

# Computing the eigenvalue in the Schoof-Elkies-Atkin algorithm using Abelian lifts

P. Mihăilescu\*, F. Morain†, É. Schost‡

February 9, 2007

## Abstract

The Schoof-Elkies-Atkin algorithm is the best known method for counting the number of points of an elliptic curve defined over a finite field of large characteristic. We use abelian properties of division polynomials to design a fast theoretical and practical algorithm for computing the eigenvalue search.

## 1 Introduction

The Schoof-Elkies-Atkin (SEA) algorithm [2] is currently the fastest known algorithm for computing the cardinality of elliptic curves defined over finite fields of large characteristic.

Following the initial work of Schoof [26], considerable work has been devoted to making this algorithm efficient [1, 10, 22, 27, 11, 19, 15]. One of the key ingredients of the SEA algorithm is the eigenvalue search in the so-called “Elkies case”, which is studied in [19, 15]. In the present article, we describe the complexity aspects and the implementation of a new approach of the first author [21], which uses abelian properties of the elliptic division polynomials to solve this particular problem, in an often faster way.

In Section 2, we recall briefly the SEA algorithm. Section 3 describes the use of Abelian lifts for computing the eigenvalue in the Elkies case. In Section 4, we first review classical algorithms that will be used in the implementation and add slight improvements to some variants; we then explain how to implement our algorithms and give examples. Section 5 gives some improvements to our basic scheme and Section 6 concludes the article with timings obtained with our NTL implementation.

## 2 The SEA algorithm

Throughout the article,  $E$  denotes an elliptic curve defined over the finite field  $\mathbb{F}_p$  by an equation of the form  $Y^2 = X^3 + AX + B$ . In all that follows, the  $X$  and  $Y$ -coordinates of a point  $P$  in the affine plane are denoted by subscripts, namely  $P_X$  and  $P_Y$ . We refer to [2] for the following facts.

### 2.1 Schoof’s original algorithm

There is a group law on an elliptic curve, that is known as the tangent-and-chord method; in particular, the  $m$ -fold multiple of a point  $P$  is written  $[m]P$ . Over any field, the addition formulae are rational; repeated use of these rules leads to the introduction of *division polynomials*, generally noted  $f_m(X)$ . For  $m \in \mathbb{N}$ , let  $E[m]$  denote the  $m$ -torsion subgroup of  $E$ ; then, the division polynomials are used to represent the coordinate ring of  $E[m]$  as

$$\mathbb{F}_p[X, Y]/(Y^2 - (X^3 + AX + B), f_m(X)).$$

---

\*Mathematisches Institut der Universität Göttingen, Germany, preda@uni-math.gwdg.de

†LIX, École polytechnique, Palaiseau, France, morain@lix.polytechnique.fr

‡ORCCA and Computer Science Department, University of Western Ontario, London, ON, Canada, schost@scl.csd.uwo.ca

Let  $\pi$  be the Frobenius endomorphism of  $E(\overline{\mathbb{F}}_p)$  that sends  $(X, Y)$  to  $(X^p, Y^p)$ . This endomorphism satisfies an equation of the form  $\pi^2 - t\pi + p = 0$ ; the polynomial  $T^2 - tT + p$  is called the *characteristic polynomial* of  $\pi$  and the number of  $\mathbb{F}_p$ -rational points of  $E$  is then equal to  $p + 1 - t$ . Hasse's bound ensures that the absolute value of the *trace*  $t$  is bounded by  $2\sqrt{p}$ . In what follows, we will be interested in *non supersingular* curves, for which  $t \neq 0$ , since the cardinality of supersingular curves is simply  $p + 1$ .

To compute the cardinality of  $E(\mathbb{F}_p)$ , Schoof's algorithm proceeds by computing  $t$  modulo small primes  $\ell$  using the action of  $\pi$  on the set of  $\ell$ -torsion points  $E[\ell]$ , until enough modular information is known to reconstruct the trace and the cardinality of  $E$  by the Chinese Remainder Theorem.

## 2.2 The Elkies case

In the so-called Elkies case, the characteristic polynomial of  $\pi$  has two linear factors modulo  $\ell$ , so that the restriction of  $\pi$  to  $E[\ell]$  has two rational eigenspaces. One of these (call it  $V$ ) is characterized by a polynomial  $f_{\ell,\lambda}(X)$  of degree  $(\ell - 1)/2$  which divides  $f_\ell(X)$ ; in particular, the coordinate ring of  $V$  can be represented as

$$\mathcal{A} = \mathbb{F}_p[X, Y]/(Y^2 - (X^3 + AX + B), f_{\ell,\lambda}(X)).$$

The action of the Frobenius endomorphism on  $(X, Y) \in \mathcal{A}$  is simply  $\pi(X, Y) = (X^p, Y^p)$ , with the first term being reduced modulo  $f_{\ell,\lambda}(X)$  and the second modulo  $f_{\ell,\lambda}(X)$  and  $Y^2 - (X^3 + AX + B)$ .

We shall be interested in computing the eigenvalue of  $\pi$ , namely the integer  $\lambda$ ,  $0 < \lambda < \ell$  such that  $\pi(X, Y) = [\lambda](X, Y)$  or:

$$(X^p, Y^p) = [\lambda](X, Y). \tag{1}$$

Indeed, the trace modulo  $\ell$  is then deduced from the formula  $t \equiv \lambda + p/\lambda \pmod{\ell}$  (note that  $\lambda \neq 0$  for non supersingular curves).

Algorithms related to this case may be found in [27, 22]. On a heuristic basis, for a general curve, we expect to be in the Elkies case for about half of the primes  $\ell$ . Combining this with Hasse's bound and the prime number theorem, we obtain that, asymptotically, the largest  $\ell$  we have to consider is about  $\log(p)$ , where  $\log$  is the natural logarithm.

For a given  $\ell$ , we are in the Elkies case if the univariate polynomial  $\Phi_\ell(X, j(E))$  has a root in  $\mathbb{F}_p$ , where  $\Phi_\ell \in \mathbb{Z}[X, Y]$  is an equation of the modular curve  $X_0(\ell)$  (see [12] for fast algorithms for computing modular polynomials). Generically, the equation  $\Phi_\ell(X, j(E)) = 0$  has actually two roots and this is what we will be considering (when the number of roots is not 2, we know the eigenvalue up to sign). One can then recover the factor  $f_{\ell,\lambda}(X)$  of  $f_\ell(X)$  in quasi-linear time [3].

We address in this paper the following question: given  $f_{\ell,\lambda}(X)$ , compute the eigenvalue  $\lambda$  satisfying Equation (1). Several variants are presented in [15] for this task; the fastest algorithm has complexity

$$O(M(\ell) \log(p) + \sqrt{\ell}M(\ell)),$$

where  $M(d)$  denotes the cost of multiplying two polynomials of degree less than  $d$  in  $\mathbb{F}_p[X]$ , see Section 4 for details. We present in the next section a different algorithm, which turns out to be faster in many cases.

## 3 Abelian lifts

We introduce in this section the main ingredients for our algorithm. Lifting the situation to characteristic zero, we first study the Galois structure of extensions generated by lifts of division polynomials. Reducing back to characteristic  $p$ , this enables us to obtain the eigenvalue  $\lambda$  by working in extensions of  $\mathbb{F}_p$  of degree  $q$ , for  $q$  a divisor of  $(\ell - 1)/2$ , as opposed to standard computations which take place modulo  $f_{\ell,\lambda}(X)$ , that is, in degree  $(\ell - 1)/2$ . Combining the values obtained for coprime divisors of  $\ell - 1$  will give us the answer.

It should be noted that the theory works for any divisor of  $\ell - 1$ , including even integers. For the even part of the index, one has to consider the  $Y$ -coordinates of the torsion points, while the odd part involves  $X$  only.

### 3.1 Lifting to characteristic zero

We start by lifting to characteristic zero; characteristic zero lifts of objects existing modulo  $p$  will usually be written with a bar. Let thus  $\mathbb{K}$  be the field of definition of the Deuring lift [17]

$$\bar{E} : Y^2 = X^3 + \bar{A}X + \bar{B}$$

of the curve  $E/\mathbb{F}_p$  and let  $\mathbb{K}_\ell = \mathbb{K}[X]/(\bar{f}_\ell(X))$  be the extension of degree  $\ell(\ell-1)/2$  generated by the  $\ell$ -division polynomial; we let  $\Theta \in \mathbb{K}_\ell$  be the residue class  $(X \bmod \bar{f}_\ell(X))$ . Let next  $\mathbb{L}_\ell$  be the extension

$$\mathbb{L}_\ell = \mathbb{K}_\ell[Y]/(Y^2 - (X^3 + \bar{A}X + \bar{B})),$$

and let  $\Gamma$  be the residue class of  $Y$  in  $\mathbb{L}_\ell$ .

Consider now the “generic” point  $\bar{P} = (\Theta, \Gamma)$  of  $\bar{E}(\mathbb{L}_\ell)$ . For  $a \in (\mathbb{Z}/\ell\mathbb{Z})^*$ , the action

$$\rho_a : \Theta \mapsto ([a]\bar{P})_X$$

defines an automorphism of  $\mathbb{K}_\ell/\mathbb{K}$  and

$$G = \left\{ \rho_a : 1 \leq a \leq \frac{\ell-1}{2} \right\}$$

is a cyclic subgroup of the Galois group  $\text{Gal}(\mathbb{K}_\ell/\mathbb{K})$ . Let  $\mathbb{K}_0$  be the fixed field  $\mathbb{K}_\ell^G$ ; then, the extension  $\mathbb{K}_\ell/\mathbb{K}_0$  is cyclic of degree  $(\ell-1)/2$ . The polynomial  $f_{\ell,\lambda}(X)$  factors as

$$\bar{f}_{\ell,\lambda}(T) = \prod_{a=1}^{\frac{\ell-1}{2}} (T - \rho_a(\Theta)) \in \mathbb{K}_0[T];$$

it is thus a cyclic polynomial for which  $\mathbb{K}_\ell = \mathbb{K}_0[T]/(\bar{f}_{\ell,\lambda}(T))$ . For  $a$  in  $(\mathbb{Z}/\ell\mathbb{Z})^*$ , define the unique polynomial  $\bar{g}_a \in \mathbb{K}_0[X]$  by the condition that  $\deg(\bar{g}_a(X)) < (\ell-1)/2$  and  $\bar{g}_a(\Theta) = \rho_a(\Theta)$ . We have the composition rules

$$\bar{g}_a(\bar{g}_b(\Theta)) = \bar{g}_b(\bar{g}_a(\Theta)) = \bar{g}_{ab}(\Theta) = \rho_{ab}(\Theta)$$

and in particular, we can rewrite

$$\bar{f}_{\ell,\lambda}(T) = \prod_{a=1}^{\frac{\ell-1}{2}} (T - \bar{g}_a(\Theta)) \in \mathbb{K}_0[T].$$

It is useful to note that the set  $\{\rho(\Theta) : \rho \in G\}$  is a normal basis of  $\mathbb{K}_\ell/\mathbb{K}_0$ . This has in particular as consequence the fact that for each subfield  $\mathbb{K}_0 \subset \mathbb{K}' \subset \mathbb{K}_\ell$ , the trace  $\mathbf{Tr}_{\mathbb{K}_\ell/\mathbb{K}'}(\Theta)$  together with its conjugates form a normal basis for  $\mathbb{K}'/\mathbb{K}_0$  and generates the field  $\mathbb{K}'$  as a simple extension.

The basic idea of our approach is to consider traces in several such subextensions  $\mathbb{K}_0 \subset \mathbb{K}' \subset \mathbb{K}_\ell$  of coprime degrees  $q = [\mathbb{K}' : \mathbb{K}_0]$ . We will thus write  $\bar{\tau}_q = \mathbf{Tr}_{\mathbb{K}_\ell/\mathbb{K}'}(\Theta)$ , so that  $\mathbb{K}' = \mathbb{K}_0[\bar{\tau}_q]$  by what was said previously. For a given  $q$ , the action of the Frobenius on these  $\bar{\tau}_q$  will allow us to compute the residue modulo  $q$  of the index of  $\lambda$  in  $(\mathbb{Z}/\ell\mathbb{Z})^*$ ; using Chinese Remaindering will yield  $\lambda$ .

For  $q$  odd, the previous setting is enough, but to treat the case  $q$  even, we have to extend the discussion and consider  $Y$ -coordinates. The extension  $\mathbb{L}_\ell = \mathbb{K}_\ell[\Gamma]$  has degree 2, with Galois group generated by  $\varsigma : \Gamma \mapsto -\Gamma$ . The automorphism  $\varsigma$  acts naturally on  $\mathbb{K}_\ell$  as the identity and this makes it an element  $\varsigma \in \text{Gal}(\mathbb{L}_\ell/\mathbb{K}_0)$ . Let  $F(X) = X^3 + \bar{A}X + \bar{B}$  be the polynomial generating the curve  $\bar{E}$  and define

$$S(Y) = \prod_{a=1}^{(\ell-1)/2} (Y^2 - F(\rho_a(\Theta))) \in \mathbb{K}_0[Y].$$

Some algebraic verifications show that  $\mathbb{L}_\ell = \mathbb{K}_0[Y]/(S(Y))$  and that  $S(Y)$  splits completely over  $\mathbb{K}_0$ , making the extension  $\mathbb{L}_\ell/\mathbb{K}_0$  Galois. From the theory of division functions, one gathers that there are polynomials  $h_b(X) \in \mathbb{K}_0[X]$  such that

$$([b]\overline{Q})_Y = \overline{Q}_Y \cdot h_b(\overline{Q}_X), \quad \forall \overline{Q} \in \langle \overline{P} \rangle.$$

We claim that  $\mathbb{L}_\ell/\mathbb{K}_0$  is abelian and in fact that

$$H = \text{Gal}(\mathbb{L}_\ell/\mathbb{K}_0) = \langle \varsigma \rangle \times G.$$

For this, we show how to lift  $\rho_a$  to  $\tilde{\rho}_a \in H$ ; we do this for  $a = c$ , a generator of  $(\mathbb{Z}/\ell\mathbb{Z})^*$ , by the natural definition:

$$\tilde{\rho}_c(\Gamma) = ([c]\overline{P})_Y = \Gamma \cdot h_c(\Theta).$$

Note first that  $\tilde{\rho}_c(\Gamma^2) = F(g_c(\Theta)) = F(\Theta) \cdot h_c(\Theta)^2$ , which shows that the restriction of  $\tilde{\rho}_c$  on  $\mathbb{K}_\ell$  acts indeed like  $\rho_c$ . One then verifies that  $\varsigma \circ \tilde{\rho}_c = \tilde{\rho}_c \circ \varsigma$  and that it generates an automorphism group acting on  $\mathbb{L}_\ell/\mathbb{L}_0$ , where  $[\mathbb{L}_0 : \mathbb{K}_0] = 2$  (we refer to [21] for the details).

Now, for an intermediate field  $\mathbb{K}_0 \subset \mathbb{K}' \subset \mathbb{K}_\ell$  as above, with  $q = [\mathbb{K}' : \mathbb{K}_0]$ , observe the existence of an intermediate *ordinate*-field

$$\mathbb{L}' = \mathbb{K}'[\overline{\tau}'_q] \quad \text{with} \quad \overline{\tau}'_q = \sum_{a=1}^{(\ell-1)/2q} \tilde{\rho}_{c^{aq}}(\Gamma),$$

which is an extension of degree 2 of  $\mathbb{K}'$ , with in fact  $\overline{\tau}'_q{}^2 \in \mathbb{K}'$  (the field  $\mathbb{L}_0$  above is one of these fields).

We finally introduce elliptic Gaussian periods. As before, we shall assume that  $c$  is a generator of  $(\mathbb{Z}/\ell\mathbb{Z})^*$ . We suppose first that  $q$  is odd and develop this case in more detail; in this case, we need only work with  $X$ -coordinates. Accordingly, we let  $q' = (\ell - 1)/(2q)$ ,  $h = c^q$ ,  $k = c^{q'}$  and

$$(\mathbb{Z}/\ell\mathbb{Z})^*/\{\pm 1\} = H \times K \quad \text{with} \quad H = \langle h \rangle, \quad K = \langle k \rangle.$$

For  $0 \leq i < q$ , we define

$$\overline{\eta}_i = \sum_{a \in H} ([k^i \cdot a]\overline{P})_X = \sum_{a \in H} \rho_a(\rho_{k^i}(\Theta));$$

then,  $\overline{\eta}_0$  is the trace  $\overline{\tau}_q$  defined above, and  $\overline{\eta}_i = \rho_k^{(i)}(\overline{\eta}_0)$  for all  $i$ . As a consequence, there is a cyclic action:

$$\overline{\eta}_0 \xrightarrow{\rho_k} \overline{\eta}_1 \xrightarrow{\rho_k} \dots \xrightarrow{\rho_k} \overline{\eta}_{q-1} \xrightarrow{\rho_k} \overline{\eta}_0, \quad (2)$$

so that the minimal polynomial of  $\overline{\eta}_0$  is

$$\overline{M}(T) = \prod_{i=0}^{q-1} (T - \overline{\eta}_i) \in \mathcal{O}(\mathbb{K}_0)[T]$$

and  $\mathbb{K}' = \mathbb{K}_0[T]/(\overline{M}(T))$ . Since the extension  $\mathbb{K}'/\mathbb{K}_0$  is cyclic, there is a polynomial  $\overline{C} \in \mathbb{K}_0[T]$  which encodes the action of  $\rho_k$ , so  $\overline{C}(\overline{\eta}_0) = \rho_k(\overline{\eta}_0) = \overline{\eta}_1$ . Iterating the construction gives

$$\overline{\eta}_i = \overline{C}^{(i)}(\overline{\eta}_0),$$

where the exponent  $(i)$  denotes  $i$ -fold composition of the polynomial  $C$  with itself.

In the case when  $q$  is even, we let  $q' = (\ell - 1)/q$  and  $h, k, H, K$  like previously, noting that this time  $H \times K = (\mathbb{Z}/\ell\mathbb{Z})^*$ . We work accordingly in  $\mathbb{L}'$  instead of  $\mathbb{K}'$  and define

$$\overline{\eta}'_i = \sum_{a \in H} ([k^i \cdot a]\overline{P})_Y = \sum_{a \in H} \tilde{\rho}_a(\tilde{\rho}_{k^i}(\Gamma));$$

then,  $\overline{\eta}'_0$  is the trace  $\overline{\tau}'_q$  defined above, and  $\overline{\eta}'_i = \tilde{\rho}_k^{(i)}(\overline{\eta}'_0)$  for all  $i$ . As a consequence, there is a cyclic action which now is two-phased:

$$\overline{\eta}'_0 \xrightarrow{\tilde{\rho}_k} \overline{\eta}'_1 \xrightarrow{\tilde{\rho}_k} \dots \xrightarrow{\tilde{\rho}_k} \overline{\eta}'_{q/2-1} \xrightarrow{\varsigma} \overline{\eta}'_{q/2} \xrightarrow{\tilde{\rho}_k} \dots \xrightarrow{\tilde{\rho}_k} \overline{\eta}'_{q-1} \xrightarrow{\tilde{\rho}_k} \overline{\eta}'_0. \quad (3)$$

Now the minimal polynomial of  $\overline{\eta}'_0$  is

$$\overline{M}(T) = \prod_{i=0}^{q/2-1} (T^2 - (\overline{\eta}'_i)^2) = \overline{N}(T^2) \in \mathcal{O}(\mathbb{K}_0)[T].$$

and  $\mathbb{L}' = \mathbb{K}_0[T]/(\overline{M}(T))$ . The extension  $\mathbb{L}'/\mathbb{K}_0$  is abelian, yet this time not cyclic. There also is a polynomial  $\overline{C} \in \mathbb{K}_0[T]$  which encodes the action of  $\tilde{\rho}_k$ , so that we have

$$\overline{C}(\overline{\eta}'_0) = \tilde{\rho}_k(\overline{\eta}'_0) = \overline{\eta}'_1 \quad \text{and} \quad \overline{\eta}'_{q-1-i} = \varsigma(\overline{\eta}'_i) = -\overline{\eta}'_i.$$

Let us define

$$\begin{aligned} j(i) &= i & \text{and} & \quad s(i) = 1 & \text{for} & \quad i < q/2 \\ j(i) &= q-1-i & \text{and} & \quad s(i) = -1 & \text{for} & \quad q/2 \leq i < q. \end{aligned}$$

With this we can group the previous to

$$\overline{\eta}'_i = s(i) \cdot \overline{C}^{(j(i))}(\overline{\eta}'_0).$$

### 3.2 Finding the eigenvalue

We can then reduce the previous construction back to  $\mathbb{F}_p$ . Let us consider the algebras

$$\mathcal{A}_0 = \mathbb{F}_p[X]/(f_{\ell,\lambda}(X))$$

and

$$\mathcal{A} = \mathbb{F}_p[X, Y]/(Y^2 - (X^3 + AX + B), f_{\ell,\lambda}(X)),$$

the later of which was defined in Subsection 2.2. Let  $\theta$  and  $\gamma$  be the residue classes of  $X$  and  $Y$  in  $\mathcal{A}$  and let  $P = (\theta, \gamma)$  be a “generic” point on  $E$ . For  $a \in (\mathbb{Z}/\ell\mathbb{Z})^*$ , we define the unique *multiplication by  $a$ -polynomial* of  $\mathcal{A}$  by the condition  $\deg(g_a(X)) < (\ell-1)/2$  and  $g_a(\theta) = ([a]P)_X \in \mathcal{A}$ . Remark that the eigenvalue  $\lambda$  satisfies the relation  $\theta^p = g_\lambda(\theta)$ .

Since  $\ell$  is an Elkies prime, there is a prime ideal  $\wp \subset p \cdot \mathcal{O}(\mathbb{K}_0)$  such that  $\overline{f}_{\ell,\lambda}(X) \bmod \wp = f_{\ell,\lambda}(X)$ ; the polynomial  $f_{\ell,\lambda}(X)$  has thus  $\overline{f}_{\ell,\lambda}(X) \in \mathcal{O}(\mathbb{K}_0)[X]$  as a cyclic lift. Similarly, for all  $a$ ,  $\overline{g}_a(X)$  is a lift of  $g_a(X)$ . The definition of the multiplication polynomials  $g_a(X)$  then shows that we have

$$f_{\ell,\lambda}(Z) = \prod_{a=1}^{(\ell-1)/2} (Z - g_a(\theta)).$$

We have depicted the situation in Figure 1.

Letting as above  $c$  be a generator of  $(\mathbb{Z}/\ell\mathbb{Z})^*$ , we denote by  $x$  the index of  $\lambda$ , so that  $\lambda = c^x \bmod \ell$ . Using the construction of the previous subsection, we describe now how to recover  $(x \bmod q)$ , giving details for  $q$  an odd divisor of  $(\ell-1)/2$ . As in the previous subsection, this approach extends to the case of even  $q$ ; we refer to Subsection 4.2 for a description of the algorithm in this case. For  $0 \leq i < q$ , we define

$$\eta_i = \sum_{a \in H} g_a(g_{k^i}(\theta)),$$

so that in particular  $\eta_i = \overline{\eta}'_i \bmod \wp$ . For odd  $\ell \geq 3$ , the discriminant of  $f_\ell(X)$  satisfies the following relation:

$$\text{Disc}(f_\ell) = (-1)^{(\ell-1)/2} \ell^{(\ell^2-3)/2} (-\Delta)^{(\ell^2-1)(\ell^2-3)/24},$$

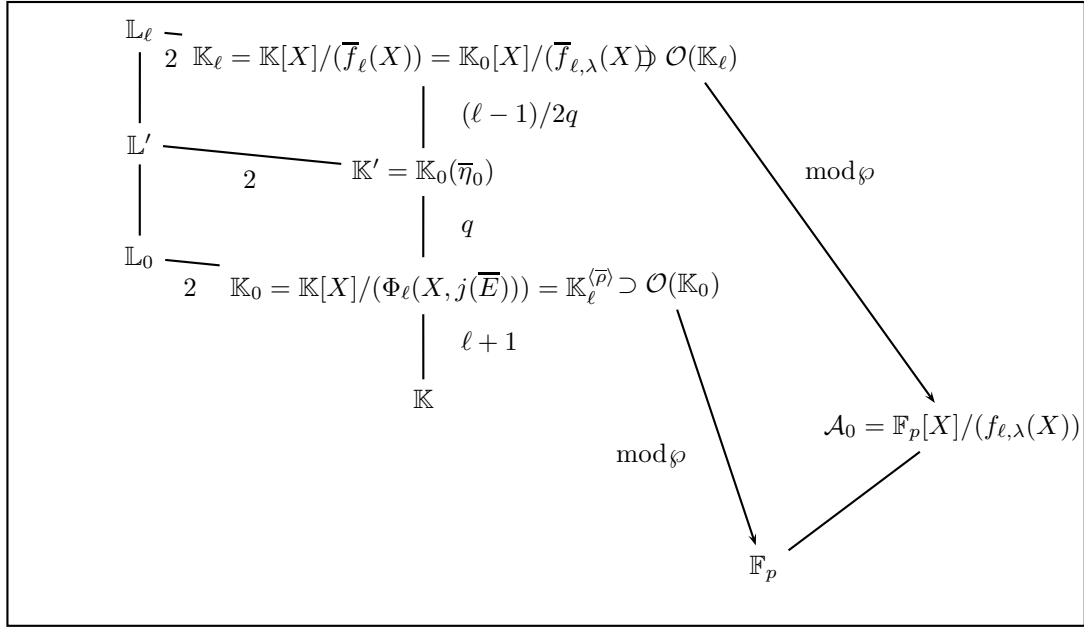


Figure 1: Extension fields.

where  $\Delta(E)$  is the discriminant  $-2^4(4A^3 + 27B^2)$ . We can thus safely assume that all roots of  $f_\ell(X)$ , and thus of  $f_{\ell,\lambda}(X)$ , are distinct. This implies that for  $i \neq j$ ,  $\eta_i \neq \eta_j$ , since such an equality would imply the existence of a linear relation between the roots of  $f_{\ell,\lambda}(X)$ . As a consequence, the reduction of the minimal polynomial  $\overline{M}(X)$  of  $\overline{\eta}_0$  is separated.

We let  $M(X) \in \mathbb{F}_p[X]$  be the minimal generating polynomial of the sequence of powers of  $\eta_0 \in \mathcal{A}_0$  and note that by the above remark, we have  $M(X) = \overline{M}(X) \bmod \wp$ ; hence,  $M(X)$  has degree  $q$ .

Writing  $C(T) = \overline{C}(T) \bmod \wp$ , we also have the relation  $\eta_i = C^{(i)}(\eta_0)$ , by reduction modulo  $\wp$  of the relation holding in  $\mathbb{K}'$ . The importance of the abelian lift consists in granting the existence of the polynomials  $C(T)$  which are not describing automorphisms in a field theoretic sense, but are images of automorphisms from the lift. In a general theory of Galois extensions of rings, see e.g. [18, 20], we encounter  $C(X)$  as automorphisms of rings (algebras).

Let us finally show how to recover  $(x \bmod q)$ . There exists  $v \in \mathbb{Z}/q\mathbb{Z}$  with  $T^p = C^{(v)}(T) \bmod M(T)$ , so that

$$\eta_0^p = C^{(v)}(\eta_0) = \eta_v.$$

But  $\eta_0^p = \eta_\lambda$ , with the natural periodic extension of the index of  $\eta$ . Therefore  $c^x = c^{q'v} \bmod \ell$  or  $x = q'v \bmod q$ ; this yields the required value of the index modulo  $q$ .

## 4 Algorithms and complexity

We give here the details of the algorithm and its complexity analysis. We use the standard notation  $M(n)$  to designate the time needed to compute the product of polynomials of degrees less than  $n$  over our base field [13, Chapter 8]. We make the classical super-linearity assumption that  $M(n+n') \geq M(n) + M(n')$  holds for all  $n, n'$ . Over fields supporting Fast Fourier Transform, one can take  $M(n) \in O(n \log(n))$ ; in all cases, one has  $M(n) \in O(n \log(n) \log \log(n))$ , using the algorithms of [25, 5].

Besides polynomial multiplication, our algorithms rely on matrix operations. We will thus denote by  $\omega < 3$  an exponent such that matrices in  $\mathbb{F}_p^{n \times n}$  can be multiplied in  $O(n^\omega)$  operations. The current record

is Coppersmith and Winograd's  $\simeq 2.38$  exponent [6].

In what follows, the *minimal polynomial* of an element  $\alpha$  in a finite-dimensional  $\mathbb{F}_p$ -algebra  $\mathcal{A}$  is the minimal generating polynomial of the sequence of powers of  $\alpha \in \mathcal{A}$  (so it is not necessarily irreducible, unless  $\mathcal{A}$  is a field). It is also the minimal polynomial of the multiplication-by- $\alpha$  endomorphism of  $\mathcal{A}$ .

## 4.1 Preliminaries

We start with some auxiliary algorithms on polynomials. Most results presented here are known; those not already in the literature are straightforward generalizations of existing ones.

**Modular composition and related problems.** Let  $\mathcal{C}(n)$  be an upper bound on the cost of computing  $g(h) \bmod f$ , where  $f, g, h$  are in  $\mathbb{F}_p[X]$ , of degrees  $n$ . Using Brent-Kung's modular composition algorithm [4], one can take

$$\mathcal{C}(n) \in O(n^{1/2}M(n) + n^{(\omega+1)/2}).$$

When  $g$  has degree  $q \leq n$ , the upper bound reduces to

$$O(q^{1/2}M(n) + q^{(\omega-1)/2}n).$$

We will assume that  $M(n) \log(n) \in O(\mathcal{C}(n))$ . We will need two variants of this algorithm. We write  $\mathcal{C}_r(n)$  for the cost of performing  $r$  modular compositions  $\{g_i(h) \bmod f\}_{1 \leq i \leq r}$ , where  $f$  and  $h$  are fixed. For  $r \in O(n)$ , using the algorithm of [29, 16], one can take

$$\mathcal{C}_r(n) \in O(r^{1/2}n^{1/2}M(n) + r^{(\omega-1)/2}n^{(\omega+1)/2}).$$

Finally, we consider iterated compositions: given  $k \in O(n)$ , compute

$$h, h(h) \bmod f, \dots, h^{(k)} \bmod f,$$

under the assumption that  $f$  divides  $f(h)$ . We can then use the doubling algorithm of [16, Lemma 4]: supposing that

$$H_1 = h, H_2 = h(h) \bmod f, \dots, H_j = h^{(j)} \bmod f$$

are known, we deduce

$$H_{j+1} = H_1(H_j) \bmod f, \dots, H_{2j} = H_j(H_j) \bmod f.$$

We repeat this scheme for  $j = 1, 2, \dots, 2^{\lceil \log(k) \rceil}$ . The total cost is then within a constant times that of the last step, that is, in  $O(\mathcal{C}_k(n))$ .

**Minimal polynomials and related problems.** Let  $f(X) \in \mathbb{F}_p[X]$  be of degree  $n$ , and let  $\alpha$  be in  $\mathcal{A} = \mathbb{F}_p[X]/(f(X))$ . Given a linear form  $\mathcal{L} : \mathcal{A} \rightarrow \mathbb{F}_p$  and  $q \leq n$ , the sequence  $[\mathcal{L}(\alpha^i)]_{0 \leq i \leq q}$  can be computed in time  $\mathcal{C}(n)$  see [28, 30]; a more precise bound is

$$O(q^{1/2}M(n) + q^{(\omega-1)/2}n).$$

In particular, if the minimal polynomial of  $\alpha$  has degree  $q$ , it can be computed within the same complexity, up to a negligible  $O(M(q) \log q)$  term [28, 30].

Let now  $\beta$  be in  $\mathcal{A}$ , and suppose that there exists  $C \in \mathbb{F}_p[X]$  of degree less than  $q$  such that  $\beta = C(\alpha)$ , where  $q$  is the degree of the minimal polynomial of  $\alpha$ . In [28, Theorem 5], Shoup gave an algorithm of complexity  $O(\mathcal{C}(n))$  for computing  $C$ , under the condition that  $f$  is irreducible.

This algorithm could be extended to the general case by using randomization. We present a different solution using the trace form as in [24, 23], which applies in characteristic  $p > n$ ; we also mention how to obtain an upper bound of

$$O(q^{1/2}M(n) + q^{(\omega-1)/2}n).$$

For simplicity, we assume that the characteristic polynomial  $\chi(X)$  of  $\alpha$  is a power of its minimal polynomial  $M(X)$ , say  $\chi(X) = M(X)^r$ . We also assume that  $f(X)$  is squarefree (all these assumptions are satisfied below).

Let  $\theta \in \mathcal{A}$  be the residue class  $(X \bmod f)$  and let  $\mathbf{Tr} = \mathbf{Tr}_{\mathcal{A}/\mathbb{F}_p}$  be the trace  $\mathcal{A} \rightarrow \mathbb{F}_p$ . The values  $\mathbf{Tr}(\theta^i)$  for  $0 \leq i < n$  can be computed in  $O(M(n))$  operations, as the coefficients of the expansion of  $f'/f$  at infinity. Using the “transposed multiplication” algorithm of [30], one can then compute in  $O(M(n))$  operations the linear form  $\mathcal{L} : \mathcal{A} \rightarrow \mathbb{F}_p$  such that  $\mathcal{L}(u) = \mathbf{Tr}(\beta u)$  holds for all  $u \in \mathcal{A}$ . Then following [24, 23], one sees that

$$\sum_{i=0}^{\infty} \frac{\mathcal{L}(\alpha^i)}{X^{i+1}} = r \frac{C(X)}{M(X)}$$

holds. Once the sequence  $(\mathcal{L}(\alpha^i))_{i \leq q}$  is known, the polynomial  $C(X)$  can thus be recovered in  $M(n)$  operations. As said above, the former sequence can be computed in time  $O(q^{1/2}M(n) + q^{(\omega-1)/2}n)$ , proving our claim.

## 4.2 Main algorithm

We return to the context of the previous sections. Let thus  $V$  be an eigenspace of  $E[\ell]$  associated to an eigenfactor  $f_{\ell,\lambda}(X)$ . We will now describe the details of our algorithm, starting in the case  $q$  odd; we conclude by the modifications to bring for  $q$  even.

**Compositions.** Most operations will be modular compositions modulo  $f_{\ell,\lambda}(X)$ . Write as before  $\mathcal{A}_0 = \mathbb{F}_p[X]/(f_{\ell,\lambda}(X))$ , and let  $\theta$  be the residue class of  $X$  in  $\mathcal{A}_0$ . Given  $R = R_X(\theta)$  and  $S = S_X(\theta)$  in  $\mathcal{A}_0$ , we write  $R[S] = T_X(\theta)$ , with  $T_X = R_X(S_X) \bmod f_{\ell,\lambda}$ . In particular, if  $R_X = g_a$  and  $S_X = g_b$  (as defined in Subsection 3.2), then  $T_X = g_{ab}$ . The cost of computing  $R[S]$  is thus of  $\mathcal{C}(\ell)$  base field operations.

**Computing  $\eta_0$ .** We continue with the previous notations. As in Section 3, we let  $c$  be a generator of  $(\mathbb{Z}/\ell\mathbb{Z})^*$ , let  $q$  be an odd divisor of  $(\ell - 1)/2$  and define

$$q' = (\ell - 1)/(2q), \quad h = c^q \bmod \ell, \quad k = c^{q'} \bmod \ell.$$

Writing  $H = \langle c \rangle$  in  $(\mathbb{Z}/\ell\mathbb{Z})^*/\{\pm 1\}$ , we need to compute

$$\eta_0 = \sum_{j=0}^{q'-1} g_{hj}(\theta).$$

For  $b$  in  $\mathbb{N}$ , define

$$T_{0,X}(h, b) = \sum_{j=0}^{b-1} g_{hj}(\theta),$$

the subscript  $X$  indicating that we consider only  $X$ -coordinates. We will compute  $\eta_0 = T_{0,X}(h, q')$  by an adaptation of von zur Gathen and Shoup’s trace algorithm [14, Algorithm 5.2]. Exploiting the relation

$$T_{0,X}(h, b + b') = T_{0,X}(h, b)[g_{hb'}] + T_{0,X}(h, b'), \quad (4)$$

we are led to the following divide-and-conquer algorithm.

```

T0,X(h, b)
1  U ← gh;
2  V ← θ;
3  W ← 0;
4  while b > 0 do
4.1  if b is even then
4.1.1 (U, V, W) ← (U[U], V[U] + V, W);
4.2  else
4.2.1 (U, V, W) ← (U[U], V[U] + V, W[U] + V);
4.3  b ← b/2;
5  return W;

```

As in von zur Gathen and Shoup's algorithm, at step  $i$ , we have

$$U = g_{2^i}, \quad V = T_{0,X}(h, 2^i), \quad W = T_{0,X}(h, b \bmod 2^i).$$

The total cost is in

$$O(\mathbf{M}(\ell) \log(\ell) + \mathcal{C}(\ell) \log(q')) \subset O(\mathcal{C}(\ell) \log(q')),$$

where, in the left-hand estimate, the first term accounts for the cost of computing  $g_h$  in Step 1 by repeated doubling, using the addition formulae modulo  $f_{\ell,\lambda}(X)$ , and the second term accounts for the loop in Step 5.

**Computing  $\eta_1$ .** We continue with the computation of

$$\eta_1 = \eta_0[g_k].$$

We denote by  $T_{1,X}(\eta_0)$  a function that performs this operation; its cost is in

$$O(\mathbf{M}(\ell) \log(\ell) + \mathcal{C}(\ell)) \subset O(\mathcal{C}(\ell)),$$

where in the left-hand estimate, the first term accounts for the cost of computing  $g_k$  by repeating doubling.

**Main algorithm.** We can now give the details of our main algorithm. Letting  $x$  be the index of  $\lambda$  in  $(\mathbb{Z}/\ell\mathbb{Z})^*$ , so that  $\lambda = c^x$ , this algorithm computes  $x \bmod q$ .

LogEigenvalue( $q, \ell, f_{\ell,\lambda}$ )	
1	$q' \leftarrow (\ell - 1)/(2q)$ ;
2	$\eta_0 \leftarrow T_{0,X}(h, q')$ ;
3	$\eta_1 \leftarrow T_{1,X}(\eta_0)$ ;
4	$M(T) \leftarrow \text{MinimalPolynomial}(\eta_0) \in \mathbb{F}_p[T]$ ;
5	compute $C(T)$ such that $\eta_1 = C(\eta_0)$ .
6	$T_p \leftarrow T^p \bmod M(T)$ .
7	find $0 \leq v < q$ such that $T_p = C^{(v)} \bmod M(T)$ .
8	return $q'v \bmod q$ .

**Proposition 4.1** *The previous algorithm has complexity*

$$O(\mathcal{C}(\ell) \log(q') + \mathbf{M}(q) \log(p) + \mathcal{C}_{\sqrt{q}}(q)).$$

PROOF. We use the results of the previous subsection. Steps 2 and 3 can be done in total time  $O(\mathcal{C}(\ell) \log(q'))$ . The cost of Steps 4 and 5 is  $O(\mathcal{C}(\ell))$  and that of Step 6 is  $O(\mathbf{M}(q) \log(p))$ . For Step 7, we use a baby steps/giant steps algorithm, by determining  $i, j \leq \sqrt{q}$  such that

$$T_p^{(i)} = C^{(j)} \bmod M(T),$$

since then we have  $v = j/i \bmod q$ . Both sequences are computed by the doubling algorithm for iterated modular composition that is presented in the previous subsection — this algorithm boils down to the one in [16, Lemma 4] in the case of  $T_p$ . The cost of this step is thus in  $O(\mathcal{C}_{\sqrt{q}}(q))$ .  $\square$

There are two extreme cases to take into consideration. If  $q \ll \ell$ , the dominant step is Step 2, of cost  $O(\mathcal{C}(\ell) \log(\ell))$ . When  $q \approx \ell$ , the dominant term is that of Steps 6 and 7; this is no better than standard methods, which rely on computing  $Y^p$  and  $X^p$  modulo  $f_{\ell,\lambda}(X)$  and  $Y^2 - (X^3 + AX + B)$ . In intermediates cases, implementation constants will decide.

**Modifications for  $q$  even.** For the case  $q = 2$ , we know the value of  $x \bmod 2$  (the sign of  $\lambda$ , actually) using Dewaghe's trick [9]. For  $q$  even  $> 2$ , we will work in

$$\mathcal{A} = \mathbb{F}_p[X, Y]/(Y^2 - (X^3 + AX + B), f_{\ell,\lambda}(X)),$$

extending the previous constructions to pairs of polynomials. Let  $\theta$  and  $\gamma$  be the residue classes of  $X$  and  $Y$  in

$$\mathcal{A} = \mathbb{F}_p[X, Y]/(f_{\ell, \lambda}(X), Y^2 - (X^3 + AX + B)).$$

Given  $R = (R_X(\theta), \gamma R_Y(\theta))$  and  $S = (S_X(\theta), \gamma S_Y(\theta))$ , both in  $\mathcal{A}^2$ , we now define  $R[S] = (T_X(\theta), \gamma T_Y(\theta))$ , with

$$T_X = R_X(S_X) \bmod f_{\ell, \lambda}, \quad T_Y = S_Y \cdot R_Y(S_X) \bmod f_{\ell, \lambda}.$$

Computing  $R[S]$  is thus slightly more expensive than in the case  $q$  odd, the cost being  $\mathcal{C}_2(\ell) + \mathbf{M}(\ell)$ . Remark that letting  $P = (\theta, \gamma)$  be a generator of  $V$ , if  $R = [a]P$  and  $S = [b]P$ , then  $R[S]$  equals  $[ab]P$ .

Next, we extend the definition of the traces  $\eta_0$  and  $\eta_1$  to take into account ordinates. We are led to compute

$$\eta'_i = \sum_{j=0}^{q'-1} ([h^j k^i]P)_Y,$$

where  $q'$  is now defined as  $(\ell - 1)/q$ . We thus define the vector analogue of the previous function  $T_{0, X}$ , namely

$$T_0(h, b) = \left( \sum_{j=0}^{b-1} ([h^j]P)_X, \sum_{j=0}^{b-1} ([h^j]P)_Y \right).$$

In this case, Equation (4) becomes

$$T_0(h, b + b') = T_0(h, b)[[h^{b'}]P] + T_0(h, b'),$$

where addition is performed component-wise. The algorithm to compute  $\eta'_0 = T_0(h, q')$  is the same, up to performing all computations with points, the initialization values being  $U = [h]P$ ,  $V = P$  and  $W = (0, 0)$ . The complexity is similar, but the constant in the big-Oh is larger by roughly 2. Similarly, the computation of  $\eta'_1$  is done using vectorial composition of  $T_0(h, q')$  by  $[k]P$ .

In the main algorithm, we then actually look for the minimal polynomial  $N(T)$  of degree  $q/2$  of  $(\theta^3 + A\theta + B)\eta'_0{}^2$ , which gives  $M(T)$  by  $M(T) = N(T^2)$ . The polynomial  $C(T)$  of Step 5 now has the form  $C(T) = TD(T^2)$ . It is more efficient to compute  $D(T)$  first; this is done by remarking that the relation

$$C(\eta'_0) = \eta'_1 \bmod f_{\ell, \lambda}(X)$$

gives

$$D((\theta^3 + A\theta + B)\eta'_0{}^2) = \eta'_1/\eta'_0 \bmod f_{\ell, \lambda}(X).$$

**General view of the complexity.** Summing the contributions of all  $q$  dividing  $\ell - 1$ , we see that the complexity of our approach has mainly two components. The contribution of the trace computations will be bounded by the sum of terms of the form  $O(\mathcal{C}(\ell) \log(\ell))$ . The second component is the sum of terms of the form  $O(\mathbf{M}(q) \log(p) + \mathcal{C}_{\sqrt{q}}(q))$ , corresponding to the last part of the algorithm.

If the largest prime power  $q$  dividing  $\ell - 1$  is small, we expect to be faster than standard approaches, whose complexity is dominated by the cost  $O(\mathbf{M}(\ell) \log(p))$  of computing  $Y^p$  and  $X^p$  modulo  $f_{\ell, \lambda}(X)$  and  $Y^2 - (X^3 + AX + B)$ .

### 4.3 Numerical examples

Take  $E : Y^2 = X^3 + X + 22$  over  $\mathbb{F}_p$ , with  $p = 1009$ . The prime  $\ell = 13$  is of Elkies type and the classical algorithms give the eigenfactor

$$f_{13, \lambda} = X^6 + 613X^5 + 898X^4 + 703X^3 + 487X^2 + 35X + 770.$$

We have  $\ell - 1 = 2^2 \times 3$  and  $(\mathbb{Z}/13\mathbb{Z})^*$  is generated by  $c = 2$ . Writing  $\theta = (X \bmod f_{13,\lambda}(X))$ , for  $q = 3$  and  $q' = 2$ , we find:

$$\begin{aligned}\eta_0 &= 488\theta^5 + 532\theta^4 + 926\theta^3 + 618\theta^2 + 610\theta + 518, \\ M(T) &= T^3 + 613T^2 + 128T + 550, \\ \eta_1 &= 392\theta^5 + 561\theta^4 + 685\theta^3 + 125\theta^2 + 18\theta + 294, \\ C(T) &= 524T^2 + 394T + 24, \\ T_p &= 524T^2 + 394T + 24,\end{aligned}$$

so that  $v = 1$  and  $u \equiv 1 \times q' \equiv 2 \pmod{3}$ . For  $q = 4$ , we have  $q' = 3$ , so that  $h = c^q = 3$ ,  $k = c^{q'} = 8$ . We compute

$$\eta'_0 = 56\theta^5 + 669\theta^4 + 545\theta^3 + 185\theta^2 + 62\theta + 860$$

from which we get

$$(\theta^3 + \theta + 22)\eta'_0{}^2 = 834\theta^5 + 167\theta^4 + 203\theta^3 + 121\theta^2 + 727\theta + 567$$

whose minimal polynomial is

$$N(T) = T^2 + 898T + 587$$

and  $M(T) = N(T^2)$ . Now:

$$\eta'_1 = 118\theta^5 + 972\theta^4 + 554\theta^3 + 725\theta^2 + 359\theta + 986.$$

We deduce  $D(T) = 767T + 241$  and  $C(T) = 767T^3 + 241T$ . We compute  $T^p \equiv 767T^3 + 241T \pmod{M(T)}$ , and therefore  $v = 1$ , leading to  $u = \log_2(\lambda) \pmod{4} = 3 \cdot 1 \pmod{4}$ . Combining all information leads to  $\lambda = 7$ .

## 5 Improvements

### 5.1 Combining values of $q$

The algorithm does not require the values of  $q$  to be prime powers: we just need a decomposition of  $\ell - 1$  as a product of pairwise coprime numbers. Using larger values of  $q$ 's tend to diminish the cost of the fast trace algorithm, but we have to balance with the cost of the latter steps.

It is difficult to anticipate what factorization of  $\ell - 1$  into coprimes should be used. Write  $\ell - 1 = 2^r q_1 \cdots q_s$  with  $r \geq 1$  and  $q_1 < q_2 < \cdots < q_s$  coprime prime powers. For instance, for  $\ell < 11,000$ , the domain of feasibility of SEA for the time being, we have  $r \leq 9$  and  $s \leq 4$ .

There are patterns for which we have no flexibility, for  $\ell - 1$  of the form  $2^r$ ,  $2^r q_1$  or  $2q_1 q_2$ . In the case  $\ell - 1 = 2q_1 \cdots q_s$ , we rewrite this as  $2q'_1 q'_2$  with  $q'_1$  as close as  $q'_2$  as possible.

For  $\ell - 1 = 2^r q_1 \cdots q_s$  with  $r > 1$ , the situation is more involved, since the cost for evaluating  $\eta_0$  for even  $q$  is roughly twice that for odd  $q$ . Consider the example of  $\ell - 1 = 2^r q_1 q_2$  with  $q_1 < q_2$ . If  $q_2 \ll \log(p)$ , then the dominant cost will be that of modular compositions to compute the various  $\eta_0$ 's. In this case, the cost of computing all  $\eta_0$ 's independently will approximately be  $L_1 \mathcal{C}(\ell)$ , with

$$\begin{aligned}L_1 &= 2 \log((\ell - 1)/2^r) + \log((\ell - 1)/q_1) \\ &\quad + \log((\ell - 1)/q_2) \\ &= 4 \log(\ell - 1) - \log(2^{2r} q_1 q_2).\end{aligned}$$

Combining as  $q'_1 = 2^r q_1$ ,  $q'_2 = q_2$ , we find that the cost will approximately be  $L_2 \mathcal{C}(\ell)$ , with

$$L_2 = 4 \log(\ell - 1) - \log(2^{2r} q_1^2 q_2),$$

which is smaller. On the other hand, the cost of Step 6 increases from  $(M(2^r) + M(q_1) + M(q_2)) \log(p)$  to  $(M(2^r q_1) + M(q_2)) \log(p)$ . Implementation constants will decide; see Section 6 for examples.

Remark further that for a given choice of  $q_1, q_2$ , savings are possible. Suppose for instance we are in the situation  $\ell - 1 = q_1 q_2$ , where  $q_1$  is even and  $q_2$  odd; in particular,  $q'_1 = q_2$  and  $q'_2 = q_1/2$ . For  $q_1$ , we have to compute

$$\eta'_0{}^{(q_1)} = \sum_{a \in H} ([a]P)_Y = \sum ([c^{q_1}]^i P)_Y$$

followed by

$$\eta'_1{}^{(q_1)} = ([c^{q_2}]P)_Y \cdot (\eta'_0{}^{(q_1)} \circ ([c^{q_2}]P)_X) \bmod f_{\ell, \lambda}.$$

In a symmetric way, for  $q_2$ , we need  $([c^{q_2}]P)_X$  to compute  $\eta_0{}^{(q_2)}$ , followed by

$$\eta_1{}^{(q_2)} = \eta_0{}^{(q_2)} \circ ([c^{q_1/2}]P)_X \bmod f_{\ell, \lambda}.$$

Therefore, we can amortize the costs of  $[c^{q_1}]P$  and  $[c^{q_2}]P$ , by computing  $[c^{q_1/2}]P$  first and performing a modular composition. Remark also that one can use an addition chain passing through  $\{q_1/2, q_1, q_2\}$ .

## 5.2 The case of isogeny cycles

The Abelian lift approach can be used to find  $\lambda \bmod \ell^m$  in the isogeny cycle algorithm [8, 7]. Using the techniques described therein, we can compute a factor  $f_{\ell^m, \lambda}(X)$  of the  $\ell^m$ -division polynomial  $f_{\ell^m}(X)$  (for some variants, this is a division polynomial for an intermediate curve  $E_m$  related to  $E$ ).

Let  $\varphi$  denotes Euler's totient function, so that  $\varphi(\ell^m) = \ell^{m-1}(\ell - 1)$  for prime  $\ell$ . Write  $\lambda = c^x \bmod \ell^m$  with  $c$  a generator of  $(\mathbb{Z}/\ell^m\mathbb{Z})^*$ . We already know  $x \bmod (\ell - 1)$  from the first computation; we will show how to determine  $x \bmod \ell^{m-1}$  and therefore recover modulo  $\varphi(\ell^m)$  using Chinese Remaindering.

To be quite general, and since this is one of the advantages of the isogeny cycle approach, we suppose that  $f_{\ell^m, \lambda}(X)$  is a factor of degree  $\delta \ell^{m-1}$  where  $\delta$  is the *semi-order* of  $\lambda \bmod \ell$  (if  $\varpi$  is the order, then the semi-order is  $\varpi$  if  $\varpi$  is odd and  $\varpi/2$  if it is even). As a consequence  $\delta \mid (\ell - 1)/2$ .

Let  $x_0 = x \bmod (\ell - 1)$ ,  $q = \ell^{m-1}$  and  $q' = (\ell - 1)/2$ . We let  $h = c^{qx_0} \bmod \ell^m$  and  $k = c^{q'} \bmod \ell^m$ , where  $c$  generates  $(\mathbb{Z}/\ell^m\mathbb{Z})^*$  and compute

$$\eta_0 = \sum_{j=0}^{q'-1} ([h^j]P)_X,$$

and  $\eta_1 = \eta_0[g_k]$  as usual. The rest of the algorithm is unchanged. Adapting the analysis of Proposition 4.1 and writing  $n = \delta \ell^{m-1}$ , the complexity is

$$O(\mathcal{C}(n) \log(q') + \mathbf{M}(q) \log(p) + \mathcal{C}_{\sqrt{q}}(q)).$$

For  $\ell \ll \log(p)$  (which is usually the case in real-life examples), the dominant cost is  $\mathbf{M}(q) \log(p)$ . This is faster than the classical algorithm, which computes  $Y^p \bmod f_{\ell^m, \lambda}(X)$  for a cost of  $O(\mathbf{M}(\delta q) \log(p))$ .

**Numerical example.** Consider the same curve  $E : Y^2 = X^3 + X + 22$  over  $\mathbb{F}_p$  with  $p = 1009$ . The eigenvalue  $\lambda = 7$  has semi-order 6 and the conjugate eigenvalue  $\lambda = 3$  has semi-order 3, so that, as described in [7], we use  $\lambda = 3$ . One finds:

$$\begin{aligned}
 f_{13^2, \lambda} &= X^{39} + 689X^{38} + 779X^{37} + 546X^{36} + 840X^{35} \\
 &\quad + 246X^{34} + 415X^{33} + 949X^{32} + 641X^{31} + 553X^{30} \\
 &\quad + 454X^{29} + 468X^{28} + 328X^{27} + 106X^{26} + 715X^{25} \\
 &\quad + 322X^{24} + 669X^{23} + 668X^{22} + 108X^{21} + 392X^{20} \\
 &\quad + 717X^{19} + 590X^{18} + 769X^{17} + 811X^{16} + 506X^{15} \\
 &\quad + 965X^{14} + 833X^{13} + 717X^{12} + 209X^{11} + 835X^{10} \\
 &\quad + 690X^9 + 938X^8 + 418X^7 + 670X^6 + 744X^5 \\
 &\quad + 29X^4 + 146X^3 + 914X^2 + 108X + 999, \\
 \eta_0 &= 933\theta^{38} + 84\theta^{37} + 829\theta^{36} + 660\theta^{35} + 179\theta^{34} + 974\theta^{33} \\
 &\quad + 187\theta^{32} + 581\theta^{31} + 773\theta^{30} + 84\theta^{29} + 227\theta^{28} \\
 &\quad + 631\theta^{27} + 938\theta^{26} + 852\theta^{25} + 962\theta^{24} + 153\theta^{23} \\
 &\quad + 969\theta^{22} + 128\theta^{21} + 588\theta^{20} + 670\theta^{19} + 120\theta^{18} \\
 &\quad + 52\theta^{17} + 906\theta^{16} + 654\theta^{15} + 934\theta^{14} + 897\theta^{13} \\
 &\quad + 966\theta^{12} + 827\theta^{11} + 569\theta^{10} + 864\theta^9 + 234\theta^8 + 671\theta^7 \\
 &\quad + 257\theta^6 + 160\theta^5 + 596\theta^4 + 995\theta^3 + 849\theta^2 + 548\theta \\
 &\quad + 228, \\
 M(T) &= T^{13} + 369T^{12} + 34T^{11} + 617T^{10} + 139T^9 + 579T^8 \\
 &\quad + 702T^7 + 471T^6 + 490T^5 + 740T^4 + 122T^3 + 271T^2 \\
 &\quad + 94T + 8, \\
 \eta_1 &= 564\theta^{38} + 986\theta^{37} + 952\theta^{36} + 146\theta^{35} + 135\theta^{34} \\
 &\quad + 589\theta^{33} + 960\theta^{32} + 368\theta^{31} + 544\theta^{30} + 662\theta^{29} \\
 &\quad + 142\theta^{28} + 278\theta^{27} + 894\theta^{26} + 610\theta^{25} + 29\theta^{24} \\
 &\quad + 852\theta^{23} + 433\theta^{22} + 305\theta^{21} + 197\theta^{20} + 380\theta^{19} \\
 &\quad + 713\theta^{18} + 595\theta^{17} + 760\theta^{16} + 268\theta^{15} + 834\theta^{14} \\
 &\quad + 587\theta^{13} + 444\theta^{12} + 153\theta^{11} + 846\theta^{10} + 87\theta^9 \\
 &\quad + 578\theta^8 + 975\theta^7 + 512\theta^6 + 533\theta^5 + 321\theta^4 \\
 &\quad + 315\theta^3 + 828\theta^2 + 683\theta + 457 \\
 C(T) &= 405T^{12} + 652T^{11} + 538T^{10} + 407T^9 + 679T^8 \\
 &\quad + 796T^7 + 890T^6 + 497T^5 + 339T^4 + 240T^3 \\
 &\quad + 441T^2 + 924T + 420, \\
 T_p &= 627T^{12} + 385T^{11} + 421T^{10} + 709T^9 + 117T^8 \\
 &\quad + 84T^7 + 911T^6 + 392T^5 + 97T^4 + 842T^3 \\
 &\quad + 646T^2 + 143T + 935
 \end{aligned}$$

leading to  $u = 12$  and  $\lambda = 3 \pmod{13^2}$ .

## 6 Timings

We implemented all the algorithms using Shoup's NTL library [29]. Timings were measured on an AMD 64 Processor 3400+ at 2.4GHz. Let  $p = 10^{2499} + 7131$  (the record size so far, see the NMBRTHRY mailing list) and  $\ell = 5861$  so that  $\ell - 1 = 2^2 \cdot 5 \cdot 293$ . We have the following timings in seconds:

$q$	$\eta_0$	$\eta_1$	$M(T)$	$C(T)$	$T^p$	$v$
4	15418	732	13	100	2	0
5	8491	446	17	43	10	0
293	3615	446	160	2509	3203	250

The total total time is 36800 sec. Using the algorithm of [15], computing  $Y^p \pmod{f_{\ell, \lambda}(X)}$  costs 33001 sec, recovering  $X^p$  from  $Y^p$  takes 898 sec; concluding by finding  $\lambda$  takes 3650 sec.

Using the combination strategy, replacing  $q = 4$  and  $q = 5$  by  $q = 20$ , we find a total time of 23880 sec, which now outperforms the traditional approach.

$q$	$\eta_0$	$\eta_1$	$M(T)$	$C(T)$	$T^p$	$v$
20	11374	719	26	160	36	0

We give in Figure 2 an example of testing all possible combinations for  $\ell = 421$ , so that  $\ell - 1 = 420 = 4 \times 3 \times 5 \times 7$ . The minimal time corresponds to the combination  $20 \times 21$ . Indeed, when  $\ell \ll \log(p)$  as is the case here, the dominant cost is that of computing  $T^p$ . This leads to selecting  $\ell - 1 = q'_1 q'_2$  with  $q'_1 \approx q'_2$ . As a matter of comparison, the time for computing  $Y^p \bmod f_{\ell, \lambda}(X)$  using classical approaches is 1855 sec, so that our new method is clearly superior in this case.

comb	$\eta$	$M$	$C$	$T^p$	$v$	Total
$4 \times 105$	103.16	9.94	12.00	889.67	18.81	1066.61
$12 \times 35$	105.02	5.67	8.39	328.43	1.07	489.57
$20 \times 21$	100.86	4.91	8.00	132.21	0.04	287.75
$28 \times 15$	94.45	4.59	7.62	188.67	0.22	343.93
$60 \times 7$	88.80	4.66	7.35	308.35	1.65	459.48
$84 \times 5$	91.03	4.94	7.36	483.28	2.91	625.08
$140 \times 3$	86.51	5.74	7.89	881.06	7.42	1039.45

Figure 2: Testing all combinations for  $\ell = 421$ .

We conclude in Figure 3 by some extensive benchmarks, comparing the *whole* Abelian Lift algorithm and the mere computation of

$$Y^p \bmod (Y^2 - (X^3 + AX + B, f_{\ell, \lambda}(X)))$$

for several values of  $\ell$ ; the column *fact* indicates the factorization of  $\ell - 1$ , and the column *comb* gives the combination we used. As can be seen, the new approach clearly brings a significant improvement.

## 7 Conclusions

We have presented the implementation of a new method of computing the discrete logarithm step in the SEA algorithm for counting points on elliptic curves. The method is practical in the Elkies variant of the algorithm and computes the index of an eigenvalue  $\lambda \in (\mathbb{Z}/\ell\mathbb{Z})^*$  in cyclic subgroups of the multiplicative groups separately and then assembles by the Chinese Remainder Theorem. If  $q$  are the coprime divisors modulo which the index is computed, then the Frobenius will be evaluated in extensions of degree  $q$  of  $\mathbb{F}_p$ , which is an improvement with respect to one degree  $(\ell - 1)/2$  extension in the classical variant.

A further improvement can be achieved by using elliptic curve Gauss and Jacobi sums and the identity

$$\tau(\chi)^{p-\bar{p}_p} = \chi(\lambda)^{-p},$$

with the Gauss sum

$$\tau(\chi) = \sum_{a=1}^{\ell-1} \chi(a) (g_a(\theta))$$

using the notations of Section 3. If  $\chi$  is a character of order  $q$ , then the identity above yields the index  $\log_\ell(\lambda) \bmod q$  and can be computed by means of Jacobi sums in an extension  $\mathbb{F}_p[\xi]$  generated by a  $q$ th root of unity. The degree is thus once more reduced to  $\text{rod}_p(q)$ . The method is described in [21] and becomes interesting when  $\text{ord}_p(q) \ll q$ .

The run time for exponentiation is reduced successively by the two variants; however this happens at the cost of estimation of traces. Herewith any improvement in the trace algorithm - for instance by using modular functions - would be of importance.

**Acknowledgments.** Thanks to P. Gaudry for helpful discussions on polynomial arithmetic and for suggesting the combination approach of Section 5.1; to E. Kalfoten on trace algorithms. Thanks also to A. Bostan and A. Enge for some help. One of us (FM) would like to thank the Fields Institute for its hospitality during which part of this work was achieved.

$\ell$	fact	comb	Total	$Y^P$
61	$4 \times 3 \times 5$	$12 \times 5$	29.5858	242.991
229	$4 \times 3 \times 19$	$12 \times 19$	161.262	952.416
241	$16 \times 3 \times 5$	$16 \times 15$	134.288	1011.21
277	$4 \times 3 \times 23$	$12 \times 23$	325.378	1463.9
281	$8 \times 5 \times 7$	$8 \times 35$	443.064	1434.83
313	$8 \times 3 \times 13$	$24 \times 13$	271.501	1468.8
337	$16 \times 3 \times 7$	$16 \times 21$	239.125	1569.6
349	$4 \times 3 \times 29$	$12 \times 29$	364.495	1587.61
397	$4 \times 9 \times 11$	$36 \times 11$	394.333	1764.68
409	$8 \times 3 \times 17$	$24 \times 17$	353.879	1825.81
421	$4 \times 3 \times 5 \times 7$	$12 \times 35$	513.213	1855.14
457	$8 \times 3 \times 19$	$24 \times 19$	391.897	2022.29
461	$4 \times 5 \times 23$	$20 \times 23$	415.302	1998.82
521	$8 \times 5 \times 13$	$40 \times 13$	512.104	2600.99
541	$4 \times 27 \times 5$	$20 \times 27$	541.541	2767.78
613	$4 \times 9 \times 17$	$36 \times 17$	620.348	3063.99
617	$8 \times 7 \times 11$	$56 \times 11$	638.473	3023.47
673	$32 \times 3 \times 7$	$32 \times 21$	618.209	3172.11
701	$4 \times 25 \times 7$	$28 \times 25$	722.865	3257.17
709	$4 \times 3 \times 59$	$12 \times 59$	948.196	3341.22
733	$4 \times 3 \times 61$	$12 \times 61$	992.232	3374
881	$16 \times 5 \times 11$	$16 \times 55$	1076.8	3911.75
937	$8 \times 9 \times 13$	$72 \times 13$	1085.57	4175.14
953	$8 \times 7 \times 17$	$56 \times 17$	980.541	4258.69
997	$4 \times 3 \times 83$	$12 \times 83$	1547.85	4329.35
1033	$8 \times 3 \times 43$	$24 \times 43$	1345.94	5484.26
1069	$4 \times 3 \times 89$	$12 \times 89$	1822.46	5608.17
1093	$4 \times 3 \times 7 \times 13$	$12 \times 91$	1849.94	5662.93
1213	$4 \times 3 \times 101$	$12 \times 101$	2065.43	6063.6
1237	$4 \times 3 \times 103$	$12 \times 103$	2103.97	6194.32
1277	$4 \times 11 \times 29$	$44 \times 29$	1566.86	6321.76
1289	$8 \times 7 \times 23$	$56 \times 23$	1622.27	6558
1361	$16 \times 5 \times 17$	$80 \times 17$	1716.28	6634.66
1381	$4 \times 3 \times 5 \times 23$	$12 \times 115$	2417.78	6683.73
1429	$4 \times 3 \times 7 \times 17$	$12 \times 119$	2582.24	6839.37
1453	$4 \times 3 \times 121$	$12 \times 121$	2591.88	6918.27
1481	$8 \times 5 \times 37$	$40 \times 37$	2011.45	7150.25
1489	$16 \times 3 \times 31$	$48 \times 31$	1855.47	7012.59
1549	$4 \times 9 \times 43$	$36 \times 43$	2147.66	7346.68
1613	$4 \times 13 \times 31$	$52 \times 31$	2118.97	7491.53
1657	$8 \times 9 \times 23$	$72 \times 23$	2291.12	7730.41
1709	$4 \times 7 \times 61$	$28 \times 61$	2488.44	7962.25
1741	$4 \times 3 \times 5 \times 29$	$12 \times 145$	3580.26	8004.96
1753	$8 \times 3 \times 73$	$24 \times 73$	2751.56	8091.46
1777	$16 \times 3 \times 37$	$48 \times 37$	2471.74	8106.22
5861	$4 \times 5 \times 293$	$20 \times 293$	22215.7	30641.1
5981	$4 \times 5 \times 13 \times 23$	$20 \times 299$	22739.5	30551.3
8009	$8 \times 7 \times 11 \times 13$	$56 \times 143$	30402.8	38457.1

Figure 3: Abelian lift vs. exponentiation

## References

- [1] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime (II). Available at <http://listserv.nodak.edu/archives/nbrthry.html>, July 1992.
- [2] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume 265 of *London Math. Soc. Lecture Note Ser.* Cambridge University Press, 1999.
- [3] A. Bostan, F. Morain, B. Salvy, and É. Schost. Fast algorithms for computing isogenies between elliptic curves, 2006.
- [4] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. ACM*, 25(4):581–595, 1978.
- [5] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.
- [6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.
- [7] J.-M. Couveignes, L. Dewaghe, and F. Morain. Isogeny cycles and the Schoof-Elkies-Atkin algorithm. Research Report LIX/RR/96/03, LIX, Apr. 1996. Available at <http://www.lix.polytechnique.fr/Labo/Francois.Morain/>.
- [8] J.-M. Couveignes and F. Morain. Schoof’s algorithm and isogeny cycles. In L. Adleman and M.-D. Huang, editors, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 43–58. Springer-Verlag, 1994. 1st Algorithmic Number Theory Symposium - Cornell University, May 6-9, 1994.
- [9] L. Dewaghe. Remarks on the Schoof-Elkies-Atkin algorithm. *Math. Comp.*, 67(223):1247–1252, July 1998.
- [10] N. D. Elkies. Explicit isogenies. Draft, 1992.
- [11] N. D. Elkies. Elliptic and modular curves over finite fields and related computational issues. In *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkin*, volume 7 of *AMS/IP Studies in Advanced Mathematics*, pages 21–76. AMS, International Press, 1998.
- [12] A. Enge. Computing modular polynomials in quasi-linear time, 2006.
- [13] J. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
- [14] J. Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Comput. Complexity*, 2(3):187–224, 1992.
- [15] P. Gaudry and F. Morain. Fast algorithms for computing the eigenvalue in the Schoof-Elkies-Atkin algorithm. In *ISSAC’06*, pages 109–115. ACM Press, 2006.
- [16] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Math. Comput.*, 67(223):1179–1197, 1998.
- [17] S. Lang. *Elliptic Functions*. Springer GTM **112**, 1986.
- [18] H. W. J. Lenstra. Galois theory and primality testing. In *Orders and their applications*, volume 1142 of *Lecture Notes in Mathematics*, pages 1–21. Springer Verlag, 1985.
- [19] M. Maurer and V. Müller. Finding the eigenvalue in Elkies’ algorithm. *Experiment. Math.*, 10(2):275–285, 2001.
- [20] P. Mihăilescu. Cyclotomy primality proofs and their certificates. *Mathematica Goettingensis*, 2006.
- [21] P. Mihăilescu. Elliptic curve Gauss sums and counting points. *Mathematica Goettingensis*, 2006.
- [22] 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.
- [23] C. Pascal and É. Schost. Change of order for bivariate triangular sets. In *ISSAC’06*, pages 277–284. ACM Press, 2006.
- [24] F. Rouillier. Solving zero-dimensional systems through the Rational Univariate Representation. *Appl. Alg. in Eng. Comm. Comput.*, 9(5):433–461, 1999.
- [25] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.
- [26] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod  $p$ . *Math. Comp.*, 44(170):483–494, 1985.
- [27] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.
- [28] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symbolic Comput.*, 17:371–391, 1994.
- [29] V. Shoup. A new polynomial factorization algorithm and its implementation. *J. Symbolic Comput.*, 20:363–397, 1995.
- [30] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *ISSAC’99*, pages 53–58. ACM Press, 1999.