3 \\ &+ 9716939937510582097\ x^2 + 4944592307816406286\ x \\ &+ 2089986280348253421\ , \end{aligned} \tag{18}$$

over the prime field of order $p = 10^{19} + 51$. The cardinality of its Jacobian is

$$\#\mathbf{J} = 99999999982871020671452277000281660080\ , \tag{19}$$

and the characteristic polynomial of the Frobenius has coefficients:

$$s_1 = 1712898036 \quad \text{and} \quad s_2 = 1145227708935235535 0\ .$$

The first step of this computation is to factor $f(x)$. It has 3 irreducible factors, thus we already know that $\#\mathbf{J} \equiv 0 \bmod 4$.

The second step is to lift the 2–power torsion divisors. The computation of the generic halving Gröbner basis (done by E. Schost) took about one hour on an Alpha workstation. Then we lifted the divisors several times and checked the Frobenius equation. In the following table we give the degree of the extension where we found a 2^k–torsion divisor, and the information on $\#\mathbf{J}$ that we got (timings on a Pentium 450).

$\#\mathbf{J}$	deg of ext	$\#\mathbf{J}$	deg of ext	time
0 mod 2	1	16 mod 32	16	
0 mod 4	1	48 mod 64	32	
0 mod 8	4	48 mod 128	64	5000 sec
0 mod 16	8	176 mod 256	128	9 hours

The next step is to perform the Schoof–like algorithm. We did so for the primes $l \in \{3, 5, 7, 11, 13\}$. The following table gives the degree of the polynomial $\tilde{R}(x_1)$ for each l, and the smallest extension where we found an l–torsion divisor (timings on a Pentium 450).

l	degree of $\tilde{R}(x_1)$	degree of ext	$\#\mathbf{J}$	time
3	240	2	1 mod 3	1200 sec
5	2256	1	0 mod 5	300 sec
7	9120	6	4 mod 7	12 hours
11	57360	1	0 mod 11	19 hours
13	112560	7	9 mod 13	205 hours

The run-time for $l = 3$ is surprisingly large in this table. For our curve, an unlucky event occurs, which becomes rare as l increases. Indeed, after testing the Frobenius equation for *all* the 3–torsion divisors several candidates (s_1, s_2) still remain, yielding several possibilities for $\#\mathbf{J}$ mod 3. What this means is that the minimal polynomial of ϕ is not the characteristic polynomial. Each remaining candidate for (s_1, s_2) gives a multiple of the minimal polynomial. By taking their GCD we obtain the exact minimal polynomial, from which we can deduce the characteristic polynomial[7] and $\#\mathbf{J}$ mod 3.

In our case, there are 3 pairs left after testing all the 3–torsion points, leading to the following candidates for $\#\mathbf{J}$ mod 3.

(s_1, s_2) mod 3	$\#\mathbf{J}$ mod 3	$\chi(t)$ mod 3
$(0, 2)$	1	$t^4 - t^2 + 1$
$(1, 2)$	2	$t^4 - t^3 - t^2 - t + 1$
$(2, 2)$	0	$t^4 + t^3 - t^2 + t + 1$

[7] See [Kam91] for more about this.

The third case is impossible because if $\#\mathbf{J} \equiv 0 \bmod 3$ then we would have found a rational 3–torsion divisor earlier. In order to decide between the two first cases we determine the minimal polynomial, which is t^2+1 and thus the characteristic polynomial must be $(t^2+1)^2$ and finally $\#\mathbf{J} \equiv 1 \bmod 3$.

However to do this we have to build all the 3–torsion divisors. This explains why the running time is higher than for $l = 5$, where we found a rational 5–torsion divisor and immediately deduced that $\#\mathbf{J} = 0 \bmod 5$.

The final step is the birthday paradox computation. The width of the Hasse-Weil interval is roughly 2.5×10^{29}. The search space is reduced by a factor $2^8 \times 3 \times 5 \times 7 \times 11 \times 13 = 3843840$ leaving 6.6×10^{22} candidates. The search was performed on ten Alpha workstations working in parallel and calculated 5×10^{11} operations in the Jacobian. On a single 500 MHz workstation, this computation would have taken close to 50 days.

7.2 Non-prime Fields

Let $\mathcal{C}$ be a genus 2 curve defined over $\mathbb{F}_{p^n}$, where p is a small odd prime. We assume that $\mathcal{C}$ is not defined over a small subfield, for in that case it is easy to compute $\chi(t)$ using a theorem due to Weil.

Here the first step is to use Cartier-Manin to get $\chi(t) \bmod p$ quickly and then continue as before, except that we avoid $l = p$ in the Schoof part.

Examples: We did not try to build big examples, however we give two medium ones. For the first, let the curve $\mathcal{C}$ be defined by

$$y^2 = x^5 + x^4 + x^3 + x^2 + tx + 1 \ , \tag{20}$$

over the finite field $\mathbb{F}_{3^{30}} = \mathbb{F}_3[t]/(t^{30}+t-1)$. The cardinality of its Jacobian is

$$\#\mathbf{J} = 42391156018493425614913594804 \ . \tag{21}$$

The second example illustrates the advantage given by Cartier-Manin in a favorable case where $p = 2^{16} - 15$. Let the curve $\mathcal{C}$ be defined by

$$y^2 = x^5 + x^4 + x^3 + x^2 + x + t \ , \tag{22}$$

over the finite field $\mathbb{F}_{p^4} = \mathbb{F}_p[t]/(t^4 - 17)$. The Cartier-Manin computation gave us $\#\mathbf{J} \equiv 58976 \bmod p$ in 17 minutes, and finishing using our other methods gave

$$\#\mathbf{J} = 339659790214687297284652908385855015466 \ . \tag{23}$$

8 Perspectives for Further Research

The present paper reports on practical algorithms for counting points on hyperelliptic curves over large finite fields and on implementations for genus 2. Although it is now possible to deal with almost cryptographic-size Jacobians, there is still a substantial amount of work to be done. Some improvements or generalizations seem to be accessible in the near future, whereas others are still quite vague. Among them we would like to mention:

- Extension of the algorithm to even characteristic. This is only a matter of translating the formulae, in order to deal with an equation of the form $y^2 + h(x)\, y = f(x)$. The Cartier-Manin part and the lifting of the 2–torsion should merge, giving an efficient way to compute the result modulo 2^k. For the Schoof-like part, the formulae of Cantor's division polynomials have to be adapted, which does not appear to be too difficult.
- Extension of the Schoof-like algorithm to genus $g > 2$. The main difficulty is that it does not appear possible to avoid manipulation of ideals.
- More use could certainly be made of the Jacobian of the twist curve.
- We believe that it may be possible to lift the curve to a local field with residue field $\mathbb{F}_q$ and use Cartier-Manin to compute $\chi(t)$ modulo small powers of the characteristic. We do not yet know how to compute the lift, however.
- A major improvement would be to elaborate a genus 2 version of the Elkies-Atkin approach for elliptic curves, which would lead to computations with polynomials of lower degree. We conjecture that it is possible to work with degrees reduced from $O(l^4)$ to $O(l^3)$. The first task is to construct modular equations for Siegel modular forms, instead of classical ones. This requires a description of isogenies for each small prime degree, which can be given by lists of cosets under left actions of the symplectic group $Sp_4(\mathbb{Z})$ instead of the classical modular group $SL_2(\mathbb{Z})$. Starting points for studying the relevant forms and groups include [Fre83] and [Kli90]. This will be explained in more detail elsewhere [Har].

All the above is the subject of active research.

References

[AH92] L. M. Adleman and M.-D. A. Huang. *Primality testing and Abelian varieties over finite fields*, vol. 1512 of *Lecture Notes in Math.* Springer–Verlag, 1992.

[BC97] W. Bosma and J. Cannon. *Handbook of Magma functions*, 1997. Sydney, `http://www.maths.usyd.edu.au:8000/u/magma/`.

[Can87] D. G. Cantor. Computing in the Jacobian of an hyperelliptic curve. *Math. Comp.*, 48(177):95–101, 1987.

[Can94] D. G. Cantor. On the analogue of the division polynomials for hyperelliptic curves. *J. Reine Angew. Math.*, 447:91–145, 1994.

[Car57] P. Cartier. Une nouvelle opération sur les formes différentielles. *C. R. Acad. Sci. Paris Sér. I Math.*, 244:426–428, 1957.

[Cou96] J.-M. Couveignes. Computing l-isogenies using the p-torsion. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 59–65. Springer Verlag, 1996. Second International Symposium, ANTS-II, Talence, France, May 1996, Proceedings.

[Elk98] N. Elkies. Elliptic and modular curves over finite fields and related computational issues. In D.A. Buell and J.T. Teitelbaum, editors, *Computational Perspectives on Number Theory*, pages 21–76. AMS/International Press, 1998. Proceedings of a Conference in Honor of A.O.L. Atkin.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62(206):865–874, April 1994.

[Fre83] E. Freitag. *Siegelsche Modulfunktionen*, volume 254 of *Grundlehren der mathematischen Wissenschaften*. Springer–Verlag, 1983.

[Har] R. Harley. On modular equations in genus 2. In preparation.

[HI98] M.-D. Huang and D. Ierardi. Counting points on curves over finite fields. *J. Symbolic Comput.*, 25:1–21, 1998.

[IR82] K. F. Ireland and M. Rosen. *A classical introduction to modern number theory*, volume 84 of *Graduate texts in Mathematics*. Springer–Verlag, 1982.

[Kam91] W. Kampkötter. *Explizite Gleichungen für Jacobische Varietäten hyperelliptischer Kurven*. PhD thesis, Univ. Gesamthochschule Essen, August 1991.

[Kli90] H. Klingen. *Introductory lectures on Siegel modular forms*, vol. 20 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1990.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *J. of Cryptology*, 1:139–150, 1989.

[Lec99] G. Lecerf. *Kronecker, Polynomial Equation System Solver, Reference manual*, 1999. `http://www.gage.polytechnique.fr/~lecerf/software/kronecker`.

[Ler97] R. Lercier. *Algorithmique des courbes elliptiques dans les corps finis*. Thèse, École polytechnique, June 1997.

[Man65] J. I. Manin. The Hasse-Witt matrix of an algebraic curve. *Trans. Amer. Math. Soc.*, 45:245–264, 1965.

[Mor95] F. Morain. Calcul du nombre de points sur une courbe elliptique dans un corps fini : aspects algorithmiques. *J. Théor. Nombres Bordeaux*, 7:255–282, 1995.

[Mum84] D. Mumford. *Tata lectures on theta II*, volume 43 of *Progr. Math.* Birkhauser, 1984.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $\mathrm{GF}(p)$ and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT–24:106–110, 1978.

[Pil90] J. Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comp.*, 55(192):745–763, October 1990.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation mod p. *Math. Comp.*, 32(143):918–924, July 1978.

[Rüc99] H. G. Rück. On the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 68(226):805–806, 1999.

[Sch] E. Schost. Computing parametric geometric resolutions. Submitted to ISSAC'2000.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44:483–494, 1985.

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.

[ST99] A. Stein and E. Teske. Catching kangaroos in function fields. Preprint, March 1999.

[Tat66] J. Tate. Endomorphisms of Abelian varieties over finite fields. *Invent. Math.*, 2:134–144, 1966.

[Ver99] F. Vercauteren. `#EC(GF(2^1999))`. E-mail message to the NMBRTHRY list, Oct 1999.

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

[Yui78] N. Yui. On the jacobian varietes of hyperelliptic curves over fields of characteristic $p > 2$. *J. Algebra*, 52:378–410, 1978.