Counting Points in Medium Characteristic Using Kedlaya's Algorithm

Pierrick Gaudry and Nicolas Gürel

CONTENTS

2000 AMS Subject Classification: Primary 11G20, 11Y16, 14Q05, 14H45; Secondary 11T71, 14H40

Keywords: Cryptography, hyperelliptic curves, Kedlaya's algorithm, point counting

Recently, many new results have been found concerning algorithms for counting points on curves over finite fields of characteristic p, mostly due to the use of p-adic liftings. The complexity of these new methods is exponential in $\log p$, therefore they behave well when p is small, the ideal case being $p = 2$. When applicable, these new methods are usually faster than those based on SEA algorithms, and are more easily extended to nonelliptic curves. We investigate more precisely this dependence on the characteristic, and in particular, we show that after a few modifications using fast algorithms for radix-conversion, Kedlaya's algorithm works in time almost linear in p. As a consequence, this algorithm can also be applied to medium values of p. We give an example of a cryptographic size genus 3 hyperelliptic curve over a finite field of characteristic 251.

1. INTRODUCTION

Computing the zeta function of curves over finite fields is an important task for cryptography. Indeed, for the design of a cryptosystem whose security is based on the discrete logarithm problem in the Jacobian of a curve, it is required that its group order is a prime (or a small cofactor times a prime).

Two classes of algorithms can be used: those based on Schoof's idea of computing the result modulo several small primes using torsion elements [Schoof 95, Pila 90, Adleman and Huang 01, Couveignes 96], and those using a lift of the curve to a p-adic ring by Satoh [Satoh 00], Kedlaya [Kedlaya 01], and Lauder and Wan [Lauder and Wan 01]. The first family of algorithms works for any finite field whereas the second family requires the characteristic to be small. When p-adic algorithms are applicable, there are usually much faster than those based on Schoof's idea; furthermore, they are more easily extended to high genus curves. For instance, the Kedlaya and Lauder-Wan algorithms have complexities which are polynomial in the genus, whereas Pila's algorithm is exponential in the genus.

© A K Peters, Ltd.
1058-6458/2003$0.50 per page
Experimental Mathematics **12**:4, page 395

The p-adic algorithms can again be divided into two classes: those based on the computation of the canonical lift of the curve, and those that use an arbitrary lift. Although the subject is recent, the literature is already quite extensive. Improvements to Satoh's original algorithm for elliptic curves have been made in various places [Fouquet et al. 00, Fouquet et al. 01, Skjernaa 03, Vercauteren et al. 01, Satoh et al. 03, Satoh 02, Gaudry 02, Harley 02]. This method was extended to genus 2 curves in characteristic 2 by Mestre [Mestre 00]. Variants of Kedlaya's algorithm have been designed [Gaudry and Gürel 01], in particular to extend it to hyperelliptic curves in characteristic 2 [Denef and Vercautern 02, Vercauteren 02]. Also, Lauder and Wan improved their algorithm for various classes of varieties [Lauder and Wan 01, Lauder 03].

The purpose of this paper is to analyse the behaviour of the p-adic algorithms when the characteristic is not so small. We concentrate on the case of nonelliptic curves. Indeed, for elliptic curves, Schoof's algorithm allows one to solve the point counting problem efficiently, whereas already in genus 2, Schoof's like algorithm is much slower and more complicated [Gaudry and Harley 00, Gaudry and Schost 02]. Hence, it makes sense to try to push the p-adic algorithms as far as possible in term of the characteristic.

There are also applications: Some cryptosystem designers like to have curves defined over a finite field with a medium characteristic, in order, for instance, to have integers modulo the characteristic that fit into one or one half of a machine word [Bailey and Paar 98].

Let us now consider the theoretical complexity of different p-adic algorithms in terms of the characteristic p. This complexity is always exponential in $\log p$, but varies quite a lot. In algorithms based on the canonical lift, the p-torsion subgroup plays an important role and it seems to be impossible to avoid working modulo an ideal defining it. As a matter of fact, it has degree p^{2g}, and the complexity is at least $\Omega(p^{2g})$. So even for elliptic curves, the complexity appears to be $\Omega(p^2)$. Note also that computing the canonical lift involves an explicit representation of p^g isogenies, namely modular equations.

On the other hand, Kedlaya's algorithm has a dependence on p which is not as clear. Our main result is that after a few adaptations using fast algorithms for radix conversions, the complexity is in fact $\widetilde{O}(p)$. Recall that the notation $\widetilde{O}(N)$ means $O(N(\log N)^k)$ for some constant integer k. We then validate this complexity with some practical experiments and give a cryptographic size genus 3 curve.

2. AN OVERVIEW OF KEDLAYA'S ALGORITHM

Let $\mathcal{C}$ be a hyperelliptic curve of genus g defined by its affine equation $y^2 = \overline{f}(x)$ over a finite field $\mathbb{F}_q$ where $q = p^n$ and $\overline{f}$ is a polynomial of degree $2g+1$; hence, we assume that $\mathcal{C}$ has a rational Weierstrass point. In [Kedlaya 01], Kedlaya gives an explicit construction of the de Rham cohomology space H^1 of the coordinate ring of a curve associated to $\mathcal{C}$ and computes the Frobenius action on a particular basis. This computation is done in the set of overconvergent series $A^\dagger$. Kedlaya shows how to reduce those series onto the initial basis; he also estimates the precision needed. We limit ourselves to a short description of the algorithm.

Kedlaya's algorithm illustrates the principle of the Monsky-Washnitzer cohomology in the case of hyperelliptic curves. For a more general introduction to p-adic cohomology, see [van der Put 86] or [Koblitz 77].

2.1 Definitions and Notation

2.1.1 The ring $\mathbb{Z}_q$. Let $\overline{P}$ be a monic polynomial which defines $\mathbb{F}_q$ as an algebraic extension of $\mathbb{F}_p$ and let P be any monic lift of degree n over $\mathbb{Z}_p$. The first step is to build the p-adic ring needed for the computation. The quotient $\mathbb{Z}_q = \mathbb{Z}_p[t]/(P(t))$ defines, up to isomorphism, the ring of integers of the unramified field extension K of degree n of $\mathbb{Q}_p$. In practice, we will be working in $\mathbb{Z}_q/p^\nu$, where ν is the p-adic precision. The output of the algorithm is the numerator of the zeta function associated to $\mathcal{C}$ modulo p^ν. Thus, we can recover the numerator of the zeta function as soon as ν is larger than a quantity which depends on Weil's bounds. One can show that $\nu = \widetilde{O}(ng)$. In the following, we extend the p-th power Frobenius of $\mathbb{F}_q$ to many different objects and for simplicity we will denote it by σ regardless of the object on which it operates. In particular, σ extends to $\mathbb{Z}_q$.

2.1.2 The algebra $A^\dagger_+$. The lift of $\overline{f}$ to f in $\mathbb{Z}_q[x]$ defines a lift of the curve in characteristic zero. We recall that $A^\dagger$ is constructed as the quotient $\mathbb{Z}_q$-algebra of power series in x, y and $1/y$ modulo $y^2 - f$, and such that the valuation of the coefficients grows at least linearly with the degree in x, y and $1/y$. In the algorithm, it suffices to consider $A^\dagger_+$, the subalgebra of $A^\dagger$ of power series in x and $\tau = 1/y^2$ (i.e., $A^\dagger_+$ is the subalgebra of $A^\dagger$ which is stable under the hyperelliptic involution).

For the computation, an important consequence of the fast convergence is that the precision μ in τ for which all the coefficients are zero modulo p^ν is linear in $p\nu$ (i.e., in $\widetilde{O}(png)$, see Lemma 2 in [Kedlaya 01] for more details).

To represent an element of $A^\dagger_+$, we adopt a normalised form:

Definition 2.1. (Normal form.) We say that an element $S(x,\tau) = S_0(x) + \sum_{i>0} S_i(x)\tau^i \in A^\dagger_+$ is represented in normal form if $\deg(S_i) \leq 2g$ for $i > 0$.

Notice that, using the equation $y^2 = f(x)$, we can write any element of $A^\dagger_+$ in normal form, and that $\deg(S_0)$ can be arbitrarily large. We will see below that in Kedlaya's algorithm, all the elements of $A^\dagger_+$ are such that $\deg(S_0) \leq 2gp - (p+1)/2$.

Remark 2.2. The set of power series $\sum_{i\geq 0} S_i(x)\tau^i \in A^\dagger_+$, in normal form, with $\deg(S_0) = 0$ is stable under multiplication. Indeed, if $P = \sum_{i\geq 0} P_i(x)\tau^i$ is the product of two such series before normalisation, then $\deg(P_0) = 0$, $\deg(P_1) \leq 2g$, and $\deg(P_i) \leq 4g$ for $i > 1$. Using the equation of the curve (i.e., computing the remainder of P_i by $f = 1/\tau$), if $i > 1$, the normalisation of $P_i\tau^i$ contributes a polynomial of degree at most $2g$ times τ^{i-1}, and normalising P_1 contributes a constant times τ^0.

The extension of the Frobenius action to $A^\dagger$ is chosen such that $x^\sigma = x^p$ and $(y^\sigma)^2 = f(x)^\sigma$.

2.1.3 Basis of differential forms. We consider the quotient space H^1_-, of differentials of the form $S(x,\tau)dx/y$, where $S \in A^\dagger_+$, modulo the differentials which are exact. Lemma 2 in [Kedlaya 01] implies in particular that $\mathcal{B} = \{x^i dx/y, i \in [0, 2g-1]\}$ is a basis of H^1_- over the p-adic number field K, and that this vector space is stable under the action induced by σ.

This action of Frobenius endomorphism on differential forms is given by $(dx)^\sigma = d(x^\sigma) = px^{p-1}dx$, hence, we can compute the action of σ on each element of the basis $\mathcal{B}$ of H^1_-. We have

$$(x^i dx/y)^\sigma = (x^i)^\sigma (dx)^\sigma / y^\sigma = px^{ip+p-1}dx/y^\sigma,$$

and that expression is then rewritten as a linear combination of elements of $\mathcal{B}$ by adding appropriate elements dh where $h \in A^\dagger$. Indeed, those exact forms are zero in H^1_-. This is explained in detail in Algorithm 1.

2.2 Kedlaya's Algorithm

In the algorithm, we compute the action of σ on each element of $\mathcal{B}$, which gives the matrix of σ in this basis. The characteristic polynomial of the norm of this matrix gives us the numerator of the zeta function, modulo the p-adic precision, due to Lefschetz fixed point formulae. We describe the whole computation in Algorithm 2.

Input: $\omega = \sum_{0\leq m\leq \mu} Q_m(x)\tau^m dx/y$, as a normalized element of $A^\dagger_+$ times dx/y.
Output: The coefficients of ω in $\mathcal{B}$.
Step i. [First reduction: write $\sum_{0\leq m\leq \mu} Q_m(x)\tau^m dx/y$ in the form $Q(x)dx/y$]
for $k := \mu$ to 1 by -1 **do**
  Compute $U_k(x)$ and $V_k(x)$ such that $Q_k(x) = U_k(x)f(x) + V_k(x)f'(x)$;
  Replace the coefficient $Q_k(x)\,\tau^k \frac{dx}{y}$ in ω by

$$\left(U_k(x) + \frac{2}{2k-1}V_k'(x)\right)\tau^{k-1}\frac{dx}{y};$$

endfor
Step ii. [Second reduction: reduce the degree of $Q(x)$]
Initialisation: $\delta \leftarrow \deg(Q)$;
while $\delta > 2g$ **do**
  Compute $\widetilde{Q}$ such that $d(2x^{\delta-2g}y) = \widetilde{Q}(x)dx/y$:

$$\widetilde{Q} = \left(2(\delta-2g)x^{\delta-2g-1}f(x) + x^{\delta-2g}f'(x)\right);$$

  (Note that $\widetilde{Q}$ has the same degree as Q.)
  Normalize $\widetilde{Q}$ so that it has the same leading coefficient as Q;
  $Q \leftarrow Q - \widetilde{Q}$;
  $\delta \leftarrow \deg(Q)$;
endw

ALGORITHM 1. Cohomological reductions.

3. COMPLEXITY IN p

Following the computation from [Kedlaya 01] of the time and space dependence on n and g, but taking into account the contribution of p, we give a proof that the complexity is almost linear in p and that the introduction of the radix-conversion has not disturbed the dependence on the other parameters. We recall that we use the Soft-Oh notation, thus any contribution in $\log p$, $\log n$, or $\log g$ in the complexity is not to be taken into account.

3.1 Bit-Size of the Different Objects

Notice that a multiplication between two objects of bit-size N is assumed to take time $\widetilde{O}(N)$, thanks to Schönhage's fast multiplication algorithm. To analyse the complexity, it is important to describe the bit-size of the different objects we manipulate in the algorithm.

Using the notations of Section 2.1, $\mathbb{Z}_q$ denotes the quotient $\mathbb{Z}_p[t]/(P(t))$ and elements in $\mathbb{Z}_p$ are truncated to

Input: A curve $\mathcal{C}$ defined by $y^2 = \overline{f}$ over the finite field $\mathbb{F}_q$.
Output: The numerator of the zeta function of $\mathcal{C}$.
Step 0. Compute the p-adic and τ-adic precisions, the element t^σ, the lift of f and f^σ, U and V such that $Uf + Vf' = 1$;
Step 1. [Computation of $1/y^\sigma$] From equations $(y^2)^\sigma = (f(x))^\sigma$ and $y^\sigma \equiv y^p \bmod p$, we see that $1/y^\sigma = \tau^{(p-1)/2} S/y$ with $S := (1 + (f(x)^\sigma - f(x)^p)\,\tau^p)^{-1/2}$. The power series S can be computed efficiently by a Newton iteration in $A_+^\dagger$;
Step 2. [Computation of the Frobenius action on $\mathcal{B}$]
for each *differential* ω_i *in* $\mathcal{B}$ **do**
  Step 2.1. [Compute $\omega_i^\sigma = p\tau^{(p-1)/2} x^{ip+p-1} S dx/y$] This is essentially the multiplication of $p\tau^{(p-1)/2} x^{ip+p-1}$ by S, as normalised elements of $A_+^\dagger$. We obtain ω_i^σ written in the form $\sum_{0\le k\le \mu} Q_k \tau^k dx/y$;
  Step 2.2. [Reduce ω_i^σ as a linear combination of elements of $\mathcal{B}$] We apply Algorithm 1 to ω_i^σ;
endfch
Step 3. [Compute the characteristic polynomial $\chi(t)$ of the norm of the Frobenius] The action of the p-th power Frobenius endomorphism on the differential forms is gathered in a matrix M. The q-th power action is then obtained by computing $\mathrm{Norm}(M) = M\, M^\sigma \cdots M^{\sigma^{n-1}}$. The characteristic polynomial χ of the Frobenius is computed as the characteristic polynomial of this matrix;
return $t^{2g}\chi(1/t)$.

ALGORITHM 2. The main algorithm.

precision p^ν where $\nu = \widetilde{O}(ng)$. This implies that an element of $\mathbb{Z}_q$ is represented as a polynomial of degree n with coefficients in $\mathbb{Z}/(p^\nu)$, therefore the bit-size of an element of $\mathbb{Z}_q$ is $\widetilde{O}(n^2 g)$. An element of the set $A_+^\dagger$, represented in normal form, is a power series in τ, truncated modulo τ^μ, over the polynomials of degree $2g+1$ over $\mathbb{Z}_q$. We recall that the precision in τ, namely μ, is linear in p, and that the coefficient $S_0(x)$ is of degree at most $O(gp)$. More precisely, we have $\mu = \widetilde{O}(png)$, therefore the bit-size of an element of $A_+^\dagger$ is $\widetilde{O}(n\nu \cdot \mu g) = \widetilde{O}(pn^3 g^3)$.

ν (p-adic precision)	$\widetilde{O}(ng)$	an element of $\mathbb{Z}_q$	$\widetilde{O}(n^2 g)$
μ (τ-adic precision)	$\widetilde{O}(png)$	an element of $A_+^\dagger$	$\widetilde{O}(pn^3 g^3)$

TABLE 1. Bit-size of main elements.

3.2 Frobenius Substitution

To compute the Frobenius action on $\mathbb{Z}_q$, we have to estimate t^σ modulo the p-adic precision. The element t^σ is a zero of P and is congruent to $t^p \bmod p$, therefore we use a Newton iteration:

$$\begin{cases} x_0 = t^p \mod p, \\ x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i)}. \end{cases}$$

It costs $\log p$ operations in $\mathbb{F}_q$ for the initialisation. At the step i of the iteration, we have computed $t^\sigma \bmod p^{2^i}$. As usual, for this type of Newton computation, the overall cost is a constant times the cost of the last step. In this case, it costs $O(n)$ multiplications in $\mathbb{Z}_q$ at maximal precision. Hence, the computation of t^σ costs $O(n)$ operations in $\mathbb{Z}_q$, i.e., the cost of this computation is $\widetilde{O}(n^3 g)$. Let

$$z = z_{n-1}t^{n-1} + z_{n-2}t^{n-2} + \cdots + z_1 t + z_0$$

be an element of $\mathbb{Z}_q$. For $z^\sigma = \sum_{i=0}^{n-1} z_i (t^\sigma)^i$, Horner's method yields a way of computing it at a cost of n operations in $\mathbb{Z}_q$. So this computation is in time $\widetilde{O}(n^3 g)$.

More generally, computing $\sigma^k(z)$ for any k can be done at the same cost, by lifting t^{σ^k} and plugging it into the expansion of z. Consequently, using the 2-adic expansion of k, we can compute the norm of an element in time $\widetilde{O}(n^3 g)$.

3.3 Normal Form

At several places in the algorithm, the elements of $A_+^\dagger$ do not come naturally in their normal form. First, this occurs in Step 1, where we have to compute the series $(f(x)^\sigma - f(x)^p)\tau^p$. Also in Step 2.1, we have to put the series $p\tau^{(p-1)/2} x^{ip+p-1}$ in normal form before multiplying it by S. More details about these steps are given below.

For small p, this normalisation step takes a negligible time and can be done in a naive way. However, for large p, using naïve algorithms makes it the dominant step in the whole algorithm. It is then required to use asymptotically fast algorithms, namely the recursive radix-conversion algorithm.

More precisely, let $Q(x)\tau^m \in A_+^\dagger$ with $\deg(Q) > 2g$. Let $k = \left\lceil \frac{\deg(Q)}{2g+1} \right\rceil$. The normal form essentially consists of the coefficients of the (f)-adic expansion of Q, that is the polynomials $(a_0, \ldots, a_k)$ such that

$$Q\tau^m = (a_0 + a_1 f + \cdots + a_{m-1} f^{m-1} + a_m f^m + \cdots + a_k f^k) f^{-m},$$

where all of the a_i have degree at most $2g$. Then, if we put $Q_0 = a_m + a_{m+1}f + \cdots + a_k f^{k-m}$ and $Q_i = a_{m-i}$ for $i > 0$, we get the normal form $Q(x)\tau^m = Q_0(x) + \sum_{i>0} Q_i(x)\tau^i$.

To compute the a_i, we compute Q_1 and Q_2 such that $Q = Q_1 f^{\lfloor k/2 \rfloor} + Q_2$ and we call the radix-conversion algorithm recursively on Q_1 and Q_2. It takes $\widetilde{O}(kg)$ operations in $\mathbb{Z}_q$ to compute those coefficients (see [von zur Gathen and Gerhard 99, Theorem 9.15] for more details).

Conversely, building Q from its (f)-adic expansion can also be done in the same time, using a similar strategy.

3.4 Cost of Cohomological Reductions

First, the polynomials U and V such that $Uf + Vf' = 1$ are computed only once. In one step of the first reduction applied to a differential $\omega_k = Q_k \tau^k dx/y$, where $\deg(Q_k) \leq 2g$, we compute $\widetilde{\omega}_k = \widetilde{Q}_{k-1}\tau^{k-1}dx/y$ in the same class of cohomology as ω_k, for $k > 0$. We refer to Algorithm 2.1.2 for the formulae that are used; they involve the multiplication of Q_k by U and V to obtain U_k and V_k, thus it costs a constant number of multiplications of polynomials of degree $O(g)$ over $\mathbb{Z}_q$, which can be performed in time $\widetilde{O}(n^2g^2)$.

One step of the second reduction decreases by 1 the degree $\delta > 2g$ of the polynomial Q in the differential $Q dx/y$. Again, in Algorithm 2.1.2, we see that this involves one inversion and $O(g)$ multiplications of scalars (the polynomial multiplications are just shifts), thus the time complexity is $\widetilde{O}(n^2g^2)$.

3.5 Overall Complexity

In Step 1, let $U = 1 + (f^\sigma - f^p)\tau^p$. We compute the normal form to write

$$U = U_0 + U_1\tau + \cdots + U_p\tau^p,$$

where $\deg(U_i) \leq 2g$. It costs the computation of the polynomial $f^\sigma - f^p$ of degree $O(pg)$ and its radix conversion as described in Section 3.3. The time complexity is $\widetilde{O}(n^3g^2 + pn^2g^2)$.

Then we perform a Newton iteration to take the inverse of the square root of U. There are $O(\log \mu)$ iterations to be done and after each step, we renormalise the series to prevent the growth of the coefficients. As usual for a Newton iteration, the overall cost is a constant times the cost of the last iteration. This last iteration involves a constant number of multiplications of two elements of $A_+^\dagger$ truncated modulo τ^μ, therefore the cost of the multiplication is $\widetilde{O}(pn^3g^3)$ (see Table 1).

The result of this operation is a truncated power series $S = S_0 + S_1\tau + \cdots + S_\mu\tau^\mu$ with $\deg(S_i) \leq 4g + 2$. By the statement after Definition 2.1 and noticing that $\deg(U_0) = 0$, we have $\deg(S_0) = 0$. Thus, the cost of a normalisation in this case is $\widetilde{O}(pn^3g^3)$.

The global cost of Step 1 is then $\widetilde{O}(pn^3g^3)$.

In Step 2, we have to compute $B = p\tau^{(p-1)/2}x^{pi+p-1}S$ for $i \in [0, 2g-1]$. We concentrate on the worst case, namely $i = 2g-1$, and we write $A = p\tau^{(p-1)/2}x^{2gp-1}$ and $B = AS$. After normalisation, we can write

$$A = A_0 + A_1\tau + \cdots + A_k\tau^k,$$

where $k := (p-1)/2$. The degree of $A_0(x)$ in x is $k_0 = (2g-1)(p+1)/2$ and the degree of S in τ is μ, both linear in p. Naïvely, there are p multiplications of a polynomial of degree p by a polynomial of degree $2g+1$ to do. Thus, the complexity for computing B appears to be quadratic in p.

A workaround to this problem is to use the "complete" (f)-adic expansion of A and obtain

$$A = \sum_{i=-k_1}^{k} A_i\tau^i = \tau^{-k_1}\left(\sum_{i=0}^{k+k_1} A_{i-k_1}\tau^i\right),$$

with k_1 linear in p and $\deg(A_i) \leq 2g$. The time complexity of this operation as shown in Section 3.3 is $\widetilde{O}(pn^2g^2)$. Now $\tau^{k_1}A$ and S are both power series truncated at precision μ in τ with polynomials coefficient of degree $2g$ in x. The cost of the multiplication of $\tau^{k_1}A$ by S is the same as the last step of the Newton iteration, thus still $\widetilde{O}(pn^3g^3)$. Shifting the result by τ^{-k_1}, we obtain an expression for B that involves negative powers in τ. We then convert it to the normalised form as follows: Write $B = B_1\tau^{-k_1} + B_2$, where $B_1\tau^{-k_1}$ contains the negative powers of τ. Converting $B_1\tau^{-k_1}$ back into a polynomial in x is the reciprocal of the transformation explained in Section 3.3, and can be done in time $\widetilde{O}(pn^2g^2)$. Therefore, we obtain B written as $B = \widetilde{B}_1(x) + B_2$, where $\widetilde{B}_1(x)$ is a polynomial in x of degree $O(gp)$ and B_2 is a power series with polynomials of degree at most $2g$.

In the cohomological reductions, we apply μ times the first reduction to reduce B_2 at a cost of $\widetilde{O}(pn^3g^3)$, and the contribution of B_2 is added to $\widetilde{B}_1$. The second reduction is applied $O(gp)$ times to $\widetilde{B}_1$ to get a polynomial of degree less than $2g$. The cost is then $\widetilde{O}(pn^2g^3)$.

The cost of treating one differential form is $\widetilde{O}(pn^3g^3)$. There are $O(g)$ of them, therefore the overall cost of Step 2 is $\widetilde{O}(pn^3g^4)$.

In Step 3, the dominant part is the computation of the norm of a $2g \times 2g$ matrix in K. As described in Section 3.2, the cost of the computation of M^σ is $\widetilde{O}(n^3g^3)$,

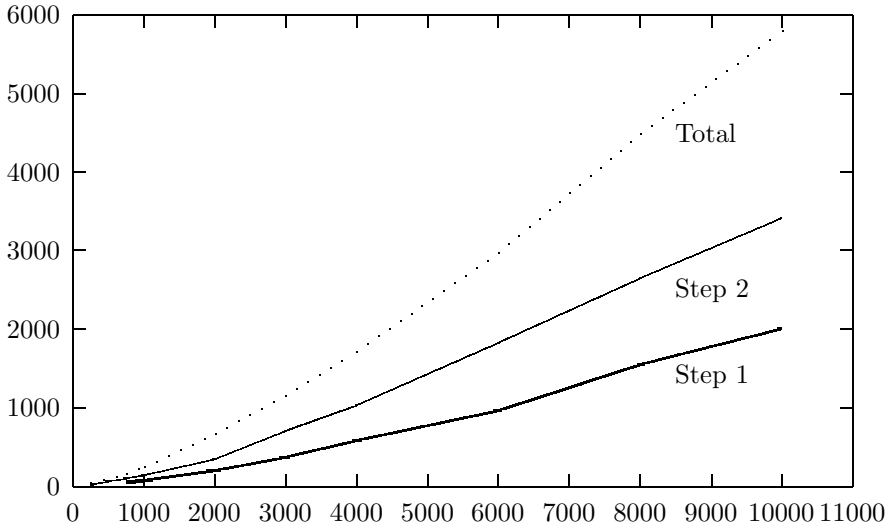

FIGURE 1. Runtimes in seconds as functions of the characteristic p.

and the cost of a matrix multiplication is $\widetilde{O}(n^2g^4)$. Therefore, the norm computation costs $\widetilde{O}(n^2g^4 + n^3g^3)$. Classical fast algorithms for computing the characteristic polynomial of matrices lead to a complexity of $\widetilde{O}(n^2g^4)$. Hence, the cost of Step 3 is $\widetilde{O}(n^2g^4 + n^3g^3)$.

Putting everything together, we obtain the main theorem of this section.

Theorem 3.1. *The overall time complexity of Kedlaya's algorithm is* $\widetilde{O}(pn^3g^4)$. *The space complexity of the algorithm is* $\widetilde{O}(pn^3g^3)$.

4. PRACTICAL EXPERIMENTS

To illustrate our analysis of the complexity, we have written an implementation of Kedlaya's algorithm adapted to the medium characteristic case. In particular, we have used the recursive radix-conversion algorithm of Section 3.3. We used the NTL library written by Shoup [Shoup 02], compiled with the GMP multiprecision package [Granlund 02]. Elements of the p-adic ring $\mathbb{Z}_q$ are represented as elements of an extension of the finite ring $\mathbb{Z}/p^\nu\mathbb{Z}$, and we wrote specific code to handle the few divisions by p that occur in the algorithm. NTL provides the arithmetic of polynomials over this extension ring. Above this, we added a simple arithmetic layer for the power series required in Kedlaya's algorithm. For multiplication of those power series, however, we convert everything to integers (Kronecker substitution) and use GMP's implementation of Schönhage's algorithm.

4.1 Running Times When Only p Varies

For each prime p, we fix a root t of the polynomial used to define $\mathbb{F}_{p^3}$ over $\mathbb{F}_p$, and we consider the genus 3 hyperelliptic curves defined over $\mathbb{F}_{p^3}$ by

$$C_t \ : \ y^2 = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + t.$$

We ran our implementation on these curves for different values of p. The required p-adic precision is 7, and the power series are truncated modulo τ^μ, where μ is about $6p$. The running times are given on a AMD-Athlon MP 2200+. We also measured the maximal amount of memory used by the program. The data are given in Table 2 and graphically represented in Figure 1.

The complexity is clearly subquadratic in p. We also insist on the fact that FFT-based algorithms are only efficient for very large data, and their essentially linear runtime is often hidden by logarithmic factors. In our analysis, we did not take into account those logarithmic factors. In fact, one factor $\log p$ is hidden in the size of the elements of $\mathbb{Z}_p$, another one is due to the cost of radix-conversion which is $O(\log p)$ multiplications, and a third $O(\log p)$ contribution comes from the complexity of Schönhage's integer multiplication algorithm. Putting all of this together, we see that there is actually a factor $O(\log^3 p)$ hidden in the complexity. This factor explains the resulting growth of the runtime.

4.2 A Cryptographic Size Curve Over $\mathbb{F}_{251^7}$

Let C be the hyperelliptic curve of genus 3 defined over $\mathbb{F}_{251^7}$ by the equation $y^2 = f(x)$, where

p	Minpol(t)	Step 1	Step 2	Total time	Memory
251	$t^3+20t^2+95t+116$	11 s	28 s	42 s	25 MB
503	$t^3+20t^2+95t+372$	25 s	67 s	100 s	48 MB
751	$t^3+20t^2+95t+445$	56 s	107 s	164 s	84 MB
1009	$t^3+79t^2+764t+463$	82 s	147 s	244 s	114 MB
2003	$t^3+746t^2+66t+1844$	200 s	347 s	664 s	225 MB
3001	$t^3+152t^2+1723t+2076$	371 s	712 s	1155 s	377 MB
4001	$t^3+1723t^2+2493t+3307$	585 s	1031 s	1720 s	503 MB
6007	$t^3+152t^2+3307t+3469$	971 s	1832 s	2975 s	750 MB
8009	$t^3+3469t^2+6172t+4424$	1551 s	2656 s	4482 s	1.0 GB
10007	$t^3+152t^2+3307t+3469$	2010 s	3423 s	5798 s	1.4 GB

TABLE 2. Time and space data for various p.

$$\begin{aligned} f(x) = x^7 &+ t^{17808079175804175}x^6 + t^{54575364231919474}x^5 \\ &+ t^{237357237234904}x^4 + t^{3218736229782234}x^3 \\ &+ t^{41232191549139817}x^2 + t^{41258843266959358}x \\ &+ t^{43871791627662980}, \end{aligned}$$

and t with minimal polynomial $t^7+197t^5+245t^4+t^3+148t^2+119t+225$ over $\mathbb{F}_{251}$. The characteristic polynomial of the Frobenius endomorphism is then

$$\chi(T) = T^6 - s_1T^5 + s_2T^4 - s_3T^3 + qs_2T^2 - q^2s_1T + q^3,$$

where

$$\begin{aligned} s_1 &= -77096895, \\ s_2 &= -482223667309721, \\ s_3 &= -1329575558557091248791717, \end{aligned}$$

which gives a prime cardinality of

$$\begin{aligned} N =& 247256747242618310605558095585341310826595788242067. \end{aligned}$$

Notice that, in this case, it takes less than seven minutes of computation for one hyperelliptic curve and uses 190 MB of memory.

5. CONCLUDING REMARKS

It is straightforward to check that our analysis is still valid for the adaptation of Kedlaya's algorithm to superelliptic curves described in [Gaudry and Gürel 01].

Instead of using the radix-conversion to write the polynomials as (f)-adic expansions, we could have used this basis for the whole computation. However, this would imply the finding of another appropriate basis for the differentials and rewriting the formula for the reductions accordingly. Furthermore, in practice, operations with polynomials represented in the classical basis are much faster and easier to handle.

It is known [Bostan et al. 03] that computing the Cartier-Manin matrix, hence also the zeta function modulo p of a hyperelliptic curve, can be done at a cost $\widetilde{O}(\sqrt{p})$ (not taking into account the dependence of the other parameters). The question is still open whether this complexity can be obtained on the whole zeta function after an adaptation of Kedlaya's algorithm. Another open question is how to reduce the space complexity. Indeed, this is now clearly the bottleneck of this method.

ACKNOWLEDGMENTS

We are grateful to Andreas Enge, Guillaume Hanrot, and François Morain for their close reading and helpful comments on the manuscript.

The computations were carried out on machines at the UMS Medicis at École Polytechnique.

REFERENCES

[Adleman and Huang 01] L. Adleman and M.-D. Huang. "Counting Points on Curves and Abelian Varieties over Finite Fields." *J. Symbolic Comput.* 32 (2001), 171–189.

[Bailey and Paar 98] D. Bailey and C. Paar. "Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms." In *Advances in Cryptology—CRYPTO '98*, Lecture Notes in Comput. Sci., 1462, pp. 472–485. Berlin: Springer-Verlag, 1998.

[Bostan et al. 03] A. Bostan, P. Gaudry, and É. Schost. "Linear Recurrence with Polynomial Coefficients and Computation of the Cartier-Manin Operator on Hyperelliptic Curves." To appear in *Proceedings of the Seventh International Conference on Finite Fields and Applications F_q7*, 2003.

[Couveignes 96] J.-M. Couveignes. "Computing l-Isogenies Using the p-Torsion." In *Algorithmic Number Theory*, Lecture Notes in Comput. Sci., 1122, pp. 59–65. Berlin: Springer-Verlag, 1996.

[Denef and Vercautern 02] J. Denef and F. Vercauteren. "An Extension of Kedlaya's Algorithm to Artin-Schreier Curves in Characteristic 2." In *Algorithmic Number Theory*, Lecture Notes in Comput. Sci., 2369, pp. 308–323. Berlin: Springer-Verlag, 2002.

[Fouquet et al. 00] M. Fouquet, P. Gaudry, and R. Harley. "An Extension of Satoh's Algorithm and its Implementation." *J. Ramanujan Math. Soc.* 15 (2000), 281–318.

[Fouquet et al. 01] M. Fouquet, P. Gaudry, and R. Harley. "Finding Secure Curves with the Satoh-FGH Algorithm and an Early-Abort Strategy." In *Advances in Cryptology—EUROCRYPT 2001*, Lecture Notes in Comput. Sci., 2045, pp. 14–29. Berlin: Springer-Verlag, 2001.

[Gaudry 02] P. Gaudry. "A Comparison and a Combination of SST and AGM Algorithms for Counting Points of Elliptic Curves in Characteristic 2." In *Advances in Cryptology—ASIACRYPT 2002*, Lecture Notes in Comput. Sci., 2501, pp. 311–327. Berlin: Springer-Verlag, 2002.

[Gaudry and Gürel 01] P. Gaudry and N. Gürel. "An Extension of Kedlaya's Point Counting Algorithm to Superelliptic Curves." In *Advances in Cryptology—ASIACRYPT 2001*, Lecture Notes in Comput. Sci., 2248, pp. 4780–494. Berlin: Springer-Verlag, 2001.

[Gaudry and Harley 00] P. Gaudry and R. Harley. "Counting Points on Hyperelliptic Curves over Finite Fields." In *Algorithmic Number Theory*, Lecture Notes in Comput. Sci., 1838, pp. 313–332. Berlin: Springer-Verlag, 2000.

[Gaudry and Schost 02] P. Gaudry and É. Schost. "Cardinality of a Genus 2 Hyperelliptic Curve over $GF(5 * 10^{24} + 41)$." Email to the NMBRTHRY list, September 2002.

[Granlund 02] T. Granlund. *The GNU Multiple Precision Arithmetic Library—4.1.* Swox AB, 2002. Distributed at http://swox.com/gmp/.

[Harley 02] R. Harley. "Elliptic Ccurve Point Counting: 32003 Bits." Email to the NMBRTHRY list, August 2002.

[Kedlaya 01] K. Kedlaya. "Counting Points on Hyperelliptic Curves Using Monsky-Washnitzer Cohomology." *J. Ramanujan Math. Soc.* 16 (2001), 323–338.

[Koblitz 77] N. Koblitz. *p-Adic Numbers, p-Adic Analysis and Zeta-Functions*, GTM, Vol. 58. New York-Heidelberg: Springer-Verlag, 1977.

[Lauder 03] A. Lauder. "Computing Zeta Functions of Kummer Curves via Multiplicative Characters." *Foundations of Computational Mathematics* 3:3 (2003), 273–295.

[Lauder and Wan 01] A. Lauder and D. Wan. "Counting Points on Varieties over Finite Fields of Small Characteristic." Preprint, 2001.

[Mestre 00] J.-F. Mestre. "Utilisation de l'AGM pour le calcul de $E(F_{2^n})$." Lettre adressée à Gaudry et Harley. Available in French at http://www.math.jussieu.fr/~mestre/, December 2000.

[Pila 90] J. Pila. "Frobenius Maps of Abelian Varieties and Finding Roots of Unity in Finite Fields." *Math. Comp.* 55:192 (1990), 745–763.

[Satoh 00] T. Satoh. "The Canonical Lift of an Ordinary Elliptic Curve over a Finite Field and Its Point Counting." *J. Ramanujan Math. Soc.* 15 (2000), 247–270.

[Satoh 02] T. Satoh. "On p-Adic Point Counting Algorithms for Elliptic Curves over Finite Fields." In *Algorithmic Number Theory*, Lecture Notes in Comput. Sci., 2369, pp. 43–66. Berlin: Springer-Verlag, 2002.

[Satoh et al. 03] T. Satoh, B. Skjernaa, and Y. Taguchi. "Fast Computation of Canonical Lifts of Elliptic Curves and Its Application to Point Counting." *Finite Fields and Their Applications* 9:1 (2003), 89–101.

[Schoof 95] R. Schoof. "Counting Points on Elliptic Curves over Finite Fields." *J. Théor. Nombres Bordeaux* 7 (1995), 219–254.

[Shoup 02] V. Shoup. *NTL: A Library for Doing Number Theory.* Available from World Wide Web (http://www.shoup.net/ntl/), 2002.

[Skjernaa 03] Berit Skjernaa. "Satoh's Algorithm in Characteristic 2." *Math. Comp.* 72 (2003), 477–487.

[van der Put 86] M. van der Put. "The Cohomology of Monsky and Washnitzer." *Mém. Soc. Math. France* 23 (1986), 33–60.

[Vercauteren 02] F. Vercauteren. "Computing Zeta Functions of Hyperelliptic Curves over Finite Fields of Characteristic 2." In *Advances in Cryptology—CRYPTO 2002*, Lecture Notes in Comput. Sci., 2442, pp. 369–384. Berlin: Springer-Verlag, 2002.

[Vercauteren et al. 01] F. Vercauteren, B. Preneel, and J. Vandewalle. "A Memory Efficient Version of Satoh's Algorithm." In *Advances in Cryptology—EUROCRYPT 2001*, Lecture Notes in Comput. Sci., 2045, pp. 1–13. Berlin: Springer-Verlag, 2001.

[von zur Gathen and Gerhard 99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge, UK: Cambridge University Press, 1999.

Pierrick Gaudry, Laboratoire d'Informatique (CNRS/FRE 2653), École Polytechnique, 91128 Palaiseau Cedex, France (gaudry@lix.polytechnique.fr)

Nicolas Gürel, Laboratoire d'Informatique (CNRS/FRE 2653), École Polytechnique, 91128 Palaiseau Cedex, France (gurel@lix.polytechnique.fr)

Received April 14, 2003, accepted in revised form August 12, 2003.