Counting Points on Elliptic Curves over Finite Fields of Small Characteristic in Quasi Quadratic Time

Reynald Lercier[1,2] and David Lubicz[1,3]

[1] CELAR
Route de Laillé
F-35570 Bruz
France
[2] Reynald.Lercier@m4x.org
[3] lubicz@celar.fr

Abstract. Let p be a small prime and $q = p^n$. Let E be an elliptic curve over $\mathbb{F}_q$. We propose an algorithm which computes without any preprocessing the j-invariant of the canonical lift of E with the cost of $O(\log n)$ times the cost needed to compute a power of the lift of the Frobenius. Let μ be a constant so that the product of two n-bit length integers can be carried out in $O(n^\mu)$ bit operations, this yields an algorithm to compute the number of points on elliptic curves which reaches, at the expense of a $O(n^{\frac{5}{2}})$ space complexity, a theoretical time complexity bound equal to $O(n^{\max(1.19,\mu)+\mu+\frac{1}{2}} \log n)$. When the field has got a Gaussian Normal Basis of small type, we obtain furthermore an algorithm with $O(\log(n)n^{2\mu})$ time and $O(n^2)$ space complexities. From a practical viewpoint, the corresponding algorithm is particularly well suited for implementations. We outline this by a 100002-bit computation.

Keywords: *elliptic curves, canonical lifts, AGM.*

1 Introduction

A prerequisite to the use in cryptography of elliptic curves defined over a finite field is to count efficiently the number of its rational points so as to avoid weak curves. Let p be a prime, $n \in \mathbb{N}^*$, $q = p^n$ and E be a given elliptic curve, the previous problem is equivalent to compute the trace of the q-th power Frobenius morphism Fr_q since $\sharp E(\mathbb{F}_q) = 1 + q - \mathrm{Tr}(\mathrm{Fr}_q)$.

The first polynomial time algorithm to compute the trace of Fr_q is due to Schoof [15]. It has been subsequently improved by Elkies and Atkin to achieve a heuristically estimated running time of $O((\log q)^{2\mu+2})$ bit operations where μ is a constant such that the product of two n-bit length integers can be carried out in $O(n^\mu)$ bit operations. Up to now, this is the only known algorithm which performs well for elliptic curves defined over fields of large characteristic.

Apart from these ℓ-adic algorithms, an other important family of algorithms uses p-adic techniques. The common idea behind these algorithms is to construct a lift of the Frobenius morphism over a characteristic zero field. The first algorithm that runs faster than the SEA algorithm is due to Satoh [13]. The idea is to compute a lift of E such that its ring of endomorphisms is the same as the ring of E. Such a lift is called the canonical lift of E. If we fix p (which shall be small in practice), the computational time of this algorithm when $n \to \infty$ is $O(n^{2\mu+1})$ and the memory consumption is $O(n^3)$. Shortly after, Vercauteren, Preneel, Vandewalle [18] reduced the memory complexity to $O(n^2)$. Independently, for the characteristic two case, an algorithm with the same asymptotic complexities was found by Mestre [10], based on the algebraic geometric mean (AGM).

In recent papers, Satoh, Skjernaa and Taguchi [12] improved further the complexity of Satoh's algorithm to reach a $O(n^{2\mu+\frac{1}{2}})$ time complexity and a $O(n^2)$ space complexity after a $O(n^3)$ bit operations precomputation. Finally, H. Kim et al. [7] showed that if the base field of the elliptic curve has a Gaussian Normal Basis (which is the case for numerous fields in a cryptographic context), the complexity in time of this algorithm can be reduced to $O(n^{2\mu+\frac{1}{1+\mu}})$.

Roughly, these algorithms can be split in two main phases. A first phase consists in computing the j-invariant of the canonical lift of the given curve. A second phase consists in a norm computation. The main contribution of this paper is a new way to perform the lift phase. In a more precise fashion, let $\mathbb{Z}_q$ be the valuation ring of the unramified extension of degree n of $\mathbb{Z}_p$ with p a prime. If we denote by $S_{m,n}$ the complexity of the Frobenius substitution with accuracy m in $\mathbb{Z}_q$ and $T_{m,n}$ the complexity of the product of two elements of $\mathbb{Z}_q$ with precision m, we present an algorithm of which the complexity in time is $\log(n)\max(T_{n/2,n}, S_{n/2,n})$. In general, one can reach $S_{m,n} = O(n^{\max(1.19,\mu)+\frac{1}{2}}m^{\mu})$. Since a norm can be computed in $O(m^{\mu+\frac{1}{2}}n^{\mu})$ [14], we obtain an algorithm to count points the complexity of which is $O(n^{\max(1.19,\mu)+\mu+\frac{1}{2}} \log n)$ in time without any precomputation and $O(n^{\frac{5}{2}})$ in space. If the base field admits a Gaussian Normal Basis, the complexity of the Frobenius substitution can be reduced as low as $O(nm)$ following [7]. We have in this case an algorithm with $O(\log(n)n^{2\mu})$ time complexity and $O(n^2)$ space complexity. We also describe how our ideas apply to the AGM algorithm.

The paper is organized as follows. After some notations, we investigate how generalized Artin-Schreier's Equations can be solved (section 2) in order to describe the new algorithm in a quite general setting (section 3). Then, we explain where the previous algorithm should be plugged for counting points on elliptic curves and give results of experiments that we have performed (section 4).

Notations and complexity hypothesis. As in the introduction, we will assume that the multiplication of two n-bit length integers takes $O(n^\mu)$ bit operations. Classically, with FFT integer multiplication algorithm, $\mu = 1 + \epsilon$.

Throughout the paper, p is a fixed small prime, $q = p^n$ and $\mathbb{F}_q$ is the finite field with q elements. We shall denote by $\mathbb{Z}_q$ the valuation ring of the unramified extension of degree n of $\mathbb{Q}_p$, σ will denote the Frobenius substitution of the fraction field of $\mathbb{Z}_q$ considered as an extension of $\mathbb{Q}_p$ and v will denote the non archimedian valuation of $\mathbb{Z}_q$. For each $m \in \mathbb{N}^*$, we have a canonical projection $\pi_m : \mathbb{Z}_q \to \mathbb{Z}_q/p^m\mathbb{Z}_q$ and we will put $\pi = \pi_1$ for the projection onto the finite field $\mathbb{F}_q$.

Many of mathematical objects involved in the algorithms described in the following can be seen as lists of elements of $\mathbb{Z}_p$. Formally speaking, images by π_m of these elements of $\mathbb{Z}_p$ will be in these algorithms said to be the mathematical objects computed "at precision m" or "at accuracy m" or "modulo p^m". Following the conventions of [12], we will denote by $T_{m,n}$ the complexity of multiplying two elements of $\mathbb{Z}_{p^n}$ with accuracy m. Furthermore, in the following $S_{m,n}$ will be the complexity in time to compute at accuracy m the image of an element of $\mathbb{Z}_q$ by a power of the Frobenius substitution.

If E is an elliptic curve over $\mathbb{F}_q$, $j(E)$ will be its j-invariant and $E^\uparrow$ the canonical lift of E. We will assume that $j(E) \in \mathbb{F}_q \setminus \mathbb{F}_{p^2}$. The point at infinity of an elliptic curve will be denoted by $\mathcal{O}$.

2 Arithmetic with p-adic Elements

2.1 Representing $\mathbb{Z}_q$

Polynomial Basis. Let L/K be a finite separated algebraic extension of a field K of degree n. By the primitive element theorem and the definition of an algebraic extension, there exists non canonical isomorphisms

$$\varphi : L \to K[t]/F(t)K[t],$$

with $F(t)$ an irreducible polynomial of $K[t]$ of degree n. Such an isomorphism yields a basis of L/K called a polynomial basis. It provides a classical way to work with L which consists in handling equivalence classes in the quotient of the commutative ring $K[t]$ by one of its maximal ideal $F(t)K[t]$. In the specific setting of p-adics, each equivalence class is uniquely determined by a polynomial of $\mathbb{Q}_p[t]$ of degree strictly smaller than n. Consequently, any element of $\mathbb{Z}_q$ can be seen as a polynomial of degree smaller than n. With such a representation, adding two elements of $\mathbb{Z}_q$ is the same as adding two elements of $\mathbb{Z}_p[t]$. Multiplying two elements of $\mathbb{Z}_q$ is the same as multiplying two elements of $\mathbb{Z}_p[t]$ and reducing the result modulo $F(t)$. This yields $T_{m,n} = O(n^\mu m^\mu)$ for multiplying two elements at precision m.

Since there are numerous irreducible elements of degree n in $\mathbb{Z}_p[t]$, there are numerous ways to represent $\mathbb{Z}_q$. In order to have a Frobenius computation as fast as possible, it is better in our case to use a polynomial with a small number of terms, each term defined over $\mathbb{F}_p$ (cf. section 2.2).

Gaussian Normal Basis. If moreover we suppose that L/K is a Galois extension and that $L = K(\alpha)$, then we know that $(\sigma^i(\alpha)),\ 0 \leq i \leq n-1$ is a basis of L/K called a normal basis. This basis is well suited to compute the action of the Galois group of L/K. Lifting to $\mathbb{Z}_q$ Gaussian Normal Basis defined on finite fields [9], we get the following result [7].

Proposition 1. *Let q be a prime or a prime power, and n, t be positive integers such that $nt+1$ is a prime not dividing q. Let γ be a primitive $(nt+1)-th$ root of unity in some extension field of of $\mathbb{Q}_p$. If $\gcd(Mt/e, n) = 1$ where e denotes the order of q modulo $nt+1$, then for any primitive $t-th$ root of unity τ in $\mathbb{Z}/(nt+1)\mathbb{Z}$,*

$$\alpha = \sum_{i=0}^{t-1} \gamma^{\tau^i},$$

generates a normal basis over $\mathbb{Q}_p$ called a Gaussian Normal Basis (GNB) of type t. Furthermore, $[\mathbb{Q}_p(\alpha) : \mathbb{Q}_p] = n$.

H. Y. Kim et al. handled elements of $\mathbb{Z}_q$ with such a representation by working in the basis defined by γ. This yields $T_{m,n} = O((tnm)^\mu)$ for multiplying two elements at precision m.

2.2 Lifting the Frobenius

We now study the complexity of the Frobenius substitution σ^k for $k \in \mathbb{Z}$.

Polynomial Basis. Let F be the defining polynomial of the extension $\mathbb{Z}_q$ which can be chosen to have very few non zero coefficients in $\mathbb{F}_p$ ($O(\log n)$ terms is a reasonable assumption in general). If t is a root of F , then we can compute $\sigma^k(t)$ with precision m by the use of a Newton iteration applied to the polynomial F and initial term t^{p^k}. The cost of the operation is $O(\log^2(n) n^\mu m^\mu)$ bit operations if F has got $O(\log(n))$ terms.

Also, we can write every elements of $\mathbb{Z}_q$ as a sum $x = \sum_{i=0}^{n-1} x_i t^i$ with $x_i \in \mathbb{Z}_p$. We have

$$\sigma^k(x) = \sum_{i=0}^{n-1} x_i \sigma^k(t)^i, \ \forall k \in \mathbb{Z}, \tag{1}$$

and to get the result, it remains to substitute $\sigma^k(t)$ as a polynomial in equation (1). This substitution can be easily done in $n(nm)^\mu$ bit operations.

Vercauteren pointed out to the authors that an algorithm due to Brent and Kung [1] can be improved to decrease significantly this complexity [17]. Briefly, Brent and Kung's algorithm consists in writing x as $\sqrt{n}$ blocks of $\sqrt{n}$ terms, in precomputing $\sqrt{n}$ powers of $\sigma^k(t)$, in substituting in each block t by $\sigma^k(t)$ and finally in obtaining the result with $\sqrt{n}$ additional multiplications in $\mathbb{Z}_q$. Vercauteren's idea is to replace the $\sqrt{n}$ substitutions involved in Brent and Kung's algorithm by the product of a $\sqrt{n} \times \sqrt{n}$ matrix (obtained with the coefficients in $\mathbb{Z}_p$ of x) by a $\sqrt{n} \times n$ matrix (obtained with the coefficients of the powers of $\sigma^k(t)$). This can be done with $\sqrt{n}$ matrix products of size $\sqrt{n} \times \sqrt{n}$. Coppersmith and Winograd [3] have an asymptotic time complexity for a $D \times D$ matrix multiplication of $O(D^{2.38})$, but with a large constant factor. In practice, Strassen's algorithm [16] is often used, it has a $O(D^{2.81})$ time complexity.

Therefore, we have from a theoretical (resp. practical) viewpoint $S_{m,n} = O(n^{\max(1.19,\mu)+\frac{1}{2}}m^{\mu})$ (resp. $O(n^{\max(1.4,\mu)+\frac{1}{2}}m^{\mu})$) in this case. Furthermore we have to store $\lfloor\sqrt{n}\rfloor$ polynomials, so a space complexity of $O(n^{\frac{3}{2}}m)$.

Remark 1. Satoh, Skjernaa and Taguchi [12] exhibit an other way to compute σ or σ^{-1}. It consists in working in a basis generated by a $q-1$-th root of unity ψ in $\mathbb{Z}_q$. This can be done at precision m at the expense of a precomputation with $O(n^{1+\mu}m)$ time complexity. Then, this yields an algorithm to compute σ or σ^{-1} with time complexity equal to $O(n^{\mu}m^{\mu})$. However, it is not clear to the authors how this can be extended in order to have an algorithm with complexity smaller than $O(n^{\max(1.19,\mu)+\frac{1}{2}}m^{\mu})$ to compute σ^k, $k \in \mathbb{Z}$.

Gaussian Normal Basis. In the case of a base field handled with a Gaussian Normal Basis of type t as described at the end of section 2.1, computing σ^k can be done by a simple permutation of the nt components of x. This can be easily done in $S_{m,n} = O(nmt)$ bit operations. A more elaborated implementation strategy (with indexes) yields a $O(n)$ time complexity [7].

2.3 Computing p-adic Norms

Polynomial Basis. Satoh outlines that when $\mathrm{ord}_p(a-1) > \frac{1}{p-1}$, the following formula can be used

$$N_{\mathbb{Z}_q/\mathbb{Z}_p}(a) = \exp(\mathrm{Tr}_{\mathbb{Z}_q/\mathbb{Z}_p}(\log a)).$$

This yields a $O(n^{\mu}m^{\mu+\frac{1}{2}})$ time complexity with space equal to $O(nm)$ with the clever algorithm described in [12].

In the more general case $a \in \mathbb{Z}_q^{\times}$ and p odd, one can compute the norm of an element by

$$N_{\mathbb{Z}_q/\mathbb{Z}_p}(a) = \mathrm{T}(N_{\mathbb{F}_q/\mathbb{F}_p}(\pi(a))N_{\mathbb{Z}_q/\mathbb{Z}_p}(\mathrm{T}(\pi(a)^{-1})a),$$

where T is the Teichmüller lifting map [14]. The complexity of this method is the same as the previous one, except when m is much larger than n. In this last case, the complexity of Satoh's approach is $O(n^{\mu+1}m^{\mu})$.

In fact, we can use the algorithm `NewtonLift` (cf. section 3) with a polynomial ϕ given as $\phi(x,y) = x^p - y$, to compute the Teichmüller lifting map too.

Gaussian Normal Basis. In a Gaussian Normal Basis of type t, H. Y. Kim et al. described an algorithm of the type "divide and conquer" in order to compute the norm of an element of $\mathbb{Z}_q$ with precision m. This algorithm has got a $O((\log n)n^{\mu}m^{\mu})$ time complexity and a $O(n^{\mu}m^{\mu})$ space complexity.

2.4 Computing Roots of Generalized Artin-Schreier's Equations

We recall that if $\mathbb{F}_q$ is a field of characteristic p, an Artin-Schreier's equation is an equation of the form $x^p - x + \beta = 0$ with $\beta \in \mathbb{F}_q$. Such an equation is known to have a solution provided that $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_p}(\beta) = 0$. In an slightly different setting, we will say that an equation is a generalized Artin-Schreier's equation if it can be written as

$$\sigma(x) + ax + b = 0, \text{ with } a, b \in \mathbb{Z}_q. \tag{2}$$

In particular, π applied to equation (2) gives all classical Artin-Schreier's equations. We present here an algorithm to find a root of such an equation.

Algorithm 2.1 `ArtinSchreierRoot`

Algorithm to solve generalized Artin-Schreier's equation.

Input: a and b in $\mathbb{Z}_q/p^m\mathbb{Z}_q$, m and ν in $\mathbb{N}$.
Output: A and B such that a solution of $\sigma(x) = ax + b \bmod p^m$ satisfies $\sigma^\nu(x) = \sigma^\nu(A)x + \sigma^\nu(B) \bmod p^m$.

```
Step 1. if $\nu = 1$ then return $\sigma^{n-1}(a) \bmod p^m$, $\sigma^{n-1}(b) \bmod p^m$;
Step 2. $A$, $B$ := ArtinSchreierRoot($a$, $b$, $m$, $\lfloor\frac{\nu}{2}\rfloor$);
Step 3. $A$, $B$ := $A\sigma^{n-\lfloor\frac{\nu}{2}\rfloor}(A) \bmod p^m, A\sigma^{n-\lfloor\frac{\nu}{2}\rfloor}(B) + B \bmod p^m$;
Step 4. if $\nu \bmod 2 = 1$ then $A$, $B$ := $A\sigma^{n-\nu}(a) \bmod p^m$, $A\sigma^{n-\nu}(b) + B \bmod p^m$;
Step 5. return $A, B$;
```

Correctness. We now explain that the algorithm returns the right result. By an easy recurrence with starting point $\sigma(x) = ax + b$, we can write that for all $k \in \mathbb{N}$, $\sigma^k(x) \equiv a_k x + b_k \bmod p^m$. We know that $\sigma^n(x) = x$, which means that $(1 - a_n)x = b_n$. To compute a_n and b_n, algorithm 2.1 is an adaptation of the classical "square and multiply" algorithm (used for exponentiations) based on the following composition formula,

$$\forall k, k' \in \mathbb{Z}^2,\ \sigma^{k+k'}(x) = \sigma^{k'}(a_k)a_{k'}x + \sigma^{k'}(a_k)b_{k'} + \sigma^{k'}(b_k).$$

Complexity. The algorithm goes through step 3 or step 4 a number of times which is $O(\log n)$ and these steps are performed in $\max(S_{m,n}, T_{m,n})$. Therefore, the complexity in time, of this algorithm is $O(\log n)\max(S_{m,n}, T_{m,n})$. We have showed previously that the space complexity of $S_{m,n}$ is $O(n^2)$ if the base field admits a Gaussian Normal Basis and $O(n^{\frac{5}{2}})$ otherwise.

Remark 2. If the valuation of a is greater than zero, it should be noted that some implementation tricks can be used to improve the constant term of the complexity of the algorithm `ArtinSchreierRoot`.

- First, if we put $(A, B) = \texttt{ArtinSchreierRoot}(a, b, m, \nu)$, we have $v(A) \geq \nu$. As a consequence, if $m \leq \nu$, then $A = 0 \bmod p^m$.
- Second, if we put $\kappa = \lfloor\frac{\nu}{2}\rfloor$ in the expression $A\sigma^{n-\kappa}(A) \bmod p^m$ (resp. $A\sigma^{n-\kappa}(B) + B$) computed in step 3, we have that $v(A\sigma^{n-\kappa}(A)) \geq 2\kappa$ (resp. $v(A\sigma^{n-\kappa}(B)) \geq \kappa$) and so this computation has to be done only to precision $m - 2\kappa$ (resp. $m - \kappa$). Similar optimizations hold for step 4.

Remark 3. Any fast exponentiating algorithm based on "addition-subtraction" chains can be adapted in this case, this yields in practice an easy speeding up.

Remark 4. Algorithm 2.1 is obviously still valid if we replace σ by any element Σ of $\mathrm{Gal}(L/\mathbb{Q}_p)$ with L the fraction field of $\mathbb{Z}_q$. In particular, it is still true for σ^{-1}.

3 Lifting Algorithms

In this section, we describe an algorithm which solves a general problem involved in the counting points algorithms considered in section 4. This problem consists for a fixed $m \in \mathbb{N}$ in finding a root in $\mathbb{Z}_q$ at precision m of an equation of the form

$$\phi(x, \Sigma(x)) = 0, \tag{3}$$

with ϕ a polynomial with coefficients in $\mathbb{Z}_q$ when a solution x_0 at low precision of such an equation is already known.

More specifically, Satoh et al. underlined the importance in the so-called SST algorithm to solve this problem when ϕ is equal to Φ_p, the p-th modular polynomial (to compute the j-invariant of the canonical lift) [12], or equal to $x^p - y$ (for computing the Teichmüller lift) [14]. Since the idea behind the SST algorithm is to use the Taylor expansion of ϕ to increment by one the precision of the root of equation (3) computed at the intermediate steps in the algorithm, it is not difficult to generalize it to a more general ϕ. This is what was done for instance in [5] with a polynomial ϕ related with the AGM method. We give the corresponding algorithm for the general case in section 3.1, before comparing it with a new approach developed in section 3.2.

3.1 SST Algorithm

This algorithm can be seen as an application of the following proposition.

Proposition 2. *Let L be a field complete with respect to a non archimedian valuation v and R be its valuation ring. Let $\phi(x,y) \in R[x,y]$ be a polynomial in two variables, Σ a valuation preserving morphism of R and $\alpha_0 \in R$ such that*

$$v(\phi(\alpha_0, \Sigma(\alpha_0))) > 2v(\frac{\partial \phi}{\partial x}(\alpha_0, \Sigma(\alpha_0))) + 1$$

and

$$v(\frac{\partial \phi}{\partial y}(\alpha_0, \Sigma(\alpha_0))) > v(\frac{\partial \phi}{\partial x}(\alpha_0, \Sigma(\alpha_0)))$$

if we define the sequence α_i by the recurrence relation

$$\alpha_{i+1} = \alpha_i - \frac{\phi(\alpha_i, \Sigma(\alpha_i))}{\frac{\partial \phi}{\partial x}(\alpha_i, \Sigma(\alpha_i))}$$

then the sequence $(\alpha_i, \Sigma(\alpha_i))$ converges toward a root of ϕ in R.

Proof. For convenience, we put $f(\alpha_i) = \phi(\alpha_i, \Sigma(\alpha_i))$, $f_x(\alpha_i) = \partial_x\phi(\alpha_i, \Sigma(\alpha_i))$ and $f_y(\alpha_i) = \partial_y(\alpha_i, \Sigma(\alpha_i))$. We show inductively that $v(\alpha_i) \geq 0$ and that $v(f(\alpha_i)/f_x^2(\alpha_i)) > 0$. In the course of the proof, we will show that $v(f(\alpha_{i+1})) > v(f(\alpha_i))$, which clearly implies the result.

1. As $v(f(\alpha_i)/f_x^2(\alpha_i)) > 0$, we have by hypothesis

$$v(\alpha_{i+1} - \alpha_i) = v(f(\alpha_i)/f_x(\alpha_i)) > v(f_x(\alpha_i)) \geq 0 \tag{4}$$

   and, as $v(\alpha_i) \geq 0$, we obtain that $v(\alpha_{i+1}) \geq 0$.

2. By a Taylor expansion, we have

$$f(\alpha_{i+1}) = f(\alpha_i) - f_x(\alpha_i)\frac{f(\alpha_i)}{f_x(\alpha_i)} - f_y(\alpha_i)\Sigma\left(\frac{f(\alpha_i)}{f_x(\alpha_i)}\right) + \Lambda(\alpha_i)$$

   with

$$v(\Lambda(\alpha_i)) \geq v\left(\frac{f^2(\alpha_i)}{f_x^2(\alpha_i)}\right) \text{ and } v(f_y(\alpha_i)\Sigma\left(\frac{f(\alpha_i)}{f_x(\alpha_i)}\right)) = v\left(\frac{f_y(\alpha_i)}{f_x(\alpha_i)}\right)v(f(\alpha_i)).$$

   As a consequence, we obtain using the hypothesis $v(f_y(\alpha_i)/f_x(\alpha_i)) > 0$, that

$$v(f(\alpha_{i+1})) \geq \min(v\left(\frac{f(\alpha_i)}{f_x^2(\alpha_i)}\right), v\left(\frac{f_y(\alpha_i)}{f_x(\alpha_i)}\right))v(f(\alpha_i)) > v(f(\alpha_i)). \tag{5}$$

   By induction hypothesis we have that $v(f(\alpha_{i+1}) - f(\alpha_i)) = v(f(\alpha_i)/f_x(\alpha_i)) \geq v(f_x(\alpha_i))$ and so, by a Taylor expansion of $f_x(\alpha_{i+1})$ combines with equation (4), we get that $v(f_x(\alpha_{i+1})) = v(f_x(\alpha_i))$ too. As a consequence of equation (5), we obtain

$$v(f(\alpha_{i+1})/f_x^2(\alpha_{i+1})) > v(f(\alpha_i)/f_x^2(\alpha_i)).$$

   From equation (4), this yields $v(\alpha_{i+2} - \alpha_{i+1}) > v(\alpha_{i+1} - \alpha_i)$.

Using proposition 2, SST algorithms can be seen as an application of algorithm 3.1.

Algorithm 3.1 `SSTLift`

Algorithm to compute a root of $\phi(x, \Sigma(x)) \bmod p^m$, knowing a solution x_0 modulo p^{2k+1} where $k = v(\partial\phi/\partial x(x_0, \Sigma(x_0)))$.

INPUT: $x_0 \in \mathbb{Z}_q/p^{2k+1}\mathbb{Z}_q$, $m \in \mathbb{N}$.
OUTPUT: x a solution of $\phi(x, \Sigma(x)) \bmod p^m$.

```
Step 1. $w := \lceil m^{\mu/(\mu+1)} \rceil$; $d$ any lift of $\partial_x\phi(x_0, \Sigma(x_0))$ to $\mathbb{Z}_q/p^{w+k}\mathbb{Z}_q$;
Step 2. $x$ any lift of $x_0$ to $\mathbb{Z}_q/p^{w+k}\mathbb{Z}_q$
Step 3. for $(i := k+1; i < w+k; i := i+1)$ {
Step 4.         $y := \Sigma(x)$;
Step 5.         $x := x - \phi(x, y)/d$;
Step 6. }
Step 7. $y := \Sigma(x) \bmod p^{w+k}$;
Step 8. $D_x := \partial_x\phi(x,y) \bmod p^{w+k}$; $D_y := \partial_y\phi(x,y) \bmod p^{w+k}$;
Step 9. for $(j := 1; jw + k < m; j := j + 1)\{$
Step 10.        Lift $x$ to $\mathbb{Z}_q/p^{(j+1)w+k}\mathbb{Z}_q$;
Step 11.        $y := \Sigma(x) \bmod p^{(j+1)w+k}$;
Step 12.        $V := \phi(x,y) \bmod p^{(j+1)w+k}$;
Step 13.        for $(i := 0; i < w; i := i + 1)$ {
Step 14.                $\Delta_x = -p^{-(jw+i)}V/d$;
Step 15.                $\Delta_y = \Sigma(\Delta_x) \bmod p^{w-i+k}$;
Step 16.                $x := x + p^{jw+i}\Delta_x \bmod p^{(j+1)w+k}$;
Step 17.                $V := V + p^{jw+i}(D_x\Delta_x + D_y\Delta_y) \bmod p^{(j+1)w+k}$;
Step 18.        }
Step 19. }
Step 20. return $x$;
```

In algorithm 3.1, we can take for Σ either the Frobenius or the inverse Frobenius substitution. If we replace ϕ in (3.1) by $\phi(x,y) = \Phi_p(y,x)$ the p-th modular polynomial (resp. by $\phi(x,y) = y^p - x$) we get the algorithm 2.1 of [12] (resp. algorithm 3 of [14]).

Complexity. We refer to the article [12] for a detailed complexity analysis of the algorithm. Briefly, the complexity depends on the time spent in the outer and inner loops which begin respectively at step 9 and 13. Due to the fact to the outer loop is executed at most $\lceil m/w \rceil$ times and the inner loop is performed at most m times, it is easy to deduce that the overall complexity of the algorithm is

$$O(\max((m/w)\max(S_{w,n}, T_{w,n}), \sum_{1\le j\le m/(w+1)} T_{(j+1)w,n})).$$

Since in Satoh's case, $S_{m,n} = T_{m,n} = O(n^\mu m^\mu)$, we recover the $O(\max(m(nw)^\mu, n^\mu m^{\mu+1}w^{-1}))$ complexity proved in [12].

Remark 5. It should be noted that proposition 2 yields a linear convergence.

3.2 Extending Newton's Algorithm

We now present an enhanced version of the preceding algorithm in the case of the morphism Σ is a power of σ. The main idea behind our approach is to slightly modify the well-known Newton's algorithm for computing roots of univariate polynomials over $\mathbb{Z}_q$ in order to recover a quadratic convergence.

Specifically, let $\phi \in \mathbb{Z}_p[x,y]$ be a bivariate polynomial with coefficients in $\mathbb{Z}_q$. Let $x_0 \in \mathbb{Z}_q$ be a zero of the equation $\phi(x, \Sigma(x)) = 0 \bmod p^w$, $w \in \mathbb{N}$. We suppose moreover that we have

$$v(\frac{\partial\phi}{\partial x}(x_0, \Sigma(x_0))) \ge v(\frac{\partial\phi}{\partial y}(x_0, \Sigma(x_0)))$$

and

$$v(\phi(x_0, \Sigma(x_0))) > v(\left(\frac{\partial\phi}{\partial y}\right)^2(x_0, \Sigma(x_0))).$$

The only difficulty against the univariate polynomial case is the composition with Σ. But, since such morphisms preserve valuations, proving a result in this case is very close to the proof for the classical Newton convergence [8, pages 493-494]. We omit it here and prefer to give directly the corresponding algorithm.

Algorithm 3.2 `NewtonLift`

Algorithm to compute a root of $\phi(x, \Sigma(x)) \bmod p^m$, knowing a solution x_0 modulo p^{2k+1} where $k = v(\partial\phi/\partial y(x_0, \Sigma(x_0)))$.

INPUT: $x_0 \in \mathbb{Z}_q/p^{2k+1}\mathbb{Z}_q$, $m \in \mathbb{N}$.
OUTPUT: x a solution of $\phi(x, \Sigma(x)) \bmod p^m$.

```
Step 1. if $m \leq 2k+1$ then return $x_0$;
Step 2. $w := \lceil \frac{m}{2} \rceil + k$;
Step 3. $x :=$ NewtonLift$(x_0, w)$;
Step 4. Lift $x$ to $\mathbb{Z}_q/p^m\mathbb{Z}_q$; $y := \Sigma(x) \bmod p^m$;
Step 5. $\Delta_x := \partial_x\phi(x, y) \bmod p^{w-k}$; $\Delta_y := \partial_y\phi(x, y) \bmod p^{w-k}$;
Step 6. $V := \phi(x, y) \bmod p^m$;
Step 7. $a, b :=$ ArtinSchreierRoot$(-V/(p^{w-k}\Delta_y), -\Delta_x/\Delta_y, w-k, n)$;
Step 8. return $x + p^{w-k}(1-a)^{-1}b$;
```

Correctness. We assume inductively that we know a root x_0 of $\phi(x, \Sigma(x))$ at precision $w = \lceil \frac{m}{2} \rceil + k$ and we explain why the algorithm returns a root of the same equation at precision m.

If we put $f(x) = \phi(x, \Sigma(x))$, we have, once x_0 lifted to precision $2w - 2k$,

$$\forall \delta \in \mathbb{Z}_q,\ f(x_0 + p^{w-k}\delta) - f(x_0) \equiv p^{w-k}(\delta\Delta_x + \Sigma(\delta)\Delta_y) \bmod p^{2w-2k},$$

with $\Delta_x \equiv \partial_x\phi(x_0, \Sigma(x_0)) \bmod p^{w-k}$ and $\Delta_y \equiv \partial_y\phi(x_0, \Sigma(x_0)) \bmod p^{w-k}$. We want to find δ at precision $w - k$ such that $f(x_0 + p^{w-k}\delta) \equiv 0 \bmod p^{2w-2k}$, which we can restate in the following form,

$$-\frac{f(x_0)}{p^{w-k}} \equiv \delta\Delta_x + \Sigma(\delta)\Delta_y \bmod p^{w-k}. \tag{6}$$

Rewriting equation (6) as $\Sigma(\delta) \equiv a\delta + b \bmod p^{w-k}$, with $a = -\Delta_x/\Delta_y$ and $b = -f(x_0)/p^{w-k}\Delta_y$, we recognize an Artin-Schreier's equation since a and b are in $\mathbb{Z}_q$. Calling algorithm 2.1 for Σ with a and b, yields a solution δ at precision $w - k$ and $x + p^{w-k}\delta$ is a root of f with precision at least equal to m.

Complexity. The algorithm calls itself recursively $O(\log n)$ times and the step with the largest cost is the call to `ArtinSchreierRoot` algorithm. The complexity of this call is $O(\log n)\max(S_{w,n}, T_{w,n})$ where w is nearly multiplied by two at each recursive call. Therefore, the complexity of this algorithm is

$O(\log n \max(S_{m,n}, T_{m,n}))$. In general, this yields a $O(m^{\mu} n^{\max(1.19,\mu)+\frac{1}{2}} \log n)$ time complexity and a $O(mn^{\frac{3}{2}})$ space complexity (cf. section 2)). Over Gaussian Normal Basis, this yields a $O(\log(n) n^{\mu} m^{\mu})$ time complexity and a $O(nm)$ space complexity.

4 Application to Point Counting

We now illustrate how we can use algorithm 3.2 to compute the canonical lift of an elliptic curve. We apply these ideas to the Satoh's and Mestre's algorithms.

4.1 Satoh's Algorithm

We quickly recall Satoh's algorithm [14] to count points on elliptic curves E defined over $\mathbb{F}_q$. For this, if f is an isogeny between two elliptic curves E_1 and E_2 and $\tau_i = -X_i/Y_i$ a local parameter of E_i around $\mathcal{O}_i$ then we have an expansion of f around $\mathcal{O}$ as $f^*(\tau_2) = c_1\tau_1 + c_2\tau_2 + \dots$ and we put $\mathrm{lc}(f) = c_1$ the leading coefficient of this expansion.

Let $E^{(i)}$ be the image of the i^{th} iterate of the little Frobenius on E and $V_p^{(i)}$ be the dual of the little Frobenius between $E^{(i-1)}$ and $E^{(i)}$, Satoh's approach is as follows.

1. Let m be the smallest integer such that $p^m > 4\sqrt{q}$.
2. Compute $j(E^{(i-1)\uparrow})$ and $j(E^{(i)\uparrow}) \bmod p^{m+O(1)}$ for some integer i.
3. Compute $c = lc(V_p^{(i)\uparrow})$.
4. Compute $t' = \sqrt{N_{\mathbb{Z}_q/\mathbb{Z}_p}(c)}$, the sign of the square root can be determined following [14].
5. Return $t \in \mathbb{Z}$ satisfying $t \equiv t' \bmod p^m$ and $|t| < 2\sqrt{q}$.

For a detailed analysis of each of these steps we refer to [14].

We emphasize here that step 2 may be improved by the use of the procedure `NewtonLift` applied to

$$\phi(x, y) = \Phi_p(x, y),$$

since, by Kroneker's relation $\phi_p(x, y) = (x^p - y)(y^p - x) \bmod p$, we have

$$\frac{\partial \Phi_p}{\partial x}(x, \sigma(x)) = 0 \bmod p \text{ and } \frac{\partial \Phi_p}{\partial y}(x, \sigma(x)) \neq 0 \bmod p.$$

Finally, the time complexity of this algorithm is trivially the same as that of the algorithm `NewtonLift` (cf. section 3.2) since the time needed to compute $N_{\mathbb{Z}_q/\mathbb{Z}_p}$ at step 4 is slightly smaller (cf. section 2.3).

4.2 Mestre's Algorithm

In [10], Mestre describes a very efficient algorithm to count points on elliptic curves defined over $\mathbb{F}_{2^n}$. It makes use of the algebraic-geometric mean (AGM).

We first describe the one variable version of this algorithm since this version of Mestre's algorithm is usually what is used in practice (cf. [5]). Let E be an elliptic curve defined over the field $\mathbb{F}_{2^n}$ by an equation $y^2 + xy = x^3 + a_6$. We can consider the sequence of elements of $\mathbb{Z}_{2^n}$ defined by

$$\alpha_{n+1} = \frac{1+\alpha_n}{2\sqrt{\alpha_n}}, \tag{7}$$

with first term equal to $\alpha_0 = 1 + 8a_6 \in \mathbb{Z}_{2^n}$. The square root in equation (7) is chosen such that $\sqrt{1+8t} = 1 + 4t'$ with $t, t' \in \mathbb{Z}_{2^n}$. Then it turns out that

$$\operatorname{Tr}(\mathrm{Fr}_q) = N_{\mathbb{Z}_{2^n}/\mathbb{Z}_2}\left(\frac{2\alpha_{\lceil n/2\rceil+3}}{1+\alpha_{\lceil n/2\rceil+3}}\right).$$

Another important fact is that $\alpha_{n+1} \simeq \sigma(\alpha_n) \bmod 2^{n+3}$. Therefore, as in Satoh's algorithm, the AGM method can be clearly divided in two parts. The first part is intended to compute a root of $4x\sigma(x)^2 = (1+x)^2$ at precision $\lceil n/2\rceil + 3$. The second part yields the trace of the Frobenius with a norm computation.

The first part can be solved with algorithm 3.1 (cf. [5]). But it may be improved by the use of the procedure `NewtonLift` applied to

$$\phi(x, y) = 4xy^2 - (1+x)^2,$$

since $v(\frac{\partial\phi}{\partial x}(\alpha_0, \sigma(\alpha_0))) \geq v(\frac{\partial\phi}{\partial y}(\alpha_0, \sigma(\alpha_0)))$.

As in section 4.1, the time complexity of this algorithm is trivially the same as that of the algorithm `NewtonLift` (cf. section 3.2).

4.3 Results

We implemented the application to AGM of the `NewtonLift` algorithm described in section 4.2. This was done with a finite field `C` library called `ZEN` [2] built over GMP [4]. We measured the time needed to compute the number of points on elliptic curves over $\mathbb{F}_{2^n}$ with n slightly larger than 1000, 2000, 4000, 8000, 16000, 32000 and 65000 on a 731 MHz DEC alpha computer. We give these timings in Table 1 and compare them with timings for finite fields of similar sizes measured with our implementation of the original AGM method as published by Mestre [10].

Let us note that at the time of writing the largest such computation ever done is a 130020-bit computation due to Harley [6].

Remark 6. We designed our implementation for finite fields with a Gaussian Normal Basis of type 1 in order to experimentally check the quadratic behavior of the algorithm. Therefore, the exponents n used for our experiments are

Table 1. Timings for counting points on elliptic curves defined over $\mathbb{F}_{2^n}$.

n	`NewtonAGM`, GNB type 1			Original AGM [10]		
	Lift	Norm	Total	Lift	Cycle	Total
1000	-	-	-	1mn 6s	4mn 22s	5mn 28s
1018	2.4s	1.6s	4s	-	-	-
2003	-	-	-	16mn 28s	57mn 49s	1h 13mn
2052	10.1s	7.2s	17.3s	-	-	-
4001	-	-	-	2h 3mn	8h 56mn	10h 59mn
4098	1mn	45s	1mn 45s	-	-	-
8009	-	-	-	23h 5mn	4d 2h	5d 1h
8218	6mn 30s	4mn 30s	11mn	-	-	-
16420	34mn	23mn	57mn	-	-	-
32770	3h 17mn	2h 18mn	5h 35mn	-	-	-
65538	15h 45mn	13h 20mn	1d 5h	-	-	-
100002	1d 18h	1d 16h	3d 10h	-	-	-

even. Finite fields with prime exponents are usually preferred for cryptographic purposes. They can be handled through Gaussian Normal of larger types. For instance, since multiplying two elements over a GNB of type 2 can be done in two times the time needed to multiply two elements over a GNB of type 1 for finite fields of similar size [7], it is not difficult to derive timings of such an implementation for counting points over GNB of type 2. Similar arguments hold for larger types.

5 Conclusion

We have described the first point counting algorithm the time complexity of which is as a function of n equal to $O(n^{2+\epsilon})$. We reach this complexity for finite fields with Gaussian Normal Basis and this seems to be what we can reasonably hope for such a problem. More generally, thanks to Vercauteren's ideas, this algorithm achieves without any precomputation a time complexity of $O(n^{2.69+\epsilon}\log n)$ bit operations with a space consumption of $O(n^{2.5})$. Furthermore, it should be noted from a practical viewpoint that implementing this algorithm is probably easier than implementing previous known algorithms.

In a forthcoming paper, we deal with a higher dimensional version of this algorithm in order to generalize AGM algorithm to the higher genus case (we refer to [11]). As a consequence, it is expected that the conclusions related to the complexity of elliptic curve point counting algorithms should apply to some higher genus cases.

Acknowledgments. The authors would like to thank Frederik Vercauteren for its valuable comments about the computation of σ^k and Pierrick Gaudry for its remarks on preliminary versions of this paper.

References

[1] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25:581–595, 1978.

[2] F. Chabaud and R. Lercier. *ZEN, User Manual.* Available at *http://www.di.ens.fr/˜zen/.*

[3] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, May 1987. New York City.

[4] Free Software Foundation. GNU MP Library. Available at *http://www.swox.com/gmp/.*

[5] Pierrick 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*, volume 2501 of *Lecture Notes in Computer Science*, pages 311–327. Springer-Verlag, December 2002.

[6] Robert Harley. Asymptotically optimal p-adic point-counting. Email at the Number Theory List, December 2002.

[7] Hae Young Kim, Jung Youl Park, Jung Hee Cheon, Je Hong Park, Jae Heon Kim, and Sang Geun Hahn. Fast Elliptic Curve Point Counting Using Gaussian Normal Basis. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, 5th International Symposium, ANTS-V*, volume 2369 of *Lecture Notes in Computer Science*, pages 292–307. Springer-Verlag, July 2002.

[8] Serge Lang. *Algebra (3rd revised edition)*, volume 211. Graduate Texts in Mathematics, 2002.

[9] Alfred J. Menezes, Ian F. Blake, XuHong Gao, Ronald C. Mullin, Scott A. Vanstone, and Tomik Yaghoobian. *Applications of finite fields.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1993.

[10] Jean-François Mestre. Lettre à Gaudry et Harley. Available at *http://www.math.jussieu/˜mestre*, 2001.

[11] Jean-François Mestre. Notes of a talk given at the seminar of cryptography of Rennes. Available at *http://www.maths.univ-rennes1.fr/crypto/2001-02/mestre.ps*, 2002.

[12] T. Satoh, B. Skjernaa, and Y. Taguchi. Fast Computation of Canonical Lifts of Elliptic Curves and its Application to Point Counting, August 2001. Preprint.

[13] Takakazu Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, 15(4):247–270, 2000.

[14] Takakazu Satoh. On p-adic Point Counting Algorithms for Elliptic Curves over Finite Fields. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, 5th International Symposium, ANTS-V*, pages 43–66. Springer-Verlag, July 2002.

[15] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théorie des nombres de Bordeaux*, 7:483–494, 1998.

[16] V. Strassen. Gaussian Elimination is Not Optimal. *Numerische Mathematik*, 13:354–356, 1969.

[17] F. Vercauteren. On AGM. Personal communication, November 2002.

[18] Frederik Vercauteren, Bart Preneel, and Joos Vandewalle. A Memory Efficient Version of Satoh's Algorithm. In *Advances in Cryptology—EUROCRYPT 2001 (Innsbruck)*, volume 2045 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2001.