An extension of Satoh's algorithm and its implementation

Mireille Fouquet*, Pierrick Gaudry* and Robert Harley†‡

October 24, 2000

Abstract

We describe a fast algorithm for counting points on elliptic curves defined over finite fields of small characteristic, following Satoh. Our main contribution is an extension to characteristics two and three. We give a detailed description with the optimisations necessary for an efficient implementation. Finally we give the number of points we have computed on a "random" curve defined over the field $\mathbb{F}_q$ with $q = 2^{8009}$.

1 Introduction

1.1 A short history of point counting

The first polynomial-time algorithm for counting points on elliptic curves over finite fields was described by Schoof in [Sch85]. The basic idea was to find the cardinality, modulo sufficiently many small primes ℓ using the ℓ–torsion points and then to recover the exact result using the Chinese Remainder Theorem. To compute with ℓ–torsion points, Schoof suggested doing polynomial arithmetic modulo the ℓ–division polynomials.

The time required for point-counting over $\mathbb{F}_q$ with Schoof's algorithm is $O(\log^{5+\varepsilon} q)$ with asymptotically fast algorithms for arithmetic (or $O(\log^8 q)$ with naïve arithmetic).

To reduce the run-time, in large characteristic, Elkies [Elk98] and Atkin [Atk92] proposed replacing the ℓ–division polynomials with lower degree ones found using the factorization of modular polynomials. In this way they reduced the degree from $O(\ell^2)$ to $O(\ell)$ yielding the so-called SEA algorithm (see [Sch95]). The improved run-time, under reasonable hypotheses, is $O(\log^{4+\varepsilon} q)$ (or $O(\log^6 q)$).

Further work by Morain [Mor95a], Müller [Mül95], Dewaghe [Dew98] and others led to more efficient implementations of SEA. Morain carried out actual computations for q as large as $10^{499} + 153$ [Mor95b]. Couveignes extended SEA to work in small characteristic [Cou94], [Cou96] and Lercier gave some improvements specific to characteristic two [Ler97], [JL]. Vercauteren reached $q = 2^{1999}$ with an optimized implementation of these ideas [Ver99].

1.2 Satoh's algorithm

Recently in [Sat00], Satoh outlined a new algorithm with run-time $O(\log^{3+\varepsilon} q)$ (or $O(\log^5 q)$) for small fixed characteristic $p \geq 5$ and suggested that it could be extended to characteristics 2 and 3. The basic idea of this new algorithm is completely different from SEA.

*LIX, École polytechnique, 91128 Palaiseau Cedex, France
†ArgoTech, 26 ter rue Nicolaï, 75012 Paris, France
‡Projet Cristal, INRIA, Domaine de Voluceau - Rocquencourt, 78153 Le Chesnay, France

First of all the curve given over $\mathbb{F}_q$, where $q = p^d$, is lifted to a curve defined over a certain p–adic ring $\mathbb{Z}_q$ above $\mathbb{F}_q$. Intuitively, $\mathbb{Z}_q$ is to $\mathbb{F}_q$ as the p–adic integers $\mathbb{Z}_p$ are to the prime field $\mathbb{F}_p$. More precisely, $\mathbb{Z}_q$ is the unique (unramified) discrete valuation ring having residue field $\mathbb{F}_q$. By a result of Lubin-Serre-Tate, there is a canonical way to lift the curve by lifting its j–invariant j from $\mathbb{F}_q$ to $\mathbb{Z}_q$ using the modular polynomial Φ_p. A direct implementation based on this result would be slow due to computations of the (rather complicated) Frobenius in $\mathbb{Z}_q$.

Satoh devised a much more efficient algorithm based on the following insight: rather than lifting j up to J in isolation, it is faster to lift j along with all its conjugates j_i simultaneously. Indeed writing out the equations $\Phi_p(J_i, J_{i+1}) = 0$ for $0 \leq i < d$ yields an algebraic system over $\mathbb{Z}_q$ without Frobenius, which can be solved quickly by a multi-variate Newton iteration.

The first stage is thus to compute all the J_i's to p–adic precision $O(p^n)$ where $n \approx \frac{d}{2}$. Next for each i we find a curve defined over $\mathbb{Z}_q$ that has j–invariant J_i. This is done with ordinary Newton iterations to find its coefficients to sufficient precision. Then for each curve, we compute the kernel of the dual isogeny of the Frobenius with a generalized Newton iteration.

Now, the trace of Frobenius can be written as a norm from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ of a certain coefficient. The square of each of the conjugates of the coefficient can be computed using Vélu's formulae. To finish, we compute the product of these conjugates and take its square root, yielding the trace c of Frobenius (except for its sign, which can easily be determined).

Since $|c| \leq 2\sqrt{q}$, the number of points $q + 1 - c$ can be determined exactly by working to sufficient precision.

1.3 Overview of the paper

The first purpose of the present paper is to describe the algorithm for $p \geq 5$ due to Satoh, with the details necessary for an efficient implementation, in an easy to understand fashion. We make the algorithm effective and apply some improvements.

An important part of our contribution is to extend Satoh's method to an algorithm for characteristics $p = 2$ and $p = 3$. The extension to $p = 3$ is relatively straightforward, whereas the $p = 2$ case introduces some difficulties not present in odd characteristic. We also describe how to minimise the constant factor in the $O(d^3 \log p)$ memory required by these algorithms.

Finally we present actual timings and in particular result we have computed for $q = 2^{8009}$ using an asymptotically fast implementation.

2 Local rings

2.1 The ring $\mathbb{Z}_p$ of p–adic integers

Let π_n be the projection from $\mathbb{Z}/p^{n+1}\mathbb{Z}$ onto $\mathbb{Z}/p^n\mathbb{Z}$. This projection is a ring homomorphism. One can give a formal definition of p–adic integers as follows.

Definition 2.1 *A p–adic integer is a sequence $x = (x_1, x_2, \ldots, x_n, \ldots)$, with $x_n \in \mathbb{Z}/p^n\mathbb{Z}$ and such that $\pi_n(x_{n+1}) = x_n$ for $n \geq 1$. The ring of p–adic integers is denoted by $\mathbb{Z}_p$.*

The sum and the product in $\mathbb{Z}_p$ are defined coordinate-wise in the natural way. Note that $\mathbb{Z}_p$ is a discrete valuation ring. Its only non-zero prime ideal is $p\mathbb{Z}_p$ and the residue field $\mathbb{Z}_p/p\mathbb{Z}_p$ is (isomorphic to) $\mathbb{F}_p$.

We extend the definition of π_n to include the projection from $\mathbb{Z}_p$ to $\mathbb{Z}/p^n\mathbb{Z}$ i.e. $x \mapsto x_n$. Clearly, once x_n is known, one can recover x_k for $k < n$ easily by projection down to $\mathbb{Z}/p^k\mathbb{Z}$. We set $\pi = \pi_1$. Note that π can be considered as projecting to $\mathbb{F}_p$.

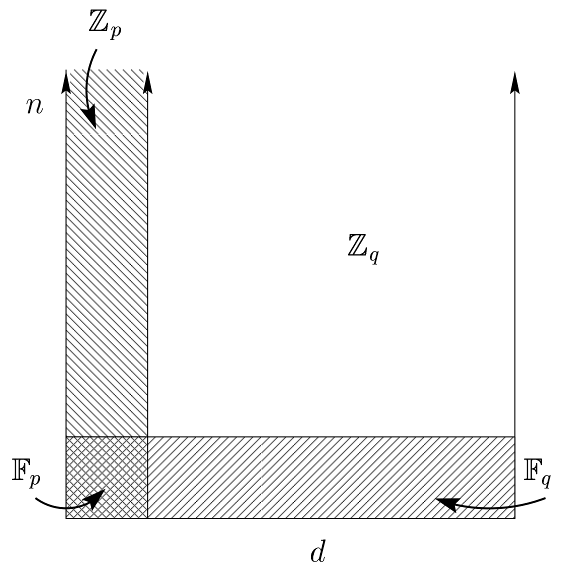
The invertible elements of $\mathbb{Z}_p$ are those not in $p\mathbb{Z}_p$ i.e., those with $\pi(x) \neq 0$. For further details on p–adic integers, see [Ser70, Chapter II].

2.2 The ring $\mathbb{Z}_q$

Let $q = p^d$ with p prime. Let $f(t)$ be a monic polynomial in $\mathbb{Z}_p[t]$ of degree d such that the polynomial $\pi(f)$ obtained by projecting the coefficients is irreducible in $\mathbb{F}_p[t]$.

Definition 2.2 *The ring $\mathbb{Z}_q$ is $\mathbb{Z}_p[t]$ modulo (the ideal generated by) $f(t)$.*

The following graph gives an idea of how $\mathbb{F}_p$, $\mathbb{F}_q$, $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are related:



An element a of $\mathbb{Z}_q$ can be represented as a polynomial $a_{d-1}t^{d-1} + \cdots + a_1 t + a_0$. The sum and the product in $\mathbb{Z}_q$ are ordinary polynomial sums and products taken modulo $f(t)$. Note that $\mathbb{Z}_q$ contains $\mathbb{Z}_p$ as a sub-ring (just set $a_1 = a_2 = \cdots = a_{d-1} = 0$).

Furthermore, $\mathbb{Z}_q$ is an unramified discrete valuation ring. Its only non-zero prime ideal is $p\mathbb{Z}_q$ and the residue field $\mathbb{Z}_q/p\mathbb{Z}_q$ is (isomorphic to) $\mathbb{F}_q$. In fact these conditions characterize $\mathbb{Z}_q$. In particular it is isomorphic to the ring $W(\mathbb{F}_q)$ of Witt vectors; the representation as Witt vectors is used in the theory below but we found it inconvenient for actual computation.

We extend the definitions of π_n and π to $\mathbb{Z}_q$ in the natural way. Note that π can be considered as projecting to $\mathbb{F}_q$. In the following we will use the notation " mod p^n " to indicate that a given relation holds modulo the ideal $p^n\mathbb{Z}_q$. The invertible elements of $\mathbb{Z}_q$ are those with $\pi(x) \neq 0$. For further details, see [Ser68, Chapters I and II].

2.3 Little Frobenius of $\mathbb{Z}_q$

The starting point is the well-known p-th power Frobenius action on the field $\mathbb{F}_q$.

Definition 2.3 *The* little Frobenius *on $\mathbb{F}_q$ is the field automorphism $\sigma : x \mapsto x^p$, fixing the subfield $\mathbb{F}_p$.*

In order to define the little Frobenius Σ lifted to $\mathbb{Z}_q$, we first need to give some definitions. We remark that although Σ can be computed as indicated here, doing so is rather slow in practice.

Satoh's algorithm (and our improvements of it) go out of their way to avoid computing Σ completely and as a result the reader can skip this subsection on a first reading.

First we define a particular lift of scalars from $\mathbb{F}_q$ to $\mathbb{Z}_q$ which respects the multiplicative structure.

Definition 2.4 *The* Teichmüller lift *is the map $\omega : \mathbb{F}_q \to \mathbb{Z}_q$ defined by: $\omega(0) = 0$ and for x non-zero, $\omega(x)$ is the unique $(q-1)$-th root of unity in $\mathbb{Z}_q$ such that $\pi(\omega(x)) = x$.*

The choice of name here is motivated by similarity with the Teichmüller character from p–adic analysis. Note that the Teichmüller lift can be computed using a Newton iteration based on the function $f(x) = x^{q-1} - 1$.

Next we define a decomposition of elements of $\mathbb{Z}_q$ which is close to the usual representation of Witt vectors[1]. See Chapter II of [Ser68] for details on Witt vectors.

Definition 2.5 *The* semi-Witt decomposition *of $x \in \mathbb{Z}_q$ is the unique sequence $(x_i)_{i\geq 0}$ of $x_i \in \mathbb{F}_q$ such that $x = \sum_{i\geq 0} \omega(x_i)p^i$.*

The coordinates x_i can be computed one after another in the obvious way by $x_0 = \pi(x)$, then $x_1 = \pi(\frac{x-\omega(x_0)}{p})$ and so on.

Now we can define Σ, the little Frobenius lifted from $\mathbb{F}_q$ to $\mathbb{Z}_q$. Naturally it satisfies $\pi(\Sigma) = \sigma$ where σ is the little Frobenius on $\mathbb{F}_q$; however Σ is *not* a simple powering like σ.

Definition 2.6 *The Frobenius $\Sigma : \mathbb{Z}_q \to \mathbb{Z}_q$ is as follows. For any $x \in \mathbb{Z}_q$, let $(x_i)_{i\geq 0}$ be its semi-Witt decomposition. Then $\Sigma(x)$ is the element of $\mathbb{Z}_q$ with decomposition $(x_i^p)_{i\geq 0}$. In other words it is $\sum_{i\geq 0} \omega(x_i^p)p^i$ or equivalently $\sum_{i\geq 0} \omega(x_i)^p p^i$.*

2.4 Computations in $\mathbb{Z}_p$ and $\mathbb{Z}_q$

For the purpose of computation one considers p–adic integers to some working precision, say n. That is to say one represents $a \in \mathbb{Z}_p$ by an approximation $a_n \in \mathbb{Z}/p^n\mathbb{Z}$ and the ring operations are carried out modulo p^n.

The inverse of an invertible $a \in \mathbb{Z}_p$ can be computed by the simple Newton iteration $x \leftarrow x - x(ax-1)$, initialized with the inverse $1/\pi(a)$ taken in $\mathbb{F}_p$.

Similarly, to represent $a = a_{d-1}t^{d-1} + \cdots + a_0 \in \mathbb{Z}_q$ to some working precision n, one can store the sequence $(\pi_n(a_{d-1}), \ldots, \pi_n(a_0))$. The space required is $s = O(nd \log p)$ and the time for addition or subtraction of two elements is $O(s)$. The polynomial $f(t)$ can be chosen to be sparse and with simple coefficients, so that the reduction modulo $f(t)$ costs only a few additions and subtractions. Multiplication using naïve arithmetic would take $O(s^2)$ whereas using Karatsuba's algorithm, in both the p–adic dimension and in the polynomial dimension, reduces it to $O(s^{\log 3/\log 2})$.

One may also use fast multiplication algorithms. An asymptotically good method is to pack the coefficients into a single large integer and then use a fast algorithm for integer multiplication. Each coefficient should be padded with zero bits to fill a field of $\lceil 2n \log_2 p + \log_2 d \rceil$ bits, so that the coefficients of the product polynomial can be recovered from fields in the integer product. Arithmetic with precision less than $O(\log d)$ takes negligible time so we can assume that the size ℓ of the resulting integers is $\Theta(nd \log p)$. The fastest algorithm for multiplying large integers is Schönhage's using Fast Fourier Transforms modulo Fermat numbers [SS71] in time $O(\ell \log \ell \log \log \ell)$ for ℓ bits.

[1] The difference is simply that for Witt vectors, the coordinates x_i are replaced by $x_i^{p^i}$.

In the following $n = O(d)$ and hence multiplication can be done in time $O(nd \log d \log \log d)$, for fixed p. Note that when $n = \Theta(d)$ the same complexity can be attained, with better constant factors, by doing the FFT's directly on the polynomial representation. We found this approach to be the fastest in practice.

As before, the inverse of an invertible a can be obtained by a simple Newton iteration initialized with $1/\pi(a)$. By varying the precision as described in the next section, division takes time proportional to multiplication.

2.5 Newton iteration

Several algorithms that are described in the present paper are based on Newton iteration. Therefore, we present it here.

Let R be an arbitrary p–adic ring. The following *Newton iteration* allows us to improve an approximate root of a polynomial. Given a good approximation to a root we can efficiently compute a better approximation, with approximately twice the precision.

It will be necessary to be quite careful about what we mean by "precision" throughout. We write $p^k \mid x$ when $x \equiv 0 \mod p^k$. In such a case $p^{-k}x$ is in R. We write $p^k \parallel x$ when $p^k \mid x$ but $p^{k+1} \nmid x$. In such a case $p^{-k}x$ is invertible.

Let $x \in R$ and $f \in R[t]$. Let k be such that $p^k \parallel f'(x)$ and assume $p^{n+k} \mid f(x)$ for some $n > k$. Under such conditions we will say that x is an approximate root of f known to precision n, in the sense that there exists an exact root y in R with $x \equiv y \mod p^n$.

We first prove a lemma showing how a Newton iteration improves a root known to precision n into one known to precision $2n - k$.

Lemma 2.1 *(Quadratic convergence of Newton iteration) Let $x \in R$ and $f \in R[t]$. Let k be such that $p^k \parallel f'(x)$ and assume $p^{n+k} \mid f(x)$ for some $n > k$. Let:*

$$\Delta = \frac{p^{-k} f(x)}{p^{-k} f'(x)}$$

and $y = x - \Delta$. Then $y \equiv x \mod p^n$, $p^{2n} \mid f(y)$ and $p^k \parallel f'(y)$.

Proof : It is clear that $p^n \mid \Delta$ so that $y \equiv x \mod p^n$. By Taylor expansion,

$$f(y) = f(x) - \Delta f'(x) + \Delta^2 \Psi(x),$$

where $\Psi(x)$ is a polynomial with coefficients in R. Since $p^{2n} \mid \Delta^2$, we get $f(y) \equiv f(x) - \Delta f'(x) \equiv 0 \mod p^{2n}$.

Finally since $y \equiv x \mod p^n$, it follows that $f'(y) \equiv f'(x) \mod p^n$. Since $p^k \parallel f'(x)$ and $n > k$ we have $p^k \parallel f'(y)$. $\square$

Repeated application of this lemma gives rise to an algorithm to compute a root of a polynomial to any desired precision, from a sufficiently precise initial root.

Procedure NEWTONITERATIONS
Inputs:

- A desired precision n.
- A polynomial $f \in R[t]$,

- A starting solution $x_0 \in R$ and an integer k with $p^k \parallel f'(x_0)$ and $p^{2k+1} \mid f(x_0)$.

Output: An approximate root $x \in R$ with $x \equiv x_0 \mod p^{k+1}$ and $p^{n+k} \mid f(x)$.

1. IF $n \leq k+1$ THEN $y \leftarrow x_0$; GO TO 5;
2. $n' \leftarrow \lceil \frac{n+k}{2} \rceil$;
3. $x \leftarrow$ NEWTONITERATIONS(n', f, x_0, k);
4. $y \leftarrow x - \frac{p^{-k} f(x)}{p^{-k} f'(x)}$;
5. RETURN y.

For an efficient implementation, it is important to work with the lowest precision possible at each stage of the algorithm. In particular, one should reduce the precision of the Newton iterations in each recursive call. The depth of the recursion necessary to reach a desired precision n is $\log(n)$, so if all operations were done at precision n then the total time taken would be $O(M(n)\log(n))$ where $M(n)$ is the time for multiplication. By doing the top level iteration at full precision, the first recursive call at roughly half precision, the second recursive call at one quarter precision and so on, the time taken is reduced to $O(M(n))$ total.

Furthermore, when calculating the improved root $y = x - f(x)/f'(x)$ in a Newton iteration, one can reduce the precision of some of the calculations by roughly half. The lowest required precisions are indicated next. We will write $x + O(p^n)$ to denote any $y \equiv x \mod p^n$. For instance $xy + O(p^n)$ can be understood to be the product xy computed with precision n.

When an element $a \in R$ is known to precision n then the product $p^k a$ is known to precision $n+k$, in the sense that $p^k(a + O(p^n)) = p^k a + O(p^{n+k})$. Conversely when a is known to precision n and satisfies $p^k \mid a$, where $k \leq n$, then $p^{-k} a$ is known to precision $n-k$. In practice the division and multiplication by p^k are fast operations e.g., right and left shifts when $p = 2$.

Now the Newton iteration can be computed as $y = x - \Delta + O(p^{2n-k})$ where :

$$\Delta = \left(\frac{(f(x) + O(p^{2n})) \cdot p^{-n-k} + O(p^{n-k})}{(f'(x) + O(p^n)) \cdot p^{-k} + O(p^{n-k})} + O(p^{n-k}) \right) \cdot p^n + O(p^{2n-k})$$

3 Frobenius morphisms and the canonical lift

3.1 Frobenius morphisms on curves

Let E be a curve over the finite field $\mathbb{F}_q$, where $q = p^d$. As is well known, we can compute the order of E by computing the trace c of the *q-th power Frobenius endomorphism* on this curve. The number of points is then given by $q+1-c$. According to the Hasse-Weil bound, we have $|c| \leq 2\sqrt{q}$.

The p-th power Frobenius σ on $\mathbb{F}_q$ can clearly be extended to map points (x, y) on E to points $(\sigma(x), \sigma(y))$ on the conjugate curve E^σ obtained by applying σ to the coefficients of E and in fact this map is a (purely inseparable) isogeny, also called the little Frobenius. Furthermore, the construction can be repeated giving an isogeny from E^σ to E^{σ^2} and so on. Repeating it d times gives a cycle of isogenies which returns to the initial curve E. Writing E_0 for E, E_{d-1} for E^σ, σ_{d-1} for the isogeny between them and so on we can draw the graph:

$$E_0 \xrightarrow{\sigma_{d-1}} E_{d-1} \xrightarrow{\sigma_{d-2}} \cdots \xrightarrow{\sigma_1} E_1 \xrightarrow{\sigma_0} E_0$$

Composing all these gives:

Definition 3.1 *The q-th power* Frobenius *on E is the curve endomorphism $F = \sigma_0 \circ \sigma_1 \circ \ldots \circ \sigma_{d-1}$.*

Of course the action of F is trivial over $\mathbb{F}_q$ but it is non-trivial in extension fields.

In the same manner, if we have a curve $\mathcal{E}$ over $\mathbb{Z}_q$, the little Frobenius Σ on $\mathbb{Z}_q$ can be extended to a map between $\mathcal{E}$ and the conjugate curve $\mathcal{E}^{\Sigma}$. Repeating this operation d times, brings back to the initial curve, thus yielding another cycle of isogenies.

3.2 Canonical lifting of curves

First, we will recall some basic facts about modular equations. The modular equation $\Phi_p(X,Y)$ is a symmetric polynomial of degree $p+1$ in each variable, with integer coefficients with the following property. Two elliptic curves E and E' defined over $\mathbb{F}_q$, are related via a cyclic isogeny of degree ℓ only if $\Phi_\ell(j(E), j(E')) = 0$.

It satisfies the so-called *Kronecker relation*:

$$\Phi_p(X,Y) \equiv (X^p - Y)(X - Y^p) \mod p.$$

The coefficients become rather large as p increases, so we just recall the modular equations Φ_2 and Φ_3:

$$\Phi_2(X,Y) = X^3 + Y^3 - X^2Y^2 + c_1(XY^2 + X^2Y) - c_2(X^2 + Y^2) + c_3XY + c_4(X+Y) - c_5$$

where: $c_1 = 1488,\ \ c_2 = 162000,\ \ c_3 = 40773375,\ \ c_4 = 8748000000,\ \ c_5 = 157464000000000$

$$\Phi_3(X,Y) = X^4 + Y^4 - X^3Y^3 + d_1(X^3Y^2 + X^2Y^3) - d_2(X^3Y + XY^3) + d_3(X^3 + Y^3)$$
$$+d_4X^2Y^2 + d_5(X^2Y + XY^2) + d_6(X^2 + Y^2) - d_7XY + d_8(X+Y)$$

where $d_1 = 2232$, $d_2 = 1069956$, $d_3 = 36864000$, $d_4 = 2587918086$, $d_5 = 8900222976000$, $d_6 = 452984832000000$, $d_7 = 770845966336000000$ and $d_8 = 1855425871872000000000$.

The following theorem, due to Lubin, Serre and Tate [LST64], allows one to lift a curve E over $\mathbb{F}_q$ to a curve $\mathcal{E}$ over $\mathbb{Z}_q$ canonically, by lifting its j–invariant. The curve $\mathcal{E}$ is the unique lift of E having the same endomorphism ring.

Theorem 3.1 (Lubin-Serre-Tate) *Let E be a curve over $\mathbb{F}_q$ with j–invariant $j \in \mathbb{F}_q \setminus \mathbb{F}_{p^2}$, then there is a unique J in $\mathbb{Z}_q$ such that*

$$\Phi_p(J, \Sigma(J)) = 0 \text{ and } J \equiv j \mod p,$$

and J is the j–invariant of the canonical lift *$\mathcal{E}$ of E (defined up to isomorphism).*

It is possible to implement this theorem directly by lifting J with its conjugate $\Sigma(J)$. This is done by computing the Witt coefficients x_n for $n > 0$. When the x_i are known for $0 \le i < n$, the equation $\Phi_p(J, \Sigma(J))/p^n \equiv 0 \mod p$ yields a p–linear polynomial in the variable x_n which can easily be solved. Unfortunately, computing the polynomial itself requires computations of Σ and this is slow in practice. Furthermore, $\Theta(d)$ such steps are required.

A much faster method is developed in the next sections using the whole cycle of J_i's. Indeed writing out all the equations $\Phi_p(J_i, J_{i+1})$ for $0 \le i < d$ produces an algebraic system over $\mathbb{Z}_q$ without Σ, which can be solved quickly by a $O(\log d)$ steps of a multi-variate Newton iteration.

The cycle of lifted curves is:

$$
\begin{tikzcd}
\mathcal{E}_0 \arrow[r, "\Sigma_{d-1}"] \arrow[d, "\pi"] & \mathcal{E}_{d-1} \arrow[r, "\Sigma_{d-2}"] \arrow[d, "\pi"] & \cdots \arrow[r, "\Sigma_1"] & \mathcal{E}_1 \arrow[r, "\Sigma_0"] \arrow[d, "\pi"] & \mathcal{E}_0 \\
E_0 \arrow[r, "\sigma_{d-1}"] & E_{d-1} \arrow[r, "\sigma_{d-2}"] & \cdots \arrow[r, "\sigma_1"] & E_1 \arrow[r, "\sigma_0"] & E_0
\end{tikzcd}
$$

Considering it as a whole yields the curve endomorphism $\mathcal{F}$ which is the lift of F:

$$
\begin{tikzcd}
\mathcal{E} \arrow[r, "\mathcal{F}"] \arrow[d, "\pi"] & \mathcal{E} \arrow[d, "\pi"] \\
E \arrow[r, "F"] & E
\end{tikzcd}
$$

Now the dual of a little Frobenius isogeny σ is a (separable) isogeny $\hat{\sigma}$ called the *Verschiebung*. Following Satoh we will work with the dual since, being separable, it can be manipulated easily via its kernel. By considering the duals, we go around the same cycles in the opposite direction:

$$
\begin{tikzcd}
\mathcal{E}_0 \arrow[r, "\hat{\Sigma}_0"] \arrow[d, "\pi"] & \mathcal{E}_1 \arrow[r, "\hat{\Sigma}_1"] \arrow[d, "\pi"] & \cdots \arrow[r, "\hat{\Sigma}_{d-2}"] & \mathcal{E}_{d-1} \arrow[r, "\hat{\Sigma}_{d-1}"] \arrow[d, "\pi"] & \mathcal{E}_0 \\
E_0 \arrow[r, "\hat{\sigma}_0"] & E_1 \arrow[r, "\hat{\sigma}_1"] & \cdots \arrow[r, "\hat{\sigma}_{d-2}"] & E_{d-1} \arrow[r, "\hat{\sigma}_{d-1}"] & E_0
\end{tikzcd}
$$

We also get the dual curve endomorphism $\hat{F} = \hat{\sigma}_{d-1} \circ \hat{\sigma}_{d-2} \circ \cdots \circ \hat{\sigma}_0$ and its lift $\hat{\mathcal{F}}$.

$$
\begin{tikzcd}
\mathcal{E} \arrow[r, "\hat{\mathcal{F}}"] \arrow[d, "\pi"] & \mathcal{E} \arrow[d, "\pi"] \\
E \arrow[r, "\hat{F}"] & E
\end{tikzcd}
$$

3.3 The main algorithm

The algorithm consists of two phases: lifting all the information we need to $\mathbb{Z}_q$ with enough precision, and then recovering the trace with the help of these data.

Following Satoh we construct a cycle of j–invariants, using the Frobenius σ in $\mathbb{F}_q$, and then lift all of them simultaneously using a multivariate Newton iteration. In this manner we obtain all the conjugate J's in $\mathbb{Z}_q$ without having to compute the Frobenius Σ in $\mathbb{Z}_q$ at all. Once that is done, we lift the curves one at a time by lifting their coefficients using (univariate) Newton iterations.

The isogenies $\hat{\sigma}_i$ and $\hat{\Sigma}_i$ are separable of degree p, so they are determined by their kernels which have order p. Therefore lifting $\hat{\sigma}_i$ to $\hat{\Sigma}_i$ will be done by lifting its kernel, which is a p–torsion subgroup.

We can lift the kernel either by lifting a single point using a Newton iteration, when $p = 2$ and $p = 3$, or by lifting a factor of the p–division polynomial with a more general Hensel lift, when $p \geq 5$.

Procedure MAINALGORITHM
Input: An elliptic curve E defined over $\mathbb{F}_q$, with $j(E) \notin \mathbb{F}_{p^2}$.
Output: The trace of the curve E.

1. Compute the cycle of d curves E_i and their j–invariants j_i;

2. Lift all the j_i simultaneously, yielding J_i;

3. Lift each curve by lifting its coefficients;
4. Lift each curve's p–torsion subgroup.
5. Compute the trace from the lifted data.

3.3.1 Computing the trace

The aim here is to compute the trace of the Frobenius endomorphism $F : E \to E$. We assume that we have computed (a good approximation of) the canonical lift $\mathcal{E}$ of E. Since canonical lifting preserves the endomorphism ring, the trace of Frobenius is unchanged. Moreover the trace of an endomorphism is the same as the trace of its dual. We have

$$\mathrm{Tr}\, F = \mathrm{Tr}\, \mathcal{F} = \mathrm{Tr}\, \hat{\mathcal{F}}.$$

where $\mathcal{F}$ is the Frobenius on $\mathcal{E}$ and $\hat{\mathcal{F}}$ is its dual. As explained above, the Frobenius F can be decomposed into the product of little Frobenius isogenies which cycle the curve E through its d conjugates, and similarly for the duals. Thus

$$\mathrm{Tr}\, \hat{\mathcal{F}} = \mathrm{Tr}\, \left(\hat{\Sigma}_{d-1} \circ \hat{\Sigma}_{d-2} \circ \cdots \circ \hat{\Sigma}_0 \right).$$

The next step is to go to the formal groups of the curve $\mathcal{E}$ and its conjugates. Let us denote the local parameter $-X/Y$ of $\mathcal{E}$ by τ, and the local parameters of $\mathcal{E}_i$ by τ_i. Expressing the action of $\hat{\mathcal{F}}$ on this parameter, we get the form

$$\hat{\mathcal{F}}(\tau) = \sum_{k \geq 1} c_k \tau^k,$$

and Satoh's proposition 4.1 shows that the equation $x^2 - \mathrm{Tr}\,(\hat{\mathcal{F}})x + q = 0$ satisfied by $\hat{\mathcal{F}}$ implies

$$\mathrm{Tr}\,(\hat{\mathcal{F}}) = c_1 + \frac{q}{c_1}.$$

Thus, computing the first coefficient c_1 is enough to solve the problem. The key point is that, since $\hat{\mathcal{F}}$ is the composition of morphisms, it is possible to compute c_1 as the product of the first coefficients in the expansions of the factors in the formal groups. More precisely, we can write $\hat{\Sigma}_i : \mathcal{E}_i \to \mathcal{E}_{i+1}$ on the formal groups as

$$\hat{\Sigma}_i(\tau_i) = g_i \tau_i + O(\tau_i^2).$$

Then we have

$$c_1 = \prod_{0 \leq i < d} g_i,$$

and

$$\mathrm{Tr}\, F \equiv \prod_{0 \leq i < d} g_i \mod q.$$

Note that this product of g_i is an expression for the norm from $\mathbb{Z}_q$ down to $\mathbb{Z}_p$ of g_0 and thus it is certainly in $\mathbb{Z}_p$. In addition it gives the trace of F to within $O(p^d)$, since if higher precision was desired it would be necessary to take the term q/c_1 into account.

The final step is to compute each g_i from the curves $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$ and the kernel of $\hat{\Sigma}_i$. This is done with the help of Vélu's formulae [Vél71]. Each g_i can be expressed as a rational fraction in terms of the data we lifted, and multiplying them together gives the trace with sufficient precision.

3.3.2 Complexity

For the field $\mathbb{F}_q$, one works with an algebraic extension of degree d to p–adic precision $O(p^{O(d)})$ so that each element of $\mathbb{Z}_q$ takes $O(d^2 \log p)$ space. For each curve one needs to store a bounded number of such elements, and there are d conjugate curves; thus the total memory space required is $O(d^3 \log p)$. Remarkably the run-time is just $O(d^3 \log d \log\log d)$ with fast arithmetic (or $O(d^5)$ with naïve arithmetic), for fixed p. The algorithm is also significantly easier to implement than an optimized SEA algorithm, and even with simple Karatsuba arithmetic, the run-time exponent is a respectable $1 + 2\frac{\log 3}{\log 2} \simeq 4.17$.

Satoh proved the following theorem for $p \geq 5$, and we extend it to the cases $p = 2$ and $p = 3$.

Theorem 3.2 (Satoh) *Let E be an elliptic curve over the finite field with $q = p^d$ elements. Assume that $j(E) \notin \mathbb{F}_{p^2}$. Then there exists a deterministic algorithm for computing the order of E which, for fixed p, requires $O(d^3)$ memory and $O(d^{3+\varepsilon})$ bit-operations.*

Note that the restriction $j(E) \notin \mathbb{F}_{p^2}$ is not essential as the cases $j(E) \in \mathbb{F}_{p^2}$ can be handled by counting points over a tiny subfield.

4 Algorithms for lifting

4.1 Lifting the cycle of elliptic curves

The theorem of Lubin, Serre and Tate ensures that the cycle of J_i we want to construct is characterized by $\Phi_p(J_i, J_{i+1}) = 0$ for all $0 \leq i < d$ (for notational convenience, the indices are understood to be taken modulo d). The algorithm for computing this cycle from the cycle of j_i is a Newton iteration based on the following function acting on a vector of size d:

$$\Theta(x_0, \ldots, x_{d-1}) = \left(\Phi_p(x_0, x_1), \Phi_p(x_1, x_2), \ldots, \Phi_p(x_{d-1}, x_0)\right).$$

The Newton algorithm presented in section 2 can be adapted to this multivariate case rather easily. Instead of considering the derivative of the function, it is now necessary to deal with the Jacobian matrix

$$D\Theta(x_0, \ldots, x_{d-1}) = \begin{pmatrix} \Phi_p'(x_0, x_1) & \Phi_p'(x_1, x_0) & 0 & \ldots & 0 \\ 0 & \Phi_p'(x_1, x_2) & \Phi_p'(x_2, x_1) & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \Phi_p'(x_{d-1}, x_{d-2}) \\ \Phi_p'(x_0, x_{d-1}) & 0 & 0 & \ldots & \Phi_p'(x_{d-1}, x_0) \end{pmatrix}$$

where $\Phi_p'(X, Y)$ denotes the partial derivative of $\Phi_p(X, Y)$ with respect to X, and the iteration is essentially

$$(x_0, \ldots, x_{d-1}) \leftarrow (x_0, \ldots, x_{d-1}) - \left((D\Theta)^{-1}\Theta\right)(x_0, \ldots, x_{d-1}).$$

We describe here a detailed "ready to implement" algorithm which takes care of two important issues: the choice of working precision at each stage, and the inversion of the Jacobian matrix.

If the computations are organized in a suitable way, the inversion can be done with a linear (in d) number of arithmetic operations. The diagonal elements are all invertible so we can do elimination without having to look for a "pivot". Furthermore the off-diagonal elements are all divisible by p.

By doing row operations on the derivative matrix (and simultaneously on Θ) we move the bottom left element $\Phi_p'(x_0, x_{d-1})$ towards the right. At the n-th step it is divisible by p^n so that it rapidly becomes zero to within working precision. Then the matrix has a simple form that can be solved directly, from bottom to top.

Procedure LIFTJINVARIANTS
Inputs: A desired precision n and a cycle $j_i \in \mathbb{F}_q \setminus \mathbb{F}_{p^2}$ s.t. $\Phi_p(j_i, j_{i+1}) \equiv 0 \mod p$ for all $0 \le i < d$.
Output: A cycle $J_i \in \mathbb{Z}_q$ s.t. $\pi(J_i) = j_i$ and $\Phi_p(J_i, J_{i+1}) \equiv 0 \mod p^n$ for all $0 \le i < d$.

1. IF $n = 1$ THEN pick arbitrary J_i s.t. $\pi(J_i) = j_i$; GO TO 5;
2. $n' \leftarrow \lceil \frac{n}{2} \rceil$;
3. $(J_0, J_1, \ldots, J_{d-1}) \leftarrow$ LIFTJINVARIANTS$(n', (j_0, j_1, \ldots, j_{d-1}))$;
4. $(J_0, J_1, \ldots, J_{d-1}) \leftarrow$ UPDATEJS$(n, (J_0, J_1, \ldots, J_{d-1}))$;
5. RETURN $(J_0, J_1, \ldots, J_{d-1})$.

Procedure UPDATEJS
Inputs: A desired precision n and a cycle $J_i \in \mathbb{Z}_q$ s.t. $\Phi_p(J_i, J_{i+1}) \equiv 0 \mod p^{\lceil n/2 \rceil}$ for all $0 \le i < d$.
Output: A cycle $\mathfrak{J}_i \in \mathbb{Z}_q$ s.t. $\Phi_p(\mathfrak{J}_i, \mathfrak{J}_{i+1}) \equiv 0 \mod p^n$ and $\mathfrak{J}_i \equiv J_i \mod p^{\lceil n/2 \rceil}$ for all $0 \le i < d$.

1. Allocate arrays $D[0..d-2]$, $P[0..d-1]$ and $\mathfrak{J}[0..d-1]$ of elements of $\mathbb{Z}_q$;
2. FOR $i = 0$ TO $d-2$ DO

   $t \leftarrow 1/\Phi'(J_i, J_{i+1})$;
   $D_i \leftarrow t \cdot \Phi'(J_{i+1}, J_i)$;
   $P_i \leftarrow t \cdot \Phi(J_i, J_{i+1})$;

3. $m \leftarrow \Phi'(J_0, J_{d-1})$;
   $f \leftarrow \Phi(J_{d-1}, J_0)$;
4. FOR $i = 0$ TO $d-2$ DO

   $f \leftarrow f - m \cdot P_i$;
   $m \leftarrow -m \cdot D_i$;
   IF $m \equiv 0 \mod p^n$ THEN BREAK;

5. $m \leftarrow m + \Phi'(J_{d-1}, J_0)$;
6. $P_{d-1} \leftarrow f/m$;
7. FOR $i = d-2$ DOWN TO 0 DO

   $P_i \leftarrow P_i - D_i \cdot P_{i+1}$;

8. FOR $i = 0$ TO $d-1$ DO

   $\mathfrak{J}_i \leftarrow J_i - P_i$;

9. Free arrays D and P;

10. RETURN $(\mathfrak{J}_0, \mathfrak{J}_1, \ldots, \mathfrak{J}_{d-1})$.

In the procedure UPDATEJS, it is possible to reduce the memory usage significantly. Indeed, the D and the P arrays can be avoided altogether by computing their entries *on the fly*, although some of the calculations need to be repeated. To do this, loop 2 of the algorithm should be removed and the values D_i and P_i should be computed on the fly in loop 4. Loops 7 and 8 need to be merged and the values D_i and P_i computed on the fly. Some of them were already computed in loop 4 but note that the $m \equiv 0$ test breaks out of that loop after roughly n iterations. In fact, by working to half precision, as described at the end of section 2.5, roughly $n/2$ iterations are sufficient. Since $n \approx d/2$ at most, not much recomputation is necessary.

4.2 Characteristic $p \geq 5$ (Satoh)

4.2.1 Lifting the curve's equation

When the characteristic p is at least 5 ($p = 2$ is treated in section 4.3 and $p = 3$ in appendix 7), we consider that the elliptic curve E over $\mathbb{F}_q$ has an equation of the form $E : y^2 = x^3 + ax + b$. In this case, we want to lift E to a curve $\mathcal{E} : y^2 = x^3 + Ax + B$ over $\mathbb{Z}_q$ whose j–invariant we already know. To do so, we lift one of the coefficients arbitrarily and lift the other with a Newton iteration based on the equation $J = -1728(4A)^3/\Delta$ where $\Delta = -16(4A^3 + 27B^2)$. Note that the hypotheses $p \geq 5$ and $J \notin \mathbb{F}_{p^2}$ ensure that the denominator in the iteration is non zero, thus the precision exactly doubles at each iteration.

Procedure LIFTAANDB
Inputs: The precision n, a j–invariant $J \in \mathbb{Z}_q$ and coefficients a and b of a curve E over $\mathbb{F}_q$ with j–invariant $\pi(J)$.
Output: Coefficients A and B in $\mathbb{Z}_q$ of a curve $\mathcal{E}$ with j–invariant J, lifted to precision n.

1. Pick arbitrary A such that $\pi(A) = a$;

2. $B \leftarrow$LIFTBGIVENA(n, J, A, b);

3. RETURN A, B.

Procedure LIFTBGIVENA
Inputs: The precision n, a j–invariant J and coefficients $A \in \mathbb{Z}_q$ and $b \in \mathbb{F}_q$.
Output: A coefficient B in $\mathbb{Z}_q$, lifted to precision n.

1. IF $n = 1$ THEN pick any B such that $\pi(B) = b$; GO TO 5;

2. $n' \leftarrow \lceil \frac{n}{2} \rceil$;

3. $B \leftarrow$ LIFTBGIVENA(n', J, A, b);

4. $B \leftarrow B - \frac{J(4A^3+27B^2)-6912A^3}{54\,J\,B}$;

5. RETURN B.

4.2.2 Lifting the p–torsion

Lifting the p–torsion subgroup is the more delicate step in Satoh's algorithm. The idea is to lift a factor of the p–division polynomial $\Psi_p(x)$ of E_i by a type of quadratic Hensel lift (which resembles a Newton iteration). Two problems arise:

- Lifting a factor is not as simple as lifting a root of a polynomial,
- The derivative of $\Psi_p(x)$ is zero modulo p, so it is necessary to be careful with precision. It is especially important to check the initial required precision.

The algorithm for dealing with this factor lifting is given in Lemma 2.1 of Satoh's paper. It is very technical and we shall not reproduce the proof here. We only reformulate the result with our notation.

Theorem 4.1 (Satoh) *Let $\Psi(x)$ be a polynomial over a p–adic field such that $p \parallel \Psi'(x)$. Let $h(x)$ be a monic polynomial known to precision n such that*

- *$\pi(h(x))$ is square-free and prime to $\pi(p^{-1}\Psi'(x))$,*
- *$h(x)$ divides $\Psi(x)$ to precision $n+1$.*

Define a new polynomial

$$\mathfrak{h}(x) = h(x) + \left(\frac{\Psi(x)}{\Psi'(x)} h'(x) \mod h(x) \right).$$

Then $\mathfrak{h}(x)$ has the following properties

- *$h(x) \equiv \mathfrak{h}(x)$ modulo p^n,*
- *$h(x)$ divides $\Psi(x)$ to precision $2n+1$.*

Note moreover that if $n \geq 1$, then $\pi(h(x)) = \pi(\mathfrak{h}(x))$, and the process can be repeated.

To turn this theorem into an algorithm, it remains to initialize the iteration. For this, we need a way to construct the first approximation for the factor $H(x)$ of $\Psi_p(x)$. Satoh proposes taking the p-th root of the p–division polynomial of the curve E_i, which is free since all the necessary information is in the p–division polynomial of the curve E_{i+1}.

The algorithm is as follows:

Procedure LIFTH
Inputs: The precision n, an elliptic curve E_{i+1} over $\mathbb{F}_q$ and an elliptic curve $\mathcal{E}_i$ over $\mathbb{Z}_q$.
Output: A monic factor $H(x)$ of the p-th division polynomial of $\mathcal{E}_i$, representing the kernel of $\hat{\Sigma}_i$.

1. $\Psi(x) \leftarrow$ the p-th division polynomial of $\mathcal{E}_i$; *(Note* deg $\Psi = \frac{p^2-1}{2}$*).*
2. $\psi(x) \leftarrow$ the p-th division polynomial of E_{i+1}; *(Note $\psi = h^p$ for some $h(x)$ with degree $\frac{p-1}{2}$).*
3. $h(x) \leftarrow 0$;
4. FOR $k = 0$ TO $\frac{p-1}{2}$ DO

$h(x) \leftarrow h(x) + \frac{u_k}{u_{(p-1)/2}} x^k$, with u_k the coefficient of x^{pk} in $\psi(x)$;

5. $H(x) \leftarrow$ LIFTHBIS$(n, \Psi(x), h(x))$;

6. RETURN $H(x)$.

Procedure LIFTHBIS
Inputs: The precision n, a polynomial $\Psi(x)$ and an approximation $h(x)$.
Output: A monic factor $H(x)$ of $\Psi(x)$ to precision n.

1. IF $n = 1$ THEN $H(x) \leftarrow h(x)$; GO TO 5;

2. $n' \leftarrow \lceil \frac{n-1}{2} \rceil$;

3. $H(x) \leftarrow$ LIFTHBIS$(n', \Psi(x), h(x))$;

4. $H(x) \leftarrow H(x) + \left(\frac{\Psi(x)}{\Psi'(x)} H'(x) \mod H(x)\right)$;

5. RETURN H(x).

4.3 Characteristic two

In this subsection, we propose lifting algorithms for the curve and the torsion subgroup in characteristic two. In remark 3.9 of [Sat00], Satoh pinpointed the main obstacle to extending his method to $p = 2$ and sketched a possible workaround using a probabilistic factoring algorithm. Here we propose an efficient and completely deterministic solution.

In characteristic two any elliptic curve E, or its twist, has the form

$$y^2 + xy = x^3 + a.$$

provided $j(E) \neq 0$. Thus we can confine ourselves to this case.

4.3.1 Lifting the curve's equation

Our algorithm for lifting the equation of the curve is to lift its a coefficient using a Newton iteration, to obtain the coefficient A in the equation $y^2 + xy = x^3 + A$ of the lifted curve $\mathcal{E}$. Since $J = \frac{1}{\Delta}$, where $\Delta = -A - 432A^2$ is the discriminant of the curve, the polynomial whose root we will calculate is $f(x) = 1 + J(x + 432x^2)$.

In this case $f'(x) = J(1 + 864x)$ and since $\pi(J) \neq 0$, we could use procedure NEWTONITERATIONS (see section 2.5 above) with $k = 0$. Thus we could go from precision n to $2n$ at each step, but in fact we can do slightly better than the generic method and instead go to $2n + 4$. Let x be an approximate root at precision n, with error term $f(x) = O(2^n)$. Let $y = x - f(x)/f'(x)$ be the improved root after one step. Then the new error term can be written explicitly as

$$\begin{aligned} f(y) &= 1 + J(y + 432y^2) \\ &= 432\, J\, \frac{f(x)^2}{f'(x)^2} \end{aligned}$$

We have $2 \nmid f'(x)$ and $2^4 \parallel 432$, therefore $f(y) = O(2^{2n+4})$. It follows that an initial approximate root is not actually needed to start the iteration, although for efficiency we might as well use $-1/J$ modulo 16.

Procedure LIFTA
Inputs: An integer n and the j–invariant J of a curve $\mathcal{E}$ in $\mathbb{Z}_q$, with precision n.
Output: The coefficient A of the lifted curve $\mathcal{E}$ in $\mathbb{Z}_q$ with precision n.

1. IF $n \leq 4$ THEN $A \leftarrow -1/J$; GO TO 5;
2. $n' \leftarrow \lceil \frac{n-4}{2} \rceil$;
3. $A \leftarrow \text{LIFTA}(n', J)$;
4. $A \leftarrow A - \frac{1+J(A+432A^2)}{J(1+864A)}$;
5. RETURN A.

(A possible alternative algorithm would be to lift the conjugates of the coefficient a directly using a modular polynomial for a instead of the usual one for j).

4.3.2 Lifting the 2–torsion

Our algorithm for lifting the 2–torsion is also based on Newton iteration. In this case, the function $f(x)$ comes from the equation satisfied by the abscissae of the non-trivial 2–torsion points on the lifted curve $\mathcal{E}$, namely the 2–division polynomial $4X^3 + X^2 + 4A$. A root $X \in \mathbb{Z}_q$ of this polynomial is necessarily 0 modulo 2. By setting $X = 2Z$, we get the modified division polynomial $8Z^3+Z^2+A$. In this case we find Z using an iteration based on the function $f(x) = 8x^3 + x^2 + A$.

A difficulty arises in that there are two candidate roots in $\mathbb{Z}_q$ and we must compute the one which corresponds to the non-trivial point in the kernel of $\hat{\Sigma}_i$. It will be sufficient to initialise the Newton iteration with the correct square root of $-A$ modulo 8.

We next describe how to do this in a deterministic way. We thus solve the problem Satoh encounters in his paper while avoiding the probabilistic polynomial factorization modulo 8 that he suggests.

Some theory is needed to justify the choice of root but the result is simple and efficient. The reader can skip the next part if desired.

Choosing the correct root. In section 5.3 below, we use Vélu's formulae to construct a curve $\mathcal{E}'_i$ whose j–invariant must equal J_{i+1}. Computing it explicitly, in terms of A and the as-yet-unknown Z, from the equation of $\mathcal{E}'_i$ yields

$$J_{i+1} = J(\mathcal{E}'_i) \equiv \frac{-1}{Z^2 - Z + A} \mod 4.$$

Since Z is a root of f, we have $Z^2 + A \equiv 0 \mod 4$ (even modulo 8) and thus

$$Z \equiv \frac{1}{J_{i+1}} \mod 4$$

which determines Z uniquely and provides a sufficiently precise initial root Z_0 for the procedure NEWTONITERATIONS. Indeed $f'(x) = 2(12x^2 + x)$ and $12Z_0^2 + Z_0 \equiv 1/J_{i+1} \mod 4$ and this is non-zero modulo 2. Hence we have $k = 1$ and we need $f(Z_0) \equiv 0 \mod 2^{2k+1}$ i.e., modulo 8 to start the iterations. Now

$$\begin{aligned} f(Z_0) &\equiv Z_0^2 + A \mod 8, \\ &\equiv (1/J_{i+1}^2) + A, && \textit{(the value of } Z_0 \textit{ mod 4 determines } Z_0^2 \textit{ mod 8)} \\ &\equiv (1/J_{i+1}^2) - (1/J_i). && \textit{(from lifting of } A \textit{ in section 4.3.1 above)} \end{aligned}$$

It remains to show $J_{i+1}^2 \equiv J_i \bmod 8$. Now $J_i = \Sigma(J_{i+1})$ so the desired result certainly holds modulo 2. Next, note that the Kronecker relation $\Phi_2(X,Y) \equiv (X^2-Y)(X-Y^2)$ actually holds modulo 16 and not just modulo 2. Thus

$$\Phi_2\left(\omega(j_i), \omega(j_{i+1})\right) \equiv 0 \mod 16$$

and the terms x_1, x_2 and x_3 in the semi-Witt decomposition of J_{i+1} are zero (see section 2.3). Thus $J_i \equiv J_{i+1}^2 \bmod 16$ and hence certainly modulo 8 as required[2].

Lifting the root. To calculate the correct root Z, we use procedure NEWTONITERATIONS with $k=1$ and with initial root $1/J_{i+1} \bmod 4$.

Here, the precision increases at each step from n to $2n-1$. Let x be the approximate value at precision n, with error term $f(x) = O(2^{n+1})$. Let $y = x - f(x)/f'(x)$ be the improved root after one step. Then the new error term is

$$\begin{aligned} f(y) &= 8y^3 + y^2 + A \\ &= \frac{f(x)^2}{f'(x)^2}(24x+1) - 8\frac{f(x)^3}{f'(x)^3}. \end{aligned}$$

We have $2 \parallel f'(x)$ and therefore $f(y) = O(2^{\min(2n,3n+3)}) = O(2^{2n})$ as expected.

Procedure LIFTZ
Inputs: An integer n, the j–invariant J_{i+1} to precision 2 and the coefficient A to precision $n+1$.
Output: The Z of the lifted curve $\mathcal{E}$ with precision n.

1. IF $n \leq 2$ THEN $Z \leftarrow 1/J_{i+1}$; GO TO 5;
2. $n' \leftarrow \lceil \frac{n+1}{2} \rceil$;
3. $Z \leftarrow$ LIFTZ(n', J_{i+1}, A);
4. $Z \leftarrow Z - \frac{8Z^3+Z^2+A}{2(12Z^2+Z)}$;
5. RETURN Z.

5 Computing the trace

Vélu's formulae lead to simple expressions for the first coefficient of the isogeny $\hat{\Sigma}$ between $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$. Computing the trace just requires taking the product of all these expressions. In this section we recall the formula given by Satoh for $p \geq 5$ and derive a similar one for characteristic two. The case $p=3$ is given in the appendix.

5.1 Vélu's formulae

Let E be an elliptic curve, and let F be a finite subgroup of E. Then E/F is isomorphic to an elliptic curve. The formulae given by Vélu [Vél71] give the *explicit* equation of an elliptic curve E' isomorphic to E/F, together with an *explicit* formula for the isogeny between E and E'. Moreover the isogeny is such that when we write it in the formal group, its first coefficient is 1.

[2]Note that this allows the first two steps of lifting the J_i's to be optimised away and replaced by $d+2$ squarings.

Let us apply this to our problem. We have two curves $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$ and a subgroup of $\mathcal{E}_i$. We want to know the first coefficient in the formal group expression of the isogeny between $\mathcal{E}_i$ and $\mathcal{E}_{i+1}$ corresponding to this kernel. The first step is to compute the curve $\mathcal{E}_i'$ given by Vélu's formulae. Then, we know that $\mathcal{E}_i'$ and $\mathcal{E}_{i+1}$ are isomorphic so we can compute this isomorphism λ explicitly and its first coefficient in the formal group expansion.

We have the following diagram:

$$\begin{tikzcd}
\mathcal{E}_i \arrow[rr, "\hat{\Sigma}_i"] \arrow[dr] & & \mathcal{E}_{i+1} \\
& \mathcal{E}_i' \arrow[ur, "\lambda"] &
\end{tikzcd}$$

Thus g_i, the first coefficient of the isogeny $\hat{\Sigma}_i$ is equal to the product of the first coefficient of Vélu's isogeny (which is 1) and that of the isomorphism λ. For the latter, it is easy to find its *square* in terms of the equations of the curves.

5.2 Characteristic $p \geq 5$ (Satoh)

In the case of characteristic $p \geq 5$, we refer to Satoh's original paper for the derivation of the formulae. (For characteristic two, see 5.3 and for characteristic three, see 7.3 in the appendix).

The equation of $\mathcal{E}_i$ is of the form

$$y^2 = x^3 + A_i x + B_i,$$

and that of $\mathcal{E}_{i+1}$ is

$$y^2 = x^3 + A_{i+1} x + B_{i+1}.$$

The kernel of $\hat{\Sigma}_i$ consists of the point at infinity on $\mathcal{E}_i$ and $p-1$ finite points whose $\frac{p-1}{2}$ distinct abscissae are given by the roots of the polynomial $H_i(x)$.

The curve $\mathcal{E}_i'$ given by Vélu's formulae has equation $y^2 = x^3 + \alpha_i x + \beta_i$ with

$$\begin{aligned}
\alpha_i &= (6-5p)A_i - 30(s_1^2 - 2s_2) \quad \text{and} \\
\beta_i &= (15-14p)B_i - 70(-s_1^3 + 3s_1 s_2 - 3s_3) + 42A_i s_1,
\end{aligned}$$

where s_k denotes the coefficient of $x^{(p-1)/2-k}$ in $H_i(x)$.

Having the equation for $\mathcal{E}_i'$, it is rather easy to compute the isomorphism with $\mathcal{E}_{i+1}$. We refer to Silverman [Sil86] for the general form of an isomorphism between elliptic curves. In this case it has the form $(x, y) \mapsto (g_i^2\, x,\, g_i^3\, y)$ where g_i is the term we are looking for. Hence

$$g_i^2 = \frac{\beta_i}{\alpha_i} \frac{A_{i+1}}{B_{i+1}}.$$

Thus the algorithm for computing the square of the trace is now clear, and we decide between the two square roots by checking the sign using the Hasse invariant h_E defined to be the norm of the coefficient of x^{p-1} in the polynomial $(x^3 + ax + b)^{(p-1)/2}$.

Procedure ComputeTraceOddChar
Input: An elliptic curve E over $\mathbb{F}_{p^d}$, with $j(E) \notin \mathbb{F}_{p^2}$, given by its equation $y^2 = x^3 + ax + b$.
Output: The trace of the curve.

1. $n \leftarrow \lceil \log_p 4 + \frac{d}{2} \rceil$;

2. $(j_i)_{0 \le i < d}$, $(a_i)_{0 \le i < d}$, $(b_i)_{0 \le i < d} \leftarrow$ Conjugates of $j(E)$, of a, of b;

3. $(J_i)_{0 \le i < d} \leftarrow \textsc{LiftJInvariants}(n, (j_i))$;

4. $N \leftarrow 1$; $D \leftarrow 1$;

5. FOR $i = 0$ TO $d - 1$ DO

   $A, B \leftarrow \textsc{LiftAAndB}(n, J_i, a, b)$;
   $H \leftarrow \textsc{LiftH}(n, a_{i+1}, b_{i+1}, A, B)$;
   $s_1, s_2, s_3 \leftarrow$ coefficients of $x^{(p-1)/2-1}$, $x^{(p-1)/2-2}$ and $x^{(p-1)/2-3}$ in $H(x)$;
   $\alpha \leftarrow (6 - 5p)A - 30(s_1^2 - 2s_2)$;
   $\beta \leftarrow (15 - 14p)B - 70(-s_1^3 + 3s_1s_2 - 3s_3) + 42As_1$;
   $N \leftarrow N\beta A$;
   $D \leftarrow D\alpha B$;

6. $c \leftarrow \sqrt{N/D}$;

7. $h_E \leftarrow$ Hasse invariant of E.

8. IF $c \not\equiv h_E \mod p$ THEN $c \leftarrow -c$;

9. Reduce c to an integer in $0 \ldots p^n$. IF $c > 2\sqrt{q}$ THEN $c \leftarrow c - p^n$;

10. RETURN c.

Note that in step 6, the numerator and denominator are in $\mathbb{Z}_p$ and c is too. The p–adic square root can be found using a Newton iteration for the inverse square root, and then multiplying by c. There are two possible roots, either of which will do, since step 8 corrects the sign if necessary.

5.3 Characteristic two

The equation of $\mathcal{E}_i$ is of the form

$$y^2 + xy = x^3 + A_i,$$

and that of $\mathcal{E}_{i+1}$ is

$$y^2 + xy = x^3 + A_{i+1}.$$

In the case of characteristic two, the kernel of $\hat{\Sigma}_i$ is a subgroup of order 2 of the 2–torsion. Let (X_i, Y_i) be the non-trivial point of this subgroup. Note that the ordinate satisfies $Y_i = -X_i/2$.

The curve $\mathcal{E}'_i$ given by Vélu's formulae [Vél71] has equation $y^2 + xy = x^3 + \mathcal{A}_4 x + \mathcal{A}_6$ where

$$\mathcal{A}_4 = -5t \quad \text{and} \quad \mathcal{A}_6 = A_i - t - 7w,$$

with

$$t = 3X_i^2 - Y_i \quad \text{and} \quad w = X_i\, t.$$

Here again, having the equation for $\mathcal{E}_i'$ and $\mathcal{E}_{i+1}$ allows us to compute the isomorphism and then the coefficient g_i^2. After translations of the axes (which do not affect the first coefficient in the formal group expansion of the isomorphism), the equations are the following:

$$\begin{aligned}\mathcal{E}_{i+1} &: y^2 = x^3 - \tfrac{1}{48}x + \tfrac{1}{864} + A_{i+1},\\ \mathcal{E}_i' &: y^2 = x^3 + \left(\mathcal{A}_4 - \tfrac{1}{48}\right)x + \tfrac{1}{864} + \mathcal{A}_6 - \tfrac{\mathcal{A}_4}{12}.\end{aligned}$$

Hence

$$g_i^2 = \frac{-\frac{1}{48}}{\frac{1}{864} + A_{i+1}} \cdot \frac{\frac{1}{864} + \mathcal{A}_6 - \frac{\mathcal{A}_4}{12}}{\mathcal{A}_4 - \frac{1}{48}},$$

which simplifies to

$$g_i^2 = \frac{72\mathcal{A}_4 - 1 - 864\mathcal{A}_6}{(48\mathcal{A}_4 - 1)(1 + 864A_{i+1})}.$$

Thereafter we can replace $\mathcal{A}_4$ and $\mathcal{A}_6$ with their expressions in terms of A_i and X_i.

We obtain

$$g_i^2 = \frac{-18144X_i^3 - 4536X_i^2 - 252X_i + 1 + 864A_i}{(1 + 120X_i + 720X_i^2)(1 + 864A_{i+1})}.$$

We can simplify this formula further by reducing modulo the minimal polynomial of X_i, which is $4X_i^3 + X_i^2 + 4A_i$. We get

$$g_i^2 = \frac{1 - 252X_i + 19008A_i}{\left(1 + 120(X_i + 6X_i^2)\right)(1 + 864A_{i+1})}.$$

Remembering that we lifted $Z_i = X_i/2$ instead of X_i and considering that we need one extra bit of precision for a 2–adic square root, we get the following algorithm.

Procedure ComputeTraceChar2
Input: An elliptic curve E over $\mathbb{F}_{2^d}$, with $j(E) \notin \mathbb{F}_4$, given by its its equation $y^2 + xy = x^3 + a$.
Output: The trace of the curve.

1. $j_0 \leftarrow j(E)$; FOR $i = d-1$ DOWN TO 1 DO $j_i \leftarrow j_{i+1}^2$;
2. $n \leftarrow \lceil \frac{d}{2} \rceil + 1$;
3. $(J_i)_{0 \le i < d} \leftarrow$ LiftJInvariants$(n, (j_i))$;
4. $N \leftarrow 1$; $D \leftarrow 1$;
5. FOR $i = 0$ TO $d-1$ DO

   $A \leftarrow$ LiftA(n, J_i);
   $Z \leftarrow$ LiftZ$(n-1, J_{i+1}, A)$;
   $A \leftarrow 864A$;
   $N \leftarrow N(1 - 504Z + 22A)$; *($N$ has precision $n+2$)*
   $D \leftarrow D(1 + 240(Z + 12Z^2))(1 + A)$; *($D$ has precision $n+2$)*

6. $c \leftarrow \sqrt{N/D}$;
7. IF $c \not\equiv 1 \mod 4$ THEN $c \leftarrow -c$;

8. Reduce c to an integer in $0 \dots 2^{n+1}$. IF $c > 2\sqrt{q}$ THEN $c \leftarrow c - 2^{n+1}$;

9. RETURN c.

Note that in step 6, the numerator and denominator are in $\mathbb{Z}_2$ and c is too. The 2–adic square root can be found via a Newton iteration for the inverse square root. There are four possible square roots modulo 2^{n+2} but just two modulo 2^{n+1} and step 7 corrects the sign if necessary.

It is possible to reduce the amount of memory required by combining the lift of the J_i's with the computations in loop 5. Indeed as soon as one of the J_i's has been computed to full precision n in procedure UPDATEJS, it can be used to lift the corresponding A and Z and to update N and D. Then J_i is no longer needed. In this way, the cycle of J_i's only needs to be stored to half precision. When this trick is combined with the one described at the end of section 4.1, the total space requirement is

$$\frac{d^3}{4} + O(d^2) \quad \text{bits.}$$

6 Numerical examples

6.1 A small example

In this section we give some intermediate data which are computed when our algorithm for characteristic two is run on the elliptic curve over $\mathbb{F}_{2^{11}} = \mathbb{F}_2[t]/(t^{11} + t^2 + 1)$ defined by $y^2 + xy = x^3 + a$ with $a = t^4 + t^2 + t$. This example is taken from [LM00] where it is used to compare different algorithms for computing isogenies.

The first steps are to lift the cycle of j–invariants and the curves to precision $O(2^7)$. We give here the resulting sequence of A_i's.

$$\begin{array}{rcl}
A_0 & = & 30t^{10} + 12t^9 + 30t^8 - 26t^7 - 18t^6 - 12t^5 - 39t^4 - 2t^3 + 41t^2 + 7t - 20 \\
A_1 & = & -31t^{10} - 55t^9 - 3t^8 + 43t^7 + 33t^6 + 10t^5 - 50t^4 - 46t^3 + 17t^2 + 20t - 5 \\
A_2 & = & -46t^{10} - 53t^9 - 6t^8 + 6t^7 - 18t^6 - 33t^5 + 17t^4 + 31t^3 + 16t^2 - 39t - 15 \\
A_3 & = & -8t^{10} - 19t^9 + 47t^8 - 50t^7 - 41t^6 - 14t^5 + 30t^4 + 46t^3 + 9t^2 - 48t - 57 \\
A_4 & = & 23t^{10} - 58t^9 - 42t^8 - 26t^7 + 18t^6 + 40t^5 + 29t^4 + 29t^3 - 30t^2 - 38t - 54 \\
A_5 & = & 39t^{10} - 43t^9 + 33t^8 + 43t^7 + 30t^6 - 39t^5 - 56t^4 + 34t^3 - t^2 - 27t + 63 \\
A_6 & = & 42t^{10} + 31t^9 - 38t^8 + 49t^7 - 49t^6 - 55t^5 - 21t^4 - 42t^3 - 42t^2 + 63t - 51 \\
A_7 & = & -62t^{10} - 53t^9 + 46t^8 - 5t^7 - 37t^6 - 26t^5 + 53t^3 + 35t^2 + 52t - 15 \\
A_8 & = & 15t^{10} + 54t^9 + 35t^8 - 31t^7 - 58t^6 - 10t^5 + 60t^4 + 49t^3 + 16t^2 + 40t + 26 \\
A_9 & = & -58t^{10} + 12t^9 - 45t^8 - 15t^7 - 38t^6 - 63t^5 + 11t^4 + 18t^3 - 8t^2 + 30t - 20 \\
A_{10} & = & 56t^{10} + 44t^9 - 57t^8 + 12t^7 + 50t^6 - 54t^5 + 19t^4 - 42t^3 - 53t^2 - 60t - 52
\end{array}$$

The J_i's can be recovered easily if desired, by the formula

$$J_i = \frac{-1}{A_i + 432A_i^2}.$$

Then the half-abscissae of the 2–torsion points can be lifted. To precision $O(2^6)$, the resulting sequence is:

$$\begin{array}{rcl}
Z_0 &=& -25t^{10} + 7t^9 - 21t^8 - 3t^7 - 25t^6 + 6t^5 - 2t^4 + 22t^3 - 5t^2 + 32t - 11 \\
Z_1 &=& 26t^{10} + t^9 - 30t^8 + 30t^7 - 2t^6 - 7t^5 + 23t^4 + 25t^3 + 28t^2 + 23t + 19 \\
Z_2 &=& 32t^{10} - 9t^9 + 9t^8 + 26t^7 + 17t^6 - 30t^5 - 18t^4 - 26t^3 - 9t^2 - 4t + 5 \\
Z_3 &=& -23t^{10} + 22t^9 - 2t^8 + 2t^7 - 30t^6 - 16t^5 + 11t^4 + 11t^3 + 10t^2 + 6t - 22 \\
Z_4 &=& 13t^{10} + 19t^9 + 7t^8 - 3t^7 + 26t^6 - 25t^5 - 12t^4 + 26t^3 + 25t^2 - 13t - 7 \\
Z_5 &=& 10t^{10} + 29t^9 + 18t^8 + 19t^7 - 7t^6 + 3t^5 + 21t^4 - 30t^3 - 2t^2 + 29t + 7 \\
Z_6 &=& 22t^{10} + 9t^9 - 22t^8 + 17t^7 - 7t^6 - 26t^5 - 28t^4 - 13t^3 + 5t^2 - 21 \\
Z_7 &=& -23t^{10} + 14t^9 + 5t^8 + 3t^7 - 2t^6 - 14t^5 - 28t^4 - 21t^3 - 12t^2 + 8t + 18 \\
Z_8 &=& 14t^{10} + 28t^9 + 17t^8 - 5t^7 - 18t^6 - 25t^5 + 5t^4 - 6t^3 - 24t^2 + 2t + 12 \\
Z_9 &=& -16t^{10} - 28t^9 - 3t^8 + 6t^6 - 30t^5 - 15t^4 + 2t^3 + 21t^2 - 28t - 28 \\
Z_{10} &=& -30t^{10} - 28t^9 + 22t^8 - 22t^7 - 22t^6 - 28t^5 - 21t^4 + 10t^3 + 27t^2 + 9t - 28
\end{array}$$

The square of the trace can now be computed modulo 2^9 by the formula

$$c^2 = \frac{\prod_i (1 - 504Z_i + 19008A_i)}{\prod_i (1 + 240(Z_i + 12Z_i^2))(1 + 864A_i)} \equiv \frac{65}{129} \equiv 449.$$

The two square roots modulo 2^8 are

$$c \equiv \pm 31.$$

And we choose the one congruent to 1 modulo 4. Thus

$$c = -31.$$

6.2 New record

Here we present a record result obtained with the third author's implementation of the algorithm described above (dubbed ECPC for Elliptic Curve Point Counting). We computed the number of points on curves over several large binary fields, the largest being $\mathbb{F}_q$ with $q = 2^{8009}$. Some details of the calculation are given below.

Processor	CPU Time	Memory	Location
Alpha EV6, 750 MHz	313 h	16.9 GB	Cornell Computer Systems Laboratory

Several multiplication algorithms are implemented in ECPC. For small precisions it uses the naïve method and Karatsuba's algorithm. For medium precisions it switches to a 3-way or 4-way Toom-type algorithm with evaluation and interpolation at 5 or 7 rational points. Last but not least, the high precision calculations are performed using Fast Fourier Transforms modulo Fermat numbers in the style of Schönhage's algorithm. Thus it reaches in practice the best speed predicted by the asymptotic analysis.

The usefulness of these advanced algorithms can be observed in the following times for multiplication over $\mathbb{Z}_q$ at the maximal precision of 4008 bits:

Algorithm	Time
Naïve	866 s
Karatsuba	26.8 s
Toom3	12.7 s
Toom4	9.80 s
Schönhage	2.56 s

We represent the field $\mathbb{F}_q$ as $\mathbb{F}_2[t]/(f(t))$ where $f(t)$ is the irreducible polynomial $t^{8009}+t^{3159}+1$. Let E be the curve $y^2+xy=x^3+a$ where a is:

```
0x3F636F6420707520732774616857
```

This hexadecimal representation is obtained by reducing a modulo f and setting t. The value was chosen to be the ASCII encoding of the phrase "What's up doc?". The result we obtained is $q+1-c$ with the trace c as follows:

```
-17378149685594750502663500286963595380822933538157005646462119394089505
```

We checked that this result is indeed a multiple of the orders of several randomly chosen points on each curve. A rigorous check of the number of points would require $q+1\pm c$ to be factored completely, but this does not seem to be feasible.

For reference, the run-times for ECPC on some smaller field sizes are as follows.

Degree	Time	Clock speed	Degree	Time	Clock speed
120	0.8 s	750 MHz	1000	22 m	750 MHz
160	2.5 s	750 MHz	2000	3 h	750 MHz
200	4.6 s	750 MHz	4000	29 h	750 MHz
240	7.1 s	750 MHz	7001	265 h	667 MHz

Remarks

A demo version of ECPC can be found at URL: `http://www.xent.com/~harley/`

We note that very recently Berit Skjernaa has independently extended Satoh's algorithm to characteristic two using a different method.

Acknowledgements

We would like to thank François Morain for his continuous support and many invaluable suggestions during this work.

We are also grateful to Paul Bourke from Swinburne University Astrophysics and Supercomputing, to Tom Morris from Alpha Processor, Inc. and to Rajit Manohar from Cornell Computer Systems Laboratory. They provided the computer resources needed for many of our calculations.

7 Appendix: Characteristic three

Curves in characteristic three, like curves in characteristic two, have a particular form which induces some particularities in Satoh's algorithm. Here we give the required formulae and briefly describe the method for lifting the coefficients of the curve and the 3–torsion subgroup. Finally we derive the equation for the trace using Vélu's formulae.

7.1 Lifting the curve's equation

In characteristic three, the equation of an elliptic curve has the form $y^2 = x^3 + a_2x^2 + a_6$ whenever its j–invariant is non-zero, and then $j = -a_2^3/a_6$. The lifted curve $\mathcal{E}$ is of the form $y^2 = x^3 + A_2x^2 + A_6$ and its j–invariant is $J = -256A_2^6/(4A_2^3A_6 + 27A_6^2)$.

Therefore, to lift the coefficients of the curve, one uses almost the same algorithms as for characteristic $p \geq 5$. That is, one first lifts the coefficient A_2 arbitrarily and then one lifts A_6 using a Newton iteration based on $f(x) = J(A_2^3x + \frac{27}{4}x^2) + 64A_2^6$ and with initial approximate root $x_0 = a_6$.

7.2 Lifting the 3–torsion

The 3–division polynomial of $\mathcal{E}$ is $\Psi_3(x) = 3x^4 + 4A_2x^3 + 12A_6x + 4A_2A_6$ whereas the one of E is $\psi_3(x) = a_2(x^3 + a_6)$. We lift the 3–torsion subgroup using a Newton iteration based on the function $\Psi_3(x)$. The initial approximate root is $x_0 = -\sqrt[3]{a_6}$. This iteration is valid since we have:

$$\begin{aligned}\Psi_3(x_0) &= -3A_6x_0 - 4A_2A_6 + 12A_6x_0 + 4A_2A_6 = 9A_6x_0\\ \Psi_3'(x_0) &= -12A_6 + 12A_2x_0^2 + 12A_6 = 12A_2x_0^2\end{aligned}$$

Thus $\Psi_3(x_0) \equiv 0$ modulo 9 and $\Psi_3'(x_0) \equiv 0$ modulo 3 but not modulo 9 (note $A_2 \neq 0$ and $A_6 \neq 0$).

7.3 Isogenies and traces in characteristic three

For the computation of the trace the method is the same as for characteristic two, except that we have to take into account the form of the curve equations and also adapt Vélu's formulae.

The curve $\mathcal{E}_i$ has an equation of the form

$$\mathcal{E}_i \ : \ y^2 = x^3 + A_ix^2 + B_i.$$

We write X_i for the abscissa of the lifted 3–torsion point. Then Vélu's formulae give the following equation for the curve $\mathcal{E}_i'$:

$$y^2 = x^3 + a_2x^2 + a_4x + a_6$$

where

$$a_2 = A_i, \quad a_4 = -5t, \quad a_6 = B_i - 4A_it - 7w,$$

with

$$t = 6X_i^2 + 4A_iX_i, \quad \text{and} \quad w = 10X_i^3 + 8A_iX_i^2 + 4B_i.$$

Now we transform the equations of $\mathcal{E}_i'$ and $\mathcal{E}_{i+1}$ by translation of the axes to obtain

$$\begin{aligned}\mathcal{E}_{i+1} \ &: \ y \ = \ x^3 - \tfrac{1}{3}A_{i+1}^2x + B_i + \tfrac{2}{27}A_{i+1}^3\\ \mathcal{E}_i' \ &: \ y \ = \ x^3 + (a_4 - \tfrac{1}{3}a_2^2)x + a_6 + \tfrac{2}{27}a_2^3 - \tfrac{1}{3}a_2a_4\end{aligned}$$

The expression for g_i^2 follows easily:

$$g_i^2 = \frac{A_i'^2(-1890X_i^3 - 1890A_iX_i^2 - 252A_i^2X_i + 2A_i^3 - 729B_i)}{(2A_i'^3 + 27B_i')\left(90X_i^2 + 60A_iX_i + A_i^2\right)}.$$

Finally the trace is given by

$$\operatorname{Tr} F \equiv \prod_{0\le i<d} g_i \mod q.$$

References

[Atk92] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime. Serie of e-mails to the NMBRTHRY mailing list, 1992.

[Cou94] J.-M. Couveignes. *Quelques calculs en théorie des nombres.* Thèse, Université de Bordeaux I, July 1994.

[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.

[Dew98] L. Dewaghe. Remarks on the Schoof-Elkies-Atkin algorithm. *Math. Comp.*, 67(223):1247–1252, July 1998.

[Elk98] N. Elkies. Elliptic and modular curves over finite fields and related computational issues. In D.A. Buell and eds. 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.

[JL] A. Joux and R. Lercier. "Chinese & match", an alternative to Atkin's "Match and sort" method used in the SEA algorithm. *Math. Comp.*, to appear.

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

[LM00] R. Lercier and F. Morain. Computing isogenies between elliptic curves over F_{p^n} using Couveignes's algorithm. *Math. Comp.*, 69(229):351–370, January 2000.

[LST64] J. Lubin, J. P. Serre, and J. Tate. Elliptic curves and formal groups. In *Lecture notes prepared in connection with the seminars held at the Summer Institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massachusetts, July 6-July 31, 1964*, 1964. Scanned copies available at `http://www.ma.utexas.edu/users/voloch/lst.html`.

[Mor95a] 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.

[Mor95b] F. Morain. #E(GF($10^{499} + 153$)). E-mail to the NMBRTHRY mailing list, January 1995.

[Mül95] V. Müller. *Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer drei.* PhD thesis, Technischen Fakultät der Universität des Saarlandes, 1995.

[Sat00] T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *Journal of the Ramanujan Mathematical Society*, December 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.

[Ser68] J. P. Serre. *Corps locaux.* Hermann, 1968.

[Ser70] J. P. Serre. *Cours d'arithmétique.* Presses universitaires de France, 1970.

[Sil86] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics.* Springer–Verlag, 1986.

[SS71] A. Schönhage and V. Strassen. Schnelle multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[Vél71] J. Vélu. Isogénies entre courbes elliptiques. *C. R. Acad. Sci. Paris Sér. I Math.*, 273:238–241, July 1971. Série A.

[Ver99] F. Vercauteren. #EC(GF(2^{1999})). E-mail to the `NMBRTHRY` mailing list, October 1999.