Étude et implémentation de l'algorithme de Schoof–Elkies–Atkin

Jean Kieffer

26 mars 2018

Table des matières

1 Introduction

Les courbes elliptiques sont devenues incontournables en cryptographie moderne. Ces groupes algébriques, lorsque leur cardinal est premier ou quasiment premier, sont utilisés notamment car le problème du logarithme discret y est réputé difficile.

Cependant, pour un entier C donné, on ne connaît pas de méthode satisfaisante pour trouver une courbe elliptique dont le nombre de points est C. Le plus souvent, on tire donc des courbes au hasard jusqu'à en trouver une de cardinal premier.

De ce fait, l'algorithme de Schoof–Elkies–Atkin (SEA), qui permet de calculer le cardinal d'une courbe, est une brique indispensable à l'existence de la cryptographie sur courbes elliptiques. On se place dans le cas d'un corps fini de grande caractéristique (au moins $p > 1000$) : dans le cas contraires, des méthodes p-adiques ou cohomologiques sont plus rapides.

L'objet de ce document est l'étude algorithmique et pratique de l'algorithme SEA. Cette étude a fait l'objet d'une implémentation en langage C, basée sur les bibliothèques de calcul formel Flint [HJP17] ou NTL [Sho17]. On porte une attention particulière à

— détailler précisément et exactement les algorithmes implémentés ;
— donner les tests permettant de s'assurer de la correction des calculs effectués (qui ne sont pas présents dans le programme final, mais peuvent être activés en compilant le programme en mode « debug ») ;
— proposer des améliorations qui ne sont pas actuellement implémentées, mais qui peuvent être l'objet de modifications futures.

Un des objectifs de ce projet est de parvenir à une implémentation qui puisse être spécialisée pour les cas cryptographiquement intéressants de corps dont l'ordre mesure 256 ou 512 bits, et qui soit compétitive avec les logiciels de calcul formel PARI, Magma ou encore Sage. Par ailleurs, on donne la possibilité de stopper le calcul lorsqu'un facteur du cardinal dépassant une certaine valeur est détecté.

Dans la section 2, on rappelle les résultats élémentaires sur les courbes elliptiques utilisés par la suite. La section 3 présente les grandes lignes de l'algorithme, qui est détaillé ensuite en 4, 5, 6, 7. L'analyse expérimentale des performances est faite en section 8. Enfin, la section 9 tient lieu de documentation de l'implémentation.

On adopte les notation suivantes : $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{C}$ sont les ensembles de nombres habituels, $\mathbb{F}_q$ désigne le corps fini de cardinal q, $\mathbb{Z}_\ell$ désigne l'anneau des entiers ℓ-adiques, et une barre dénote la clôture algébrique. Les courbes elliptiques sont désignées par la lettre E, et $[n]$ désigne la multiplication scalaire par n sur E. Le cardinal d'un ensemble S est noté $\mathrm{Card}(S)$. Enfin, lorsque A est un ensemble algébrique et k un corps, $A(k)$ désigne l'ensemble des k-points de A, c'est à dire des points de A à coordonnées dans k.

2 Prérequis généraux sur les courbes elliptiques

Dans cette section, on rappelle les propriétés des courbes elliptiques sur un corps fini qui interviennent au cours de l'algorithme SEA, ainsi que les notations utilisées dans ce document. On renvoie aux preuves contenues dans le livre de Silverman [Sil86].

2.1 Endomorphisme de Frobenius et théorie de Hasse

On fixe une courbe elliptique E sur un corps fini $\mathbb{F}_q$. Dans ce document, on écrira toujours une courbe elliptique sous la forme de Weierstrass

$$E \ : \ y^2 = x^3 + ax + b$$

Rappelons que l'on travaille en grande caractéristique, donc E peut toujours s'écrire sous cette forme [Sil86, III.1]).

L'endomorphisme de Frobenius

$$\pi_E \ : \ (x, y) \mapsto (x^q, y^q)$$

est l'objet fondamental dans l'algorithme SEA, ce qui est justifié par le résultat suivant.

Théorème 2.1 (Hasse)**.** *Il existe un unique entier $t \in \mathbb{Z}$, appelé* trace du Frobenius *sur la courbe E, tel que l'on ait*

$$\pi_E^2 - t\pi_E + q = 0$$

dans l'anneau $\mathrm{End}(E)$ *des endomorphismes de E. Il vérifie l'inégalité (dite* borne de Hasse*)*

$$|t| \leq 2\sqrt{q}.$$

Si α et β désignent les deux racines (complexes conjuguées) du polynôme $X^2 - tX + q$ dans $\mathbb{C}$, le nombre de points de E sur $\mathbb{F}_{q^n}$ est

$$\mathrm{Card}(E(\mathbb{F}_{q^n})) = q + 1 - \alpha^n - \beta^n$$

pout tout $n \geq 1$. En particulier, le nombre de points de E sur $\mathbb{F}_q$ est

$$\mathrm{Card}(E(\mathbb{F}_q)) = q + 1 - t.$$

Démonstration. Voir [Sil86], V.1 (théorème 1.1) et V.2 (théorème 2.4). □

2.2 Sous-groupes d'une courbe elliptique

Le Frobenius π_E étant de degré q, le manipuler directement engendre un coût de l'ordre de q, c'est à dire exponentiel en $\log(q)$. Dans l'algorithme SEA, on va restreindre π_E à des sous-groupes de la courbe E pour contourner ce problème.

Lorsque l'on parle ici d'un sous-groupe G de E, on se réfère toujours à un sous-groupe fini de $E(\overline{\mathbb{F}_q})$: les éléments de G sont donc définis sur une clôture algébrique de $\mathbb{F}_q$. À un sous-groupe G, on associe un polynôme défini de la manière suivante :

— Lorsque G ne contient pas d'éléments de 2-torsion, on regroupe les éléments de G par paire $(P_i, -P_i)$ pour $1 \leq i \leq n$. Rappelons que l'application $P \mapsto -P$ sur E s'écrit

$$(x, y) \mapsto (x, -y).$$

On représente alors G par le polynôme $K = \prod_{i-1}^{n}(X - x(P_i))$. C'est un polynôme unitaire de degré $\mathrm{Card}(G)/2$.

— Dans le cas général, pour chaque point de 2-torsion $(x, 0)$ de G, on ajoute un facteur $(X - x)$ à ce polynôme.

On appelle souvent K le *polynôme de noyau* de G. La raison de cette représentation est donnée par la proposition suivante.

Proposition 2.2. *Soit G un sous-groupe de E* sans *point de 2-torsion. Alors, avec la notation ci-dessus, l'anneau de coordonnées de G est le quotient*

$$\mathbb{F}_q[X, Y]/(Y^2 = X^3 + aX + b,\ K(X) = 0).$$

Autrement dit, pour qu'une certaine égalité algébrique soit vraie sur tout G, il faut et suffit qu'elle soit vraie pour le point générique *de G :*

$$P = (X, Y)$$

en réduisant les deux coordonnées modulo $Y^2 = X^3 + aX + b$ et $K(X) = 0$.

Démonstration. C'est essentiellement une reformulation du théorème des zéros de Hilbert : voir [Nek16], théorème 7.5, particulièrement d). L'idéal

$$(K(X),\ Y^2 - X^3 - aX - b)$$

de $\mathbb{F}_q[X, Y]$ est radical, car K est sans facteur carré et premier à $X^3 + aX + b$ par hypothèse. De plus l'ensemble d'annulation de cet idéal dans $\overline{\mathbb{F}_q}$ est exactement G. □

Cette proposition sera largement utilisée tout au long de l'algorithme SEA. Attention ! Cela *n'est pas vrai* lorsque G contient un point de 2-torsion. Le polynôme de noyau K défini ci-dessus n'est pas le bon. En quelque sorte,

chaque point de 2-torsion est compté deux fois, mais cela est inévitable si l'on souhaite uniquement manipuler des polynômes univariés.

On dit qu'un sous-groupe G est rationnel, ou défini sur $\mathbb{F}_q$, G est globalement stable par le Frobenius π_E. Cela implique que le polynôme K est à coefficients dans $\mathbb{F}_q$, mais cela n'implique pas que les points de G soient eux-mêmes définis sur $\mathbb{F}_q$ (c'est le cas si G est stable par π_E point par point).

2.3 Isogénies

Dans l'exécution de l'algorithme SEA, on est amené à manipuler et à calculer des isogénies entre courbes elliptiques. On se donne un entier ℓ que l'on suppose premier à q (en pratique, on aura toujours $\ell \ll q$).

Une isogénie $\phi : E \to E'$ de degré ℓ [Sil86, III.4] peut être décrite comme un morphisme de groupes

$$E(\overline{\mathbb{F}_q}) \to E'(\overline{\mathbb{F}_q})$$

donné par des formules algébriques, et dont le noyau est fini de cardinal ℓ.

Proposition 2.3. *Lorsque les courbes sont écrites sous forme de Weierstrass, on a*

$$\phi(x, y) = (\phi_x(x), cy\phi'_x(x))$$

pour une certaine fraction rationnelle ϕ_x et un certain $c \in \overline{\mathbb{F}_q}$ non nul.

Démonstration. Sur E et E', on considère les différentielles invariantes $\frac{dx}{y}$ et $\frac{dx'}{y'}$ respectivement. Comme ces différentielles forment un espace de dimension 1 par le théorème de Riemann–Roch [Sil86, II.5], le pullback de $\frac{dx'}{y'}$ par ϕ est nécessairement $c\frac{dx}{y}$ pour un certain $c \in \overline{\mathbb{F}_q}$, ce qui donne le résultat. □

On dira que ϕ est *normalisée*, ou que les équations de E et E' sont normalisées, si l'on a $c = 1$. Si l'équation de E est fixée et si le j-invariant $j(E')$ est donné (ce qui est équivalent à la donnée de E' à isomorphisme près sur $\overline{\mathbb{F}_q}$), alors il existe une unique équation de Weierstrass pour E' qui est normalisée. On peut le constater en regardant les isomorphismes possibles de E' [Sil86, III.1].

Les isogénies sont intimement liées aux sous-groupes de E : si G est un sous-groupe fini de E, alors il existe une unique isogénie $E \to E'$ de noyau G, à isomorphisme de E' près.

On dit que ϕ est rationnelle, ou définie sur $\mathbb{F}_q$, si la fraction rationnelle ϕ_x est à coefficients dans $\mathbb{F}_q$ et $c \in \mathbb{F}_q$. C'est équivalent à demander que le noyau de ϕ soit défini sur $\mathbb{F}_q$ (modulo un bon choix de tordue pour E'). On rappelle enfin que ϕ admet toujours une duale $\hat{\phi} : E' \to E$ de même degré vérifiant

$$\phi \circ \hat{\phi} = [\ell]_{E'}, \quad \hat{\phi} \circ \phi = [\ell]_E.$$

(voir [Sil86, III.6]).

Deux courbes liées par une isogénie de degré ℓ sont liées par certaines équations polynomiales, dites *équations modulaires de niveau* ℓ. Ce nom s'explique par le lien fort entre courbes elliptiques et formes modulaires lorsque l'on travaille sur le corps $\mathbb{C}$. Par exemple, le *polynôme modulaire classique* Φ_ℓ est un polynôme bivarié qui vérifie

$$\Phi_\ell(j(E), j(E')) = 0$$

si et seulement si E et E' sont liées par une isogénie de degré ℓ dont le noyau est cyclique (c'est toujours le cas si ℓ est premier). C'est un polynôme à coefficients entiers, dont les propriétés sont valables sur tous les corps finis, au moins lorsque ℓ est premier à la caractéristique. Pour le lien entre courbes elliptiques sur $\mathbb{C}$ et fonctions modulaires, voir [Sil86, VI] ; pour une définition du polynôme Φ_ℓ, voir [Cox89, 11.C].

2.4 Torsion et polynômes de division

Soit $\ell \geq 2$ un entier premier à la caractéristique. Un exemple important de sous-groupe de E est le sous-groupe de ℓ-torsion $E[\ell]$ formé des points $P \in E(\overline{\mathbb{F}}_q)$ vérifiant $[\ell]P = 0$. On a le résultat suivant :

Proposition 2.4. *$E[\ell](\overline{\mathbb{F}}_q)$ est isomorphe (non canoniquement) à $(\mathbb{Z}/\ell\mathbb{Z})^2$. En particulier, c'est un sous-groupe de cardinal ℓ^2.*

Démonstration. Voir [Sil86], corollaire III.6.4. □

Le polynôme de noyau de $E[\ell]$ est (aux points de 2-torsion près) le ℓ-ième polynôme de division de la courbe E : il s'agit du polynôme bivarié $f_{\ell,E}$ apparaissant dans l'égalité

$$[\ell](x, y) = \left(\frac{A(x, y)}{f_{\ell,E}(x, y)^2}, \frac{B(x, y)}{f_{\ell,E}(x, y)^3} \right)$$

pour certains polynômes A et B (que l'on peut également exprimer en fonction des polynômes de division) : voir [Sil86], exercice 3.7.

Un sous-groupe de E de cardinal ℓ est nécessairement constitué de points de ℓ-torsion, donc est un sous-groupe de $E[\ell]$. Les polynômes de noyau de ces sous-groupes sont des facteurs du polynôme de division.

Lorsque ℓ est premier, il est souvent intéressant de considérer $E[\ell]$ comme un espace vectoriel de dimension 2 sur le corps $\mathbb{F}_\ell$ vu l'isomorphisme ci-dessus. Les sous-groupes de E de cardinal ℓ sont alors les droites vectorielles de cet espace : ils sont au nombre de $\ell + 1$. Le fait que ces sous-groupes soient ou non définis sur $\mathbb{F}_q$ est alors lié au comportement de π_E en tant qu'endomorphisme de $E[\ell]$: les sous-groupes rationnels de cardinal ℓ sont les droites propres pour π_E. On peut mettre cette remarque en relation avec la proposition suivante :

Proposition 2.5. *Le polynôme caractéristique de π_E en tant qu'endomorphisme de $E[\ell]$ est $X^2 - tX + q$, où t est la trace du Frobenius de E introduite à la section 2.1.*

Démonstration. Le module de Tate $T_\ell(E)$ [Sil86, III.7] est un $\mathbb{Z}_\ell$-espace vectoriel de dimension 2 dont le quotient par ℓ est $E[\ell]$. L'injection

$$\mathrm{End}(E) \otimes \mathbb{Z}_\ell \to \mathrm{End}(T_\ell(E))$$

montre que le polynôme minimal de π_E sur $T_\ell(E)$ est $X^2 - tX + q$: c'est donc également son polynôme caractéristique. Celui-ci étant compatible avec le quotient par ℓ, on en déduit le résultat. □

Le comportement de la matrice 2×2 à coefficients dans $\mathbb{F}_\ell$ qui représente π_E est donc lié à la valeur du discriminant $t^2 - 4q$ modulo ℓ. Par exemple, les trois énoncés suivants sont équivalents :

— $t^2 - 4q$ est un carré modulo ℓ,
— La matrice du Frobenius sur $E[\ell]$ est trigonalisable dans $\mathbb{Z}/\ell\mathbb{Z}$,
— La courbe E admet un sous-groupe rationnel d'ordre ℓ.

3 Le principe de l'algorithme SEA

Schoof [Sch85] propose en 1985 le premier algorithme permettant de calculer le nombre de points d'une courbe elliptique sur un corps fini $\mathbb{F}_q$ en un temps polynomial en $\log q$. Aujourd'hui, l'algorithme de Schoof et ses améliorations sont des prérequis nécessaires à l'existence même de la cryptographie sur courbes elliptiques.

Tout d'abord, on présente la méthode originale de Schoof, puis on décrira les améliorations proposées par Atkin et Elkies [Atk92, Sch95, Elk97] qui rendent cette algorithme praticable, ainsi que la mise en œuvre globale de l'algorithme SEA. L'étude algorithmique détaillée des différents ingrédients est faite dans la suite de ce document.

3.1 L'algorithme de Schoof

Le principe de l'algorithme de Schoof est de calculer le nombre de points d'une courbe elliptique modulo toute une série de petits nombres premiers, et de reconstruire le résultat grâce au théorème chinois.

Plus précisément, soit $E/\mathbb{F}_q$ une courbe elliptique. D'après la section 2, la trace t du Frobenius π_E est l'unique entier vérifiant

$$\pi_E^2 - t\pi_E + q = 0 \tag{1}$$

et le nombre de points de E sur $\mathbb{F}_q$ est

$$\mathrm{Card}(E(\mathbb{F}_q)) = q + 1 - t.$$

Fixons un entier auxiliaire ℓ premier à q, et notons t_ℓ (resp. q_ℓ) l'entier compris entre 0 et $\ell - 1$ tel que $t_\ell = t \mod \ell$ (resp. $q_\ell = q \mod \ell$). Sur le sous-groupe de ℓ-torsion $E[\ell]$, on a toujours l'égalité (1). Comme l'endomorphisme $[\ell]$ y est nul, on en déduit

$$\pi_E^2 - t_\ell\pi_E + q_\ell = 0 \quad \text{sur } E[\ell]. \tag{2}$$

De plus, t_ℓ est l'unique entier compris entre 0 et $\ell - 1$ pour lequel l'égalité (2) est vérifiée. En effet, si $n\pi_E = 0$ sur $E[\ell]$, on a nécessairement $n = 0 \mod \ell$ car π_E est un automorphisme de $E[\ell]$.

L'équation (2) permet donc de déterminer uniquement t_ℓ. Selon 2.2, si P désigne le point générique de $E[\ell]$, il est équivalent de demander que

$$\pi_E^2(P) - [t_\ell]\pi_E(P) + [q_\ell]P = 0,$$

ce qui est directement calculable par des opérations polynomiales *modulo* le ℓ-ième polynôme de division $f_{\ell,E}$. Par exemple, une méthode naïve consiste à tester toutes les valeurs possibles de t_ℓ, de 0 à $\ell - 1$, jusqu'à trouver une valeur pour laquelle l'égalité (2) est vérifiée.

Afin de calculer $\mathrm{Card}(E(\mathbb{F}_q))$, on calcule la valeur de t_{ℓ_i} pour des entiers $\ell_i,\ i \in I$ premiers entre eux tels que

$$\prod_{i \in I} \ell_i > 4\sqrt{q}. \tag{3}$$

Les bornes de Hasse $|t| \leq 2\sqrt{q}$ permettent alors de déterminer la valeur de t. Par exemple, on peut choisir pour les ℓ_i la suite des petits nombres premiers. Toutefois, en pratique, le degré élevé (en ℓ_i^2) des polynômes de division rend cet algorithme inopérant.

3.2 Les améliorations d'Elkies et Atkin

Les améliorations apportées par Elkies et Atkin à l'algorithme de Schoof peuvent être grossièrement résumées de la façon suivante : *pour obtenir des informations sur la trace modulo ℓ, il suffit de calculer modulo des polynômes de degré ℓ*. Il s'agit d'éviter le surcoût de la méthode de Schoof dû au degré élevé des polynômes de division. Dans ce qui suit, on fixe un nombre premier ℓ premier à q.

On peut étudier la structure de $E[\ell]$ via une équation modulaire de niveau ℓ, comme le montre le résultat suivant (on remarquera le lien avec la section 2.4).

Proposition 3.1. *Soit $E/\mathbb{F}_q$ une courbe elliptique ordinaire, de j-invariant distinct de 0 et 1728, dont la trace du Frobenius est notée t. Soit $\Phi_\ell(X, Y)$ le polynôme modulaire classique de niveau ℓ, et soit $Q_1 \cdots Q_s$ la décomposition en facteurs irréductibles de $\Phi_\ell(j(E), Y)$ dans $\mathbb{F}_q[X]$.*

Alors les possibilités pour les degrés des Q_i sont les suivantes :

1. $(1, \ldots, 1)$. *Dans ce cas, $t^2 - 4q = 0 \mod \ell^2$.*
2. $(1, \ell)$. *Dans ce cas, $t^2 - 4q = 0 \mod \ell$.*
3. $(1, 1, r, \ldots, r)$ *avec $r > 1$. Dans ce cas, $t^2 - 4q$ est un carré modulo ℓ. La courbe E admet deux sous-groupes cycliques d'ordre ℓ définis sur $\mathbb{F}_q$. Les valeurs propres du Frobenius sur ces deux sous-groupes sont les racines de $X^2 - tX + q = 0$ dans $\mathbb{Z}/\ell\mathbb{Z}$.*
4. $(r, \ldots, r)$ *avec $r > 1$. Dans ce cas, $t^2 - 4q$ n'est pas un carré modulo ℓ, et l'on a*

$$t^2 = q(\zeta + \zeta^{-1})^2 \mod \ell$$

pour une certaine racine primitive r-ième de l'unité $\zeta \in \overline{\mathbb{F}}_\ell$. Les deux valeurs propres du Frobenius dans $\mathbb{F}_{\ell^2}$ sont λ et $\zeta\lambda$ pour un certain $\lambda \in \mathbb{F}_{\ell^2}$. De plus, le nombre de facteurs irréductibles s vérifie

$$(-1)^s = \left(\frac{q}{\ell}\right), \tag{4}$$

où $\left(\frac{\cdot}{\cdot}\right)$ désigne le symbole de Jacobi.

Démonstration. Voir [Sch95], proposition 6.2. (Tout découle presque immédiatement de la discussion en 2.4). Notons que Schoof ne montre pas l'égalité modulo ℓ^2 dans le cas 1, mais seulement modulo ℓ. On regarde l'action de π_E sur le module de Tate $T_\ell(E)$ en quotientant au niveau de ℓ^2 : la matrice est de la forme

$$\begin{pmatrix} \alpha + \ell x_1 & \ell x_2 \\ \ell x_3 & \alpha + \ell x_4 \end{pmatrix}$$

modulo ℓ^2, dont le polynôme caractéristique est

$$X^2 - (2\alpha + \ell\beta)X + (\alpha^2 + \ell\alpha\beta)$$

en notant $\beta = x_1 + x_4$. Son discriminant est $\ell^2\beta^2$, c'est à dire 0 modulo ℓ^2, or ce discriminant est exactement $t^2 - 4q$ (voir la proposition 2.5). □

La méthode d'Elkies. Elle s'applique dans le cas 3 de la proposition ci-dessus. Dans cette première description, on suppose ℓ impair.

L'idée est de calculer un sous-groupe G de la courbe E, d'ordre ℓ et défini sur $\mathbb{F}_q$. Ce sous-groupe est décrit par un polynôme de degré $\frac{\ell-1}{2}$. On peut alors remplacer avantageusement l'égalité (2) par

$$\pi_E^2 - t_\ell \pi_E + q_\ell = 0 \quad \text{sur } G,$$

ce qui permet de calculer modulo un polynôme de degré $\frac{\ell-1}{2}$ au lieu de $\frac{\ell^2-1}{2}$.

D'autre part, on peut éviter de calculer π_E^2 en cherchant directement la valeur propre v du Frobenius sur le sous-espace propre G : elle vérifie

$$\pi_E = v \quad \text{sur } G.$$

Le produit des deux valeurs propres est q, on récupère donc la trace t_ℓ via $t_\ell = v + \frac{q}{v} \mod \ell$.

Pour utiliser cette méthode, il faut en premier lieu calculer G. Utiliser un algorithme de factorisation sur $f_{\ell,E}$ annihile le gain obtenu. Pour éviter ce problème, on calcule, à l'aide de l'équation modulaire, une isogénie de degré ℓ dont G est le noyau (voir section 4).

Enfin, la méthode d'Elkies admet une généralisation naturelle en remplaçant G par le noyau d'une composition de plusieurs isogénies. G est alors un sous-groupe cyclique de E d'ordre ℓ^r, et son polynôme de noyau est un facteur du polynôme de division $f_{\ell^r,E}$. L'étude de π_E sur G permet alors de calculer la valeur de t modulo ℓ^r. Cette version étendue est intéressante également dans les cas 1 et 2, mais n'est pas toujours applicable. On renvoie à la section 4 pour une description algorithmique détaillée.

La méthode d'Atkin. Bien qu'elle puisse s'appliquer dans tous les cas de la proposition ci-dessus, on l'appliquera uniquement dans le cas 4.

On ne calcule pas directement la quantité t_ℓ, mais un ensemble (plus ou moins grand) d'éléments de $\mathbb{Z}/\ell\mathbb{Z}$ donné par la proposition ci-dessus qui contient t_ℓ. Une fois la borne (3) atteinte, un crible à la Shanks permet de calculer la bonne valeur de t parmi toutes les possibilités offertes par le théorème chinois (voir section 7). Pour calculer la liste des valeurs possibles de t_ℓ, il faut tout d'abord calculer le degré de décomposition r, puis manipuler des racines r-ièmes de l'unité dans $\overline{\mathbb{F}}_\ell$ (voir section 5). Ces idées remontent aux années 1980 [Atk88].

Du fait de l'utilisation d'équations modulaires et de la proposition 3.1, les améliorations d'Elkies et d'Atkin à l'algorithme de Schoof ne peuvent pas être utilisées pour les courbes supersingulières, ou isogènes à une courbe dont le j-invariant est égal à 0 ou 1728. On verra en 3.5 comment détecter et traiter ces courbes particulières.

3.3 Utilisation des équations modulaires

Dans l'algorithme SEA, on va utiliser des équations modulaires comme le polynôme Φ_ℓ introduit à la section 2.3. On choisit de stocker ces équations sous forme de polynômes bivariés à coefficients entiers, qui ont été préalablement calculés. On réduit ensuite leurs coefficients dans le corps fini qui nous intéresse.

Cela pose un problème de mémoire : le polynôme Φ_ℓ est un polynôme bivarié de degré $\ell+1$ en chaque variable, qui n'est pas creux, et dont les coefficients croissent de manière exponentielle en ℓ. Les bases de données les plus fournies comme [Sut10] ne dépassent pas $\ell = 300$, ce qui n'est pas suffisant lorsque q mesure 512 bits.

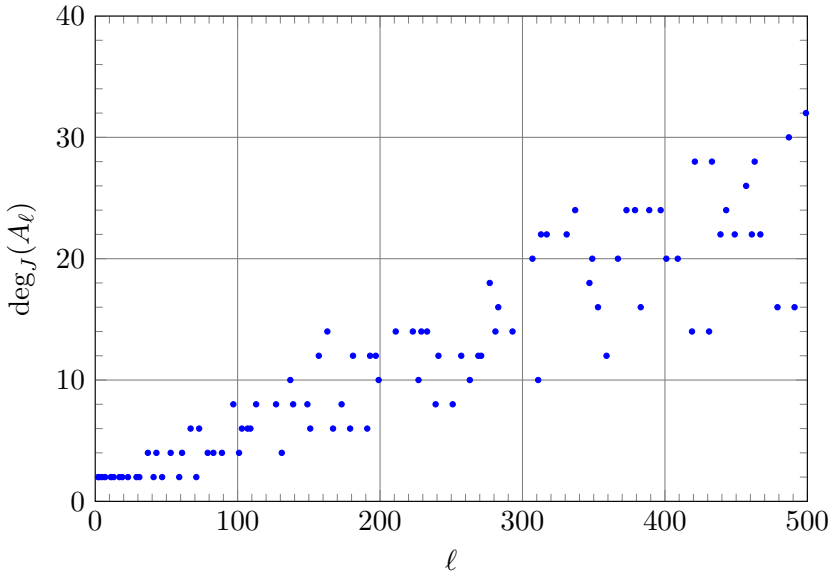
Pour pallier ces inconvénients, on utilise un autre type d'équations modulaires appelées *polynômes modulaires d'Atkin* et notées A_ℓ. Plutôt que de relier les j-invariants des deux courbes isogènes (ce qui donne le polynôme Φ_ℓ), les polynôme A_ℓ relie le j-invariant d'une courbe et une autre quantité traditionnellement notée f. Dans un corps fini (au moins lorsque ℓ est distinct de la caractéristique), ils ont la propriété suivante : si E et E' sont reliées par une isogénie rationnelle de degré ℓ, alors il existe un élément $f \in \mathbb{F}_q$ tel que l'on ait

$$A_\ell(j(E), f) = 0 \quad \text{et} \quad A_\ell(j(E'), f) = 0. \tag{5}$$

Le degré de $A_\ell(J, F)$ en F est égal à $\ell+1$, et le polynôme modulaire d'Atkin partage certaines propriétés du polynôme modulaire classique : par exemple, la factorisation de $A_\ell(j(E), F)$ et celle de $\Phi_\ell(j(E), Y)$ sont du même type. Par conséquent, l'analogue de la proposition 3.1 pour A_ℓ reste vraie.

Si le degré de A_ℓ en J est égal à 2, l'équation (5) permet de déterminer uniquement $j(E')$ une fois une racine de A_ℓ connue. Bien que cela soit le

FIGURE 1 – $\deg_J(A_\ell)$ en fonction de ℓ



cas lorsque ℓ est petit ($\ell \leq 31$ ou $\ell \in \{41, 47, 59, 71\}$), ce n'est pas vrai en général, et des valeurs parasites peuvent apparaître : on peut trouver des éléménts $h \in \mathbb{F}_q$ tels que $A_\ell(j(E), f) = A_\ell(h, f) = 0$ mais qui ne sont pas le j-invariant d'une courbe isogène à E. Plus le degré de A_ℓ en J est élevé et plus l'existence de tels parasites est probable (voir la figure 1).

Comme on le voit, le degré en J reste plutôt faible, ce qui représente un gain de place important par rapport aux polynômes classiques Φ_ℓ. Il se trouve également que les coefficients de A_ℓ croissent moins vite (voir figure 2) : les stocker jusqu'à $\ell = 500$ ne pose pas de problème. Cette valeur semble suffisante pour traiter l'immense majorité des courbes lorsque $\log_2(q) \simeq 512$.

En pratique, on utilisera les deux types d'équations modulaires : Φ_ℓ pour les petits premiers, car cette équation est plus simple d'utilisation, et A_ℓ pour les autres. On stocke ces équations dans des fichiers dont chaque ligne représente un coefficient : par exemple, la ligne

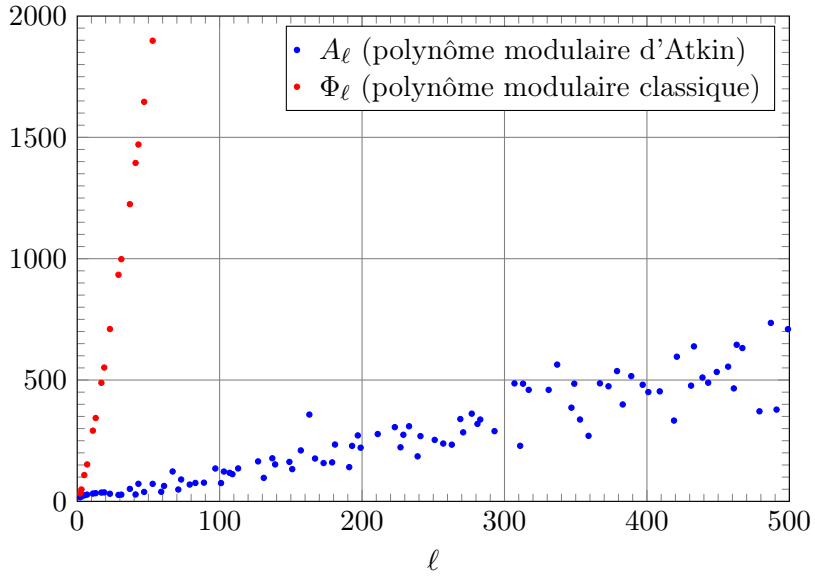
```
1 7 -16023299
```

dans le fichier décrivant A_{17} signifie que le coefficient de J^1F^7 est -16023299. La taille du fichier décrivant A_{499} est 2,7 Mo.

Lors de la lecture de ce fichier, on réduit les coefficients dans le corps fini, et on les stocke dans un tableau qui est ensuite utilisé autant de fois que nécessaire. Pour évaluer une équation modulaire ou ses dérivées en une (ou deux) coordonnées, on utilise la méthode de Horner.

Une amélioration possible. *On dispose des polynômes Φ_ℓ et A_ℓ précalculés jusqu'à respectivement $\ell = 113$ et $\ell = 500$. Ces valeurs semblent suffisantes*

FIGURE 2 – Logarithme du plus grand coefficient en fonction de ℓ



pour un corps fini de 512 bits, mais ne le sont pas pour un corps de 1024 bits. Pour traiter ces cas, il est nécessaire de trouver ou de savoir calculer des équations modulaires supplémentaires.

Un algorithme de Sutherland [BLS12] permet de calculer directement le polynôme $\Phi_\ell(j, Y)$ lorsque $j \in \mathbb{F}_q$ est donné en un temps quasi-linéaire. Asymptotiquement, il s'agit de la meilleure solution, et elle permet de se dispenser des parasites apportés par les équations A_ℓ, mais le calcul doit être refait à chaque utilisation. Quel est le seuil à partir duquel cette solution devient meilleure que le stockage de polynômes A_ℓ précalculés en tant que polynômes bivariés sur $\mathbb{Z}$?

Une autre solution consisterait à utiliser des équation modulaires utilisant la fonction de Weber, encore plus compactes que les A_ℓ, que certaines bases de données stockent jusque $\ell = 3000$.

3.4 Le traitement d'un nouveau premier

Lors de l'algorithme SEA, on peut choisir de consacrer plus ou moins de temps à un nombre premier auxiliaire ℓ donné. Par exemple, si l'on applique la méthode d'Elkies, on peut tenter de calculer la trace modulo ℓ^r pour r supérieur à 1. Lorsque l'on peut appliquer la méthode d'Atkin, on pourrait choisir la méthode originale de Schoof, plus coûteuse mais qui apporte une information plus précise.

Par conséquent, on peut proposer deux manières de programmer l'algorithme SEA :

— Une manière optimale : tant que la borne (3) n'est pas atteinte, on

évalue le premier ℓ le plus avantageux à étudier (soit choisir un nouveau premier, soit consacrer plus de ressources à un ℓ déjà étudié).

— Une manière moins optimale : on traite les nombres premiers dans l'ordre croissant, avec une quantité de ressources prédéfinie à consacrer à chacun d'eux. Tant que la borne (3) n'est pas atteinte, on étudie des ℓ de plus en plus grands.

La seconde méthode a l'avantage de la simplicité, et est la méthode choisie dans cette implémentation. Toutefois, elle présente l'inconvénient de devoir prédéfinir la quantité de calculs à fournir pour chaque ℓ. Concrètement, on définit une notion d'*exposant idéal* : pour chaque ℓ, il s'agit de l'entier $r \geq 1$ tel que l'on est prêt à manipuler des polynômes de degré au plus ℓ^r. Des mesures expérimentales (section 8) permettent d'éclairer la meilleure manière de choisir cet exposant idéal.

Lors de l'étude d'un nouveau premier ℓ, la première étape est de déterminer son type relativement à la proposition 3.1. Pour cela, on utilise l'équation modulaire d'Atkin. On étudie ses facteurs linéaires en calculant $F = X^q \mod A_\ell(j(E), X)$ par une méthode d'exponentiation rapide, puis le pgcd de $F - X$ et $A_\ell(j(E), X)$: c'est le début de l'algorithme de factorisation de Cantor–Zassenhaus [vzGG03]. Celui-ci est de degré respectivement 1, $\ell + 1$, 2 et 0 dans les cas 1, 2, 3 et 4.

On choisit la méthode à appliquer en fonction du type de ℓ et de l'exposant idéal r.

— Si ℓ est de type 1 et $r \geq 2$, on tente d'appliquer la méthode d'Elkies étendue (section 4, plus précisément section 4.3) pour calculer $t \mod \ell^r$. Si $r = 1$, on applique directement la proposition 3.1 : t est l'une des deux racines carrées de $4q$ dans $\mathbb{Z}/\ell\mathbb{Z}$.

— Si ℓ est de type 2 et $r \geq 3$, on tente d'appliquer la méthode d'Elkies étendue pour calculer $t \mod \ell^r$. Si $r \leq 2$, on applique directement la proposition 3.1 : t est l'une des deux racines carrées de $4q$ modulo ℓ^2.

— Si ℓ est de type 3, on applique la méthode d'Elkies, éventuellement étendue si $r \geq 2$ (section 4).

— Enfin, si ℓ est de type 4, on procède de la manière suivante. Si $r \geq 2$, on applique la méthode originale de Schoof pour déterminer uniquement la valeur de $t \mod \ell^{\lfloor \frac{r}{2} \rfloor}$ (section 6). Si $r = 1$, on peut appliquer la méthode d'Atkin pour déterminer un ensemble de traces possibles modulo ℓ (section 5). En pratique, on ne met pas toujours en œuvre la méthode d'Atkin, car l'information collectée est parfois inutilisable (voir section 7).

3.5 Détection des cardinaux interdits

Comme indiqué à la section 3.2, les méthodes d'Elkies et d'Atkin ne peuvent être utilisées lorsque $j(E) \in \{0, 1728\}$ ou lorsque E est supersingulière. En réalité, des problèmes de division par zéro apparaissent dès que l'on

manipule une courbe ayant ces j-invariants, y compris s'il s'agit de l'image d'une isogénie que l'on calcule lors de la méthode d'Elkies. Les courbes problématiques sont donc celles de j-invariant 0 ou 1728, toutes les courbes elliptiques qui leur sont isogènes, et les courbes supersingulières.

Plutôt que de tenter d'adapter l'algorithme SEA à ces cas particuliers, on préfère détecter ces problèmes avant l'entrée dans l'algorithme. Plus précisément, on va calculer un ensemble C contenant les valeurs possibles pour le nombre de points sur $\mathbb{F}_q$ des courbes supersingulières et de tous les twists des courbes

$$E_0 \ : \ y^2 = x^3 + x \quad \text{et} \quad E_{1728} \ : \ y^3 = x^3 + 1$$

de j-invariant respectivement 0 et 1728. Lors de la donnée d'une courbe E, on choisit $P \in E(\mathbb{F}_q)$ aléatoirement, puis l'on calcule $[c]P$ pour chaque $c \in C$. Si l'un des $[c]P$ est nul, on déclare que E est un cas pathologique et que c est probablement son cardinal ; sinon, on applique normalement l'algorithme SEA.

Une amélioration possible. *Ce test probabiliste peut présenter des faux positifs, notamment lorsque le point P choisi est de petit ordre. À défaut d'un test déterministe de l'égalité* $\mathrm{Card}(E(\mathbb{F}_q)) = c$*, on pourrait utiliser un test probabiliste donnant une meilleure garantie (par exemple, choisir un deuxième point P).*

Actuellement, on recalcule l'ensemble C à chaque nouvelle courbe : ce n'est pas très coûteux, mais on pourrait fournir un moyen de stockage si l'on étudie plusieurs courbes sur le même corps $\mathbb{F}_q$.

On est donc amené à calculer les nombres de points sur $\mathbb{F}_q$ de toutes les courbes pathologiques. La situation est simple pour les courbes supersingulières :

Proposition 3.2. *Une courbe elliptique $E/\mathbb{F}_q$ dont la trace du Frobenius est t est supersingulière si et seulement si $t = 0$ dans $\mathbb{F}_q$. Autrement dit, si $q = p^d$ avec p premier, les cardinaux des courbes supersingulières sur $\mathbb{F}_q$ sont parmi les $q + 1 - np$ avec $|n| \leq \frac{2\sqrt{q}}{p}$ entier.*

Démonstration. Voir [Sil86], exercice 5.10. □

Lorsque q est premier ou le carré d'un nombre premier, il y a donc respectivement 1 et 5 valeurs à tester, mais ce nombre explose dès que $d \geq 3$. En fait, on peut donner un meilleur résultat (rappelons que l'on suppose $p \geq 5$) :

Proposition 3.3. *Soit E une courbe elliptique supersingulière sur $\mathbb{F}_q$, où $q = p^r$ avec p premier. Alors les possibilités pour $(p, r, \mathrm{Card}(E(\mathbb{F}_q)))$ sont les suivantes :*

— $Card(E(\mathbb{F}_q)) = p + 1$ *et* r *est impair.*
— $Card(E(\mathbb{F}_q)) = p + 1 \pm 2\sqrt{q}$ *et* r *est pair.*
— $Card(E(\mathbb{F}_q)) = p + 1$, r *est pair et* $p \neq 1 \mod 4$.
— $Card(E(\mathbb{F}_q)) = p + 1 \pm \sqrt{q}$, r *est pair et* $p \neq 1 \mod 3$.

Démonstration. Voir [Wat69], théorème 4.10. □

Cela donne les valeurs à tester pour évacuer les courbes supersingulières dans tous les cas.

Pour les courbes $j = 0$ et $j = 1728$, la situation est différente. Premièrement, si $q = p^d$ avec p premier, alors ces deux courbes sont définies sur $\mathbb{F}_p$. On peut donc calculer les cardinaux de leurs différentes tordues sur $\mathbb{F}_p$, et en déduire leurs cardinaux sur $\mathbb{F}_q$ via la proposition 2.1.

Pour calculer les cardinaux de ces courbes sur $\mathbb{F}_p$, on utilise la méthode de la multiplication complexe. Cette méthode, beaucoup plus efficace que l'algorithme SEA, peut être utilisée lorsque l'anneau d'endomorphismes de la courbe est connu. Rappelons que l'on ne travaille pas en petite caractéristique, on a donc toujours $p \geq 5$.

Proposition 3.4. *Avec les notations précédentes,*
— *Si* $X^2 + 1$ *a une racine dans* $\mathbb{F}_p$ *(i.e. si* p *est décomposé dans* $\mathbb{Q}(i)$*), alors* $\mathrm{End}(E_0) \simeq \mathbb{Z}[i]$. *Sinon,* E_0 *est supersingulière.*
— *Si* $X^2 + X + 1$ *a une racine dans* $\mathbb{F}_p$ *(i.e. si* p *est décomposé dans* $\mathbb{Q}(j)$*), alors* $\mathrm{End}(E_{1728}) \simeq \mathbb{Z}[j]$. *Sinon,* E_{1728} *est supersingulière.*

Démonstration. Voir [Sil86], exemples V.4.4 et 4.5. On remarque par exemple que si $i \in \overline{\mathbb{F}_q}$ est une racine carrée de -1, l'endomorphisme de E_0

$$[i] \ : \ (x, y) \mapsto (ix, -y)$$

commute avec le Frobenius si et seulement si $i \in \mathbb{F}_q$, si et seulement si p est décomposé dans $\mathbb{Q}(i)$, si et seulement si $p = 1 \mod 4$. Si $[i]$ ne commute pas avec le Frobenius, alors E_0 est nécessairement supersingulière puisque son anneau d'endomorphismes n'est pas commutatif. □

Le test de décomposition de p dans $\mathbb{Q}(i)$ ou $\mathbb{Q}(j)$ est un calcul de symbole de Jacobi, ou un simple test de congruence modulo 3 et 4 en utilisant la réciprocité quadratique.

La méthode de la multiplication complexe est la suivante. Soit E une courbe elliptique sur $\mathbb{F}_p$ vérifiant $\mathrm{End}(E) \simeq \mathcal{O}$, où $\mathcal{O}$ est un ordre dans un corps quadratique imaginaire, et π son Frobenius. Soit $\hat{\pi} \in \mathrm{End}(E)$ le dual de l'endomorphisme de Frobenius. Alors l'égalité

$$[p] = \pi\hat{\pi}$$

dans $\mathrm{End}(E)$ implique que (p) est le produit dans $\mathcal{O}$ de deux idéaux principaux, et π est un générateur de l'un d'eux. On a alors

$$\mathrm{Card}(E(\mathbb{F}_p)) = p + 1 - t$$

où t est la trace de π en tant qu'élément de $\mathcal{O}$. Le choix de l'idéal conjugué n'influence pas la trace, et le choix d'un autre générateur (i.e., la multiplication par une unité de $\mathcal{O}$) mène aux cardinaux des différentes tordues de E. Il y a quatre tordues pour E_0, et six pour E_{1728}.

La factorisation de p dans $\mathcal{O}$ peut être faite de manière très efficace, par exemple par l'algorithme de Cornacchia [Sch95].

4 La méthode d'Elkies

4.1 Calcul du noyau de l'isogénie

Lors de l'entrée dans la méthode d'Elkies, le polynôme $A_\ell(j(E), F)$ peut posséder 1, 2 ou $\ell + 1$ racines dans $\mathbb{F}_q$ selon les cas. Pour calculer un sous-groupe cyclique de E de cardinal ℓ, on va calculer une isogénie de degré ℓ au départ de E. On cherche donc une racine f de $A_\ell(j(E), F)$ (ou plus précisément, une racine du pcgd déjà calculé avec $X^q - X$) : le j-invariant j_1 de la courbe image est ensuite l'une des racines de $A_\ell(J, f)$ distinctes de $j(E)$.

Lorsque l'on utilise le polynôme Φ_ℓ, on prend simplement pour j_1 l'une des racines de $\Phi_\ell(j(E), Y)$. Pour le calcul des racines, on applique l'algorithme de Cantor–Zassenhaus, avec un calcul de racine carrée lorsque l'on atteint le degré 2.

Le calcul du noyau de l'isogénie de E vers une courbe de j-invariant j_1 se déroule en deux étapes :

— Calculer une équation normalisée pour une courbe E_1 de j-invariant j_1, ainsi que la demi-somme s des abscisses des points du noyau.

— Calculer le polynôme de noyau à partir de ces données.

La première étape dépend du type d'équation modulaire choisi, mais la seconde en est indépendante. Dans le cas de A_ℓ, il faut également tenir compte des valeurs parasites de j_1.

On appelle $K(X)$ le polynôme de noyau : son degré est $n = \frac{\ell-1}{2}$ si ℓ est impair, et $n = 1$ si $\ell = 2$. C'est ce polynôme que l'on souhaite calculer et que l'on utilisera pour le calcul de la valeur propre. On appelle s la demi-somme des abscisses des points du noyau : le coefficient de X^{n-1} dans K est $-s$ si ℓ est impair, et $-2s$ sinon. L'équation de la courbe E est notée $y^2 = x^3 + ax + b$, et son j-invariant est noté j.

Calcul de l'équation normalisée. On s'appuie sur les propositions suivantes. On abrègera la dérivée partielle

$$\frac{\partial \Phi_\ell(X, Y)}{\partial X}(j, j_1)$$

en $\partial_X(j, j_1)$ dans la première proposition, avec une notation similaire pour A_ℓ dans la seconde. Ces résultats proviennent de calculs sur les formes modulaires : voir [Mor95] (section 3.2, notamment 3.2.2 pour le polynôme A_ℓ) et [Elk97] (section 3, « Quadratic twists »).

Proposition 4.1 (Équation normalisée, cas de Φ_ℓ)**.** *On définit successive-*

ment :

$$E_4 = -\frac{a}{3} \qquad E_6 = -\frac{b}{2}$$
$$j' = -j\frac{E_6}{E_4} \qquad j_1' = -\frac{j'}{\ell}\frac{\partial_X(j,j_1)}{\partial_Y(j,j_1)}$$
$$E_{4,1} = \frac{j_1'^2}{j_1(j_1-1728)} \qquad E_{6,1} = -\frac{j_1'^3}{j_1^2(j_1-1728)}$$
$$a_1 = -3\ell^4 E_{4,1} \qquad b_1 = -2\ell^6 E_{6,1}$$

Alors $y^2 = x^3 + a_1x + b_1$ *est une équation normalisée pour* E_1.

On définit de plus

$$v = \frac{3\ell}{2}\frac{E_4^2}{E_6} + 2\ell\frac{E_6}{E_4}, \qquad v_1 = \frac{3\ell^2}{2}\frac{E_{4,1}^2}{E_{6,1}} + 2\ell^2\frac{E_{6,1}}{E_{4,1}},$$
$$s = v_1 - v + 3\ell\frac{j'^2\partial_X^2(j,j_1) + 2\ell j' j_1'\partial_X\partial_Y(j,j_1) + \ell^2 j_1'^2\partial_Y^2(j,j_1)}{j'\partial_X(j,j_1)}$$

Alors s *est la somme des abscisses cherchée.*

Proposition 4.2 (Équation normalisée, cas de A_ℓ)**.** *On définit successivement :*

$$E_4 = -\frac{a}{3} \qquad E_6 = -\frac{b}{2}$$
$$f' = j\frac{E_6\partial_J(j,f_1)}{E_4\partial_F(j,f_1)} \qquad q = \frac{f'\partial_F(j_1,f_1)}{\ell j_1\partial_J(j_1,f_1)}$$
$$E_{4,1} = q^2\frac{j_1}{(j_1-1728)} \qquad E_{6,1} = qE_{4,1}$$
$$a_1 = -3\ell^4 E_{4,1} \qquad b_1 = -2\ell^6 E_{6,1}$$

Alors $y^2 = x^3 + a_1x + b_1$ *est une équation normalisée pour* E_1.

On définit de plus

$$u = -f'\partial_F^2(j,f_1) + 2j\partial_J\partial_F(j,f_1)\frac{E_6}{E_4} - f'\frac{E_4^2}{E_6^2}(j\partial_J(j,f_1) + j^2\partial_J^2(j,f_1))$$
$$v = \frac{u}{\partial_F(j,f_1)} + \frac{E_6}{3E_4} - \frac{E_4^2}{2E_6}$$
$$u_1 = -f'\partial_F^2(j_1,f_1) + 2\ell j_1\partial_J\partial_F(j_1,f_1)\frac{E_{6,1}}{E_{4,1}} - \ell f'\frac{E_{4,1}^2}{E_{6,1}^2}(\ell j_1\partial_J(j_1,f_1) + \ell^2 j_1^2\partial_J^2(j_1,f_1))$$
$$v_1 = \frac{u_1}{\partial_F(j_1,f_1)} + \frac{\ell E_{6,1}}{3E_{4,1}} - \frac{\ell E_{4,1}^2}{2E_{6,1}}$$
$$s = 3\ell(v - v_1)$$

Alors s *est la somme des abscisses cherchée.*

Les noms E_4, E_6, j', etc. ne sont pas choisis au hasard : dans le cas complexe, on peut les interpréter comme les séries d'Eisenstein usuelles et la dérivée de la fonction j, respectivement. La quantité s, elle, est liée à la série d'Eisenstein E_2. Les calculs effectués restent valides dans un corps fini, sauf si l'on rencontre une division par zéro.

En pratique, on s'arrange pour stocker certaines quantités qui apparaissent plusieurs fois, comme $\ell j_1 \partial_J(j_1, f_1)$. Cela permet de gagner un peu de temps, mais surtout d'alléger le code. Dans le même ordre d'idées, les dérivées des équations modulaires sont calculées une seule fois, et l'on stocke les quantités j, f, $\partial_J(j, f)$, etc. dans une même structure.

Cas d'échec. Dans le cas où une des dérivées $\frac{\partial \Phi_\ell}{\partial_X}(j, j_1)$, $\frac{\partial \Phi_\ell}{\partial_Y}(j, j_1)$ ou leurs analogues dans le cas de A_ℓ s'annulent, cet algorithme échoue. Cela est relié à l'existence d'endomorphismes de E de petit degré : la proposition suivante découle par exemple des propriétés des équations modulaires.

Proposition 4.3. *Avec les notations précédentes dans le cas du polynôme d'Atkin A_ℓ, les énoncés suivants sont équivalents :*

1. $\partial_F(j, f_1) = 0$
2. $\partial_J(j, f_1) = 0$
3. *f_1 est une racine double de $A_\ell(j, X)$*
4. *La courbe E admet un endomorphisme de degré ℓ.*

Lorsque de tels endomorphismes existent (ce qui n'est pas le cas pour des courbes choisies « aléatoirement »), on sait que l'anneau d'endomorphismes de la courbe a un petit discriminant. On quitte alors l'algorithme SEA pour appliquer la méthode CM, identique à celle de la section 3.5, qui permet très rapidement de calculer le nombre de points de E.

Dans la suite de ce document, on suppose que ces annulations n'ont pas lieu, on continue donc normalement l'algorithme SEA.

Calcul du noyau. La littérature concernant le calcul du noyau à partir d'une équation normalisée est vaste. En termes d'opérations dans $\mathbb{F}_q$, des algorithmes quadratiques en le degré ont d'abord été proposés [Elk97], suivis par des algorithmes quasi-linéaires [BMSS08]. Ces algorithmes sont valables lorsque ℓ est petit devant la caractéristique, ce que l'on suppose dans le cadre de SEA. On suppose également que ℓ est impair, puisque le polynôme K est déjà connu entièrement à ce stade lorsque $\ell = 2$.

Actuellement, seul un algorithme quadratique est implémenté. Il consiste à calculer les sommes de puissances symétriques du polynôme de noyau $K(X)$, et repose sur le résultat suivant :

Proposition 4.4 (Calcul du noyau, algorithme quadratique)**.** *On suppose ℓ impair. Muni des données précédentes, on définit*

$$c_0 = 0, \quad c_1 = \frac{a - a_1}{5}, \quad c_2 = \frac{b - b_1}{7}$$
$$\forall k \geq 2,\ c_{k+1} =$$
$$\frac{1}{(k-1)(2k+5)} \left(3 \sum_{i=1}^{k-1} c_i c_{k-i} - (k-2)(2k-2) b c_{k-2} - (k-1)(2k-1) a c_{k-1} \right)$$
$$p_0 = n, \quad p_1 = s, \quad p_2 = \frac{c_1 - 2an}{6}, \quad p_3 = \frac{c_2 - 6as - 4bn}{10}$$
$$\forall k \geq 4,\ p_k = c_{k-1} - (4k-6) a c_{k-2} - (4k-8) b c_{k-3}$$

Alors p_k est la somme des racines de $K(X)$ mises à la puissance k. Par conséquent, si l'on applique l'algorithme de Newton en posant

$$e_0 = 1, \qquad \forall k \geq 1,\ e_k = \frac{-1}{k} \sum_{i=1}^{k} p_i e_{k-i}$$

alors $K(X) = \sum_{i=0}^{n} e_{n-i} X^i$.

On obtient ces égalités à partir des formules de Vélu, qui donnent l'expression d'une isogénie à partir de son noyau : voir [Elk97] (section 3, « The kernel of the isogeny ») pour une preuve complète.

Le degré de K est $n = \frac{\ell-1}{2}$. Pour calculer K, il faut connaître les quantités p_k jusqu'à $k = n - 1$: il faut donc calculer c_k jusqu'à $k = n - 2$. Par conséquent, il faut savoir diviser un élément de $\mathbb{F}_q$ par tous les entiers entre 2 et

$$2(n-3) + 5 = \ell - 2.$$

La caractéristique de $\mathbb{F}_q$ doit donc être strictement supérieure à cette valeur.

Il reste à évacuer les valeurs parasites de j_1 qui ne correspondent pas à une courbe liée à E par une isogénie de degré ℓ. Lorsque j_1 est parasite, les valeurs de a_1 et b_1 sont arbitraires, et les p_k ne sont en général pas la suite des sommes de Newton d'un polynôme de degré n. Il est donc utile de calculer quelques p_k supplémentaires, et de vérifier que

$$\sum_{i=0}^{n} e_{n-i}\, p_{i+t} = 0$$

par exemple pour $1 \leq t \leq 3$. Si ce test échoue, alors j_1 est une valeur parasite, autrement on accepte le résultat $K(X)$ comme valide. Cette idée est tirée de [Mor95].

Une amélioration possible. *En général, cette étape de calcul du noyau est négligeable devant le reste de SEA, notamment devant le calcul des racines*

de l'équation modulaire. Ce n'est pas tout à fait vrai lorsqu'un grand nombre d'essais doit être fait pour éliminer les valeurs parasites de j_1. Un algorithme quasi-linéaire pourrait-il apporter un gain sur l'algorithme quadratique dans notre plage de valeurs ?

Plutôt que d'éliminer les valeurs parasites de j_1 après avoir calculé entièrement le faux noyau, on pourrait utiliser la remarque suivante : si j_1 est valide, alors la trace d'une courbe de j-invariant j_1 (au signe près, par l'existence de la tordue) doit être égale à celle de E. Tester cette égalité modulo 5, 7, ou 11 (ou les trois), à l'aide de la méthode de Schoof par exemple, permettrait d'évacuer plus rapidement une grande partie des valeurs parasites.

Un test de validité. *Une fois que le polynôme de noyau a été calculé, si P désigne le point générique sur ce sous-groupe, on peut vérifier que $[\ell]P$ est effectivement nul.*

4.2 Calcul de la valeur propre

On dispose maintenant d'un polynôme K définissant un sous-groupe cyclique d'ordre ℓ de E, et l'on aimerait connaître la valeur propre du Frobenius sur ce sous-groupe. On suppose ici que ℓ est impair : il n'y a rien à faire pour $\ell = 2$, car la valeur propre du Frobenius n'est jamais nulle.

Calculer la valeur propre sur K fait intervenir des calculs sur le point générique modulo K, donc des points dont les coordonnées vivent a priori dans l'anneau

$$\mathbb{F}_q[X,Y]/(Y^2 = X^3 + aX + b,\ K(X) = 0).$$

En pratique, tous les points que l'on va manipuler sont *impairs* dans le sens suivant : si l'on écrit toutes leurs coordonnées sous la forme $A_1(X) + YA_2(X)$, on a $A_2 = 0$ pour la coordonnée x, et $A_1 = 0$ pour la coordonnée y. Par conséquent, on écrira le point $(A_1(X), YB_1(X))$ sous la forme du couple de polynômes *univariés* (A_1, B_1). De même, un point projectif écrit sous la forme $(A : B : C)$ représente en réalité le point $(A(X) : YB(X) : C(X))$.

Afin d'éviter les divisions de polynômes, qui sont beaucoup plus coûteuses que les multiplications et qui peuvent échouer lorsque K n'est pas irréductible, on utilise les coordonnées projectives. Attention ! Il ne faut pas oublier le facteur Y caché lors des opérations sur les points.

La première étape est de calculer le Frobenius sur ce sous-groupe. Suivant la convention ci-dessus, il s'agit de calculer les polynômes

$$Y^{q-1} = (X^3 + aX + b)^{\frac{q-1}{2}} \quad \text{et} \quad X^q \mod K.$$

On peut faire cela par deux exponentiations rapides. On désigne par P le point générique, et par F le Frobenius sur le sous-groupe défini par K.

L'algorithme naïf. Il consiste à calculer les multiples $[v]P$, pour v compris entre 1 et $\ell - 1$, et à s'arrêter dès que l'un d'eux est égal à F. Il n'est pas nécessaire de les stocker : on effectue donc simplement $\ell - 1$ additions par P et tests d'égalité.

Le test d'égalité de deux points

$$Q_1 = (X_1 : Y_1 : Z_1) \quad \text{et} \quad Q_2 = (X_2 : Y_2 : Z_2)$$

prend alors la forme suivante : si $X_1 Z_2 = X_2 Z_1$, on a $Q_1 = \pm Q_2$, avec égalité lorsque $Y_1 Z_2 = Y_2 Z_1$. Par conséquent, il s'agit d'un test simultané d'égalité et d'égalité avec l'opposé, et on peut donc diviser par deux le coût de la recherche en calculant les multiples seulement jusqu'à $v = \frac{\ell-1}{2}$ inclus.

Pour additionner les points, on utilise les formules de Weierstrass en coordonnées projectives. Elles sont classiques, mais on reproduit ici l'algorithme utilisé (rappelons que les coordonnées sont des polynômes univariés en X, et que les multiplications s'entendent modulo K) : voir algorithme 1.

Entrées : $P_1 = (X_1 : Y_1 : Z_1)$, $P_2 = (X_2 : Y_2 : Z_2)$
Sorties : La somme $P_3 = (X_3 : Y_3 : Z_3)$ de P_1 et P_2

si $Z_1 = 0$ **alors**
  retourner P_2 ;
si $Z_2 = 0$ **alors**
  retourner P_1 ;

$X_{\text{det}} \leftarrow X_2 Z_1 - X_1 Z_2$;
$Y_{\text{det}} \leftarrow Y_2 Z_1 - Y_1 Z_2$;
$S \leftarrow X^3 + aX + b$; `/* Carré du facteur Y caché */`
si $X_{\text{det}} = Y_{\text{det}} = 0$ **alors**
  `/* Doublement */`
  $X_3 \leftarrow 2SY_1 Z_1[(aZ_1^2 + 3X_1^2)^2 - 8X_1 SY_1^2 Z_1]$;
  $Y_3 \leftarrow (aZ_1^2 + 3X_1^2)[12X_1 SY_1^2 Z_1 - (aZ_1^2 + 3X_1^2)^2] - 8S^2 Y_1^4 Z_1^2$;
  $Z_3 \leftarrow (2Y_1 Z_1)^3 S^2$;
  retourner $P_3 = (X_3 : Y_3 : Z_3)$;
sinon
  `/* Addition générique */`
  $X_3 \leftarrow X_{\text{det}}[SY_{\text{det}}^2 Z_1 Z_2 - X_{\text{det}}^2 (X_2 Z_1 + X_1 Z_2)]$;
  $Y_3 \leftarrow Y_{\text{det}}[X_{\text{det}}^2 (X_2 Z_1 + 2X_1 Z_2) - SY_{\text{det}}^2 Z_1 Z_2] - X_{\text{det}}^3 Y_1 Z_2$;
  $Z_3 \leftarrow X_{\text{det}}^3 Z_1 Z_2$;
  retourner $P_3 = (X_3 : Y_3 : Z_3)$;
fin

Algorithme 1 : Addition de points sur la courbe E

Bien sûr, dans l'implémentation réelle, on peut s'arranger pour calculer une seule fois des quantités apparaissant à plusieurs reprises.

Une amélioration possible. *Il est possible de gagner encore quelques multiplications dans l'implémentation réelle de l'algorithme 1. Par exemple, on calcule actuellement deux fois le carré de $aZ_1^2 + 3X_1^2$ dans le cas du doublement. Dans le cas générique, on ne réutilise pas les quantités X_1Z_2 et Y_1Z_2 déjà utilisées lors du calcul des déterminants. Éviter des multiplications accélérerait directement le calcul de la valeur propre.*

Un test de validité. *Lorsque l'on manipule des points sur une courbe elliptique, on peut vérifier que chaque point vérifie l'équation de la courbe.*

La méthode de Shanks. Lorsque ℓ devient plus grand, on peut remplacer l'algorithme naïf ci-dessus par une méthode inspirée des pas de bébé–pas de géants de Shanks. Cela réduit le nombre d'additions de points de $O(\ell)$ à $O(\sqrt{\ell})$. On utilise l'idée suivante, tirée de [GM06] :

— Choisir des bornes I, J tels que $IJ > \ell$;
— Calculer les multiples $[i]P$, pour $1 \leq i \leq I$;
— Calculer les multiples $[j]F$, pour $1 \leq j \leq J$;
— Chercher une collision au signe près entre ces deux ensembles.

La réussite est assurée par le fait élémentaire suivant : étant donné v non nul dans $\mathbb{Z}/\ell\mathbb{Z}$, il est toujours possible de trouver $1 \leq i \leq I$ et $1 \leq j \leq J$ tel que $v = \pm i/j \mod \ell$ pourvu que $IJ > \ell$.

On peut calculer les multiples par additions successives. En revanche, la recherche de collision n'est pas directe : comme les points sont projectifs, il ne suffit pas simplement de trier les tableaux. Calculer tous les produits croisés annulerait le gain apporté par la méthode de Shanks, et l'on souhaite toujours éviter les divisions, qui sont trop coûteuses. Deux solutions tirées de [GM06] ont été implémentées.

La première consiste à mettre toutes les abscisses au même dénominateur par $O(\sqrt{\ell})$ multiplications, puis à trier ces deux tableaux pour trouver une collision. Comparer les ordonnées permet ensuite de déterminer le signe dans $v = \pm i/j$.

La seconde est plus astucieuse : fixant un vecteur $w = (w_0, \ldots, w_{n-1})$, où n est le degré de K, on définit la forme linéaire

$$\begin{aligned} L_w \ : \ \mathbb{F}_q[X]/K(X) &\to \mathbb{F}_q \\ \sum_{i=0}^{n-1} a_i X^i &\mapsto \sum_{i=0}^{n-1} a_i w_i. \end{aligned}$$

Plutôt que de calculer tous les produits croisés $X_iZ_j - Z_iX_j$, on peut simplement calculer les quantités $L_w(X_iZ_j)$ et $L_w(Z_iX_j)$. Si ces deux quantités sont différentes, alors l'égalité de polynômes n'a pas lieu. Si elles sont égales, on vérifie que l'égalité polynomiale a bien lieu. La probabilité de faux positifs est de l'ordre de $\frac{1}{q}$, donc très faible.

Pour calculer les quantités $L_w(X_iZ_j)$ sans effectuer le produit, on utilise le fait suivant : si M_a désigne la matrice de multiplication par le polynôme a dans $\mathbb{F}_q[X]/K$, on a

$$L_w(uv) = ({}^tM_u w) \cdot v,$$

où $\cdot$ désigne le produit scalaire coefficient par coefficient. Le vecteur ${}^tM_u v$ peut être calculé dans la même complexité que $M_u w$, à savoir pour le coût d'une multiplication : c'est une fonctionnalité présente dans NTL [Sho17]. On procède donc comme suit :

- Pour chaque $1 \leq i \leq I$, on calcule $A_i = {}^tM_{X_i} w$;
- On construit la matrice M_1 de taille $I \times n$ dont l'élément (i,k) est le k-ième coefficient de A_i ;
- On construit la matrice M_2 de taille $n \times J$ dont l'élément (k,j) est le k-ième coefficient de Z_j ;
- On calcule le produit $M_3 = M_1 M_2$; le coefficient (i,j) de M_3 est alors $L_w(X_iZ_j)$.

On procède de même pour calculer la matrice dont les entrées sont les $L_w(Z_iX_j)$, et une collision entre ces deux matrices donne le résultat.

Asymptotiquement, cet algorithme est moins bon que le précédent du fait des produits de matrices, mais il est tout de même intéressant en pratique. C'est cette seconde méthode que l'on retient avec la librairie NTL, qui propose un algorithme de calcul de ${}^tM_a b$; on garde la première avec Flint.

Un contrôle peut être exercé sur la méthode des pas de bébé–pas de géants au niveau du choix de I et J : typiquement, on veut que I et J soient de l'ordre de $\sqrt{\ell}$, mais on veut aussi $I > J$ car les multiples du Frobenius sont plus coûteux à calculer que ceux du point générique (voir section 8).

L'utilisation des polynômes de division. Dans la méthode ci-dessus, on est amené à calculer les multiples d'un point pour toute une plage de valeurs. Plus précisément, étant donné $P = (x : y : z)$, on veut calculer des polynômes N_i, D_i tels que l'abscisse de $[i]P$ soit N_i/D_i, ainsi qu'un moyen de calculer l'ordonnée de $[i]P$. On supposera dans ce paragraphe que $z = 1$: c'est le cas pour le Frobenius et le point générique. Pour cela, il existe une méthode plus efficace que des additions successives, donnée dans [GM06].

Soit Ψ_i le i-ième polynôme de division (bivarié) de la courbe E :

$$\Psi_0 = 0,\ \Psi_1 = 1,\ \Psi_2 = 2Y,$$
$$\Psi_3 = 3X^4 + 6aX^2 + 12bX - a^2,$$
$$\Psi_4 = 4Y(X^6 + 5aX^4 + 20bX^3 - 5a^2X^2 - 4abX - 8b^2 - a^3),\ \ldots$$

On définit le polynôme univarié f_i par $\Psi_i = f_i$ si i est pair, et $\Psi_i = 2Y f_i$ si i est impair. Alors, pour tout point $Q = (x : y : z)$ sur la courbe E vérifiant $z = 1$ et tout $i \geq 1$, on a

$$[i]Q = \left(x - \frac{\Psi_{i-1}\Psi_{i+1}}{\Psi_i^2}(x), \frac{1}{4y}\frac{\Psi_{i+2}\Psi_{i-1}^2 - \Psi_{i-2}\Psi_{i+1}^2}{\Psi_i^3}(x) \right). \tag{6}$$

Lorsque Q vit dans un sous-groupe, et x, y, z sont donc des polynômes univariés suivant la convention précédente, on définit

$$F_i = f_i(x),\ V_i = f_i(x)^2,\ U_i = f_{i-1}(x)f_{i+1}(x),\ R = 4(x^3 + ax + b).$$

Vu 6, l'abscisse de $[i]Q$ est alors N_i/D_i où

$$\begin{aligned} N_i &= xRV_i - U_i,\ D_i = RV_i && \text{si } i \text{ est pair,} \\ N_i &= xV_i - RU_i,\ D_i = V_i && \text{si } i \text{ est impair.} \end{aligned}$$

On voit donc que l'abscisse de $[i]Q$ peut être obtenue à partir des quantités F_i, V_i, U_i en deux multiplications, lorsque xR est précalculé. De même, l'ordonnée de $[i]Q$ est N_i/D_i où

$$\begin{aligned} N_i &= F_{i+2}V_{i-1} - F_{i-2}V_{i+1}, \\ D_i &= R^2y^3F_iV_i \end{aligned}$$

si i est pair (ne pas oublier le facteur Y omis dans l'ordonnée de Q), et

$$\begin{aligned} N_i &= y(F_{i+2}V_{i-1} - F_{i-2}V_{i+1}), \\ D_i &= F_iV_i \end{aligned}$$

si i est impair.

Il reste à calculer efficacement les quantités F_i, U_i, V_i. Pour cela, on peut utiliser les formules récursives suivantes, tirées des formules classiques pour les polynômes de division Ψ_i : pour tout $i \geq 2$, on a

$$\begin{aligned} F_{2i+1} &= R^2V_iU_{i+1} - V_{i+1}U_i && \text{si } i \text{ est pair,} \\ F_{2i+1} &= V_iU_{i+1} - V_{i+1}U_i && \text{si } i \text{ est impair,} \end{aligned}$$

et pour tout $i \geq 3$,

$$F_{2i} = V_{i-1}U_{i+1} - V_{i+1}U_{i-1}.$$

Pour chaque i, on doit donc effectuer 2 ou 3 multiplications pour le calcul de F_i, une multiplication pour le calcul de V_i, deux multiplications pour le calcul de U_i, et deux multiplications pour le calcul final de l'abscisse. On économise donc des multiplications par rapport à l'algorithme 1, même optimisé.

Un calcul plus rapide du Frobenius. Enfin, il est possible d'accélérer le temps de calcul du Frobenius, qui est le coût dominant dans le calcul de la valeur propre. La remarque est la suivante : une fois calculé le polynôme Y^{q-1}, on peut facilement calculer

$$F(X) = Y^{2q} = (X^3 + aX + b)^q.$$

Alors, l'élément $X^q \mod K$ est solution des deux équations

$$W^3 + aW + b = F,$$
$$K(W) = 0$$

dans l'anneau $\mathbb{F}_q[X]/K$. Sans faire de division modulo K, on peut calculer un pseudo-pgcd de $K(W)$ et $W^3+aW+b-F(W)$ dans l'anneau $(\mathbb{F}_q[X]/K)[W]$. Lorsque que l'on obtient un polynôme de degré 1 (ce qui est le cas en général), on sait que son unique racine est la valeur de X^q. Sinon, on calcule X^q par exponentiation.

L'étape la plus coûteuse dans cette méthode est la réduction de $K(W)$ modulo $W^3+aW+b-F$. Concrètement, on fait une évaluation à la Horner ; on ne manipule que des éléments sous la forme

$$c_2(X)W^2 + c_1(X)W + c_0(X),$$

et la multiplication par W est donnée par

$$c_2 \leftarrow c_1,\ c_1 \leftarrow c_0 - ac_2,\ c_0 \leftarrow (-b+F)c_2.$$

Ensuite, le pseudo-pgcd est calculé avec un nombre constant de multiplications. Une fois celui-ci calculé, on obtient un polynôme de degré 1 qui n'a aucune raison d'être unitaire : il faut donc effectuer une division modulo K pour retrouver la valeur de X^q. On pourrait aussi incorporer une coordonnée Z non triviale dans la méthode des polynômes de division ci-dessus, mais cela engendrerait un nombre de multiplications supplémentaire prohibitif : effectuer cette division est donc plus intéressant.

Remarque. *Lorsque l'on utilise NTL, on a l'opportunité de faire des précalculs relatifs à la multiplication de polynômes par B modulo H où B et H sont fixés. On utilise cette fonctionnalité dans le calcul rapide du Frobenius pour $B = -b + F$, mais aussi pour x et xR dans le calcul des abscisses à l'aide des polynômes de division.*

4.3 Relèvement modulo ℓ^r : le cas des petits premiers

On a vu dans le début de cette section comment calculer un sous-groupe cyclique d'ordre ℓ sur une courbe E lorsqu'il en existe, puis comment calculer la valeur propre sur ce sous-groupe. Il est également possible de calculer la valeur propre du Frobenius modulo ℓ^r, si l'on connaît un sous-groupe cyclique de E d'ordre ℓ^r. Pour obtenir un tel sous-groupe, on va composer des isogénies de degré ℓ : la préimage par une telle isogénie d'un sous-groupe cyclique d'ordre ℓ^r est (sous certaines conditions) un sous-groupe cyclique d'ordre ℓ^{r+1}.

Cette remarque a deux cas d'utilisation, suivant l'exposant idéal r de ℓ (voir section 3.4) :

— Lorsque $r \geq 1$, ℓ est petit (par exemple $\ell < 15$, voir section 8), on est sûr de vouloir calculer la valeur propre modulo une puissance de ℓ. Cette puissance peut parfois être élevée, notamment lorsque $\ell = 2$ ou $\ell = 3$: pour cette raison, on va s'efforcer de trouver la plus longue chaîne d'isogénies possible. On utilise le polynôme classique Φ_ℓ.
— Lorsque $r = 1$ et ℓ est moyen (par exemple $15 < \ell < 100$), on va éventuellement calculer la valeur propre modulo ℓ^2 si le polynôme de noyau $K(X)$ admet un facteur de petit degré. On va calculer une seule isogénie et on utilise l'équation modulaire A_ℓ, que l'on évite au maximum de manipuler.

Ces deux cas sont différents d'un point de vue algorithmique. Dans cette partie, on traite celui des petits premiers : le cas des premiers moyens fait l'objet de la partie 4.4. La plupart des idées proviennent de [CDM96], bien que la structure des graphes d'isogénies ne soit pas utilisée dans cet article.

Graphes d'isogénies. On introduit le *graphe de ℓ-isogénies* sur $\mathbb{F}_q$. Les sommets de ce graphe sont les courbes elliptiques sur $\mathbb{F}_q$ à isomorphisme près sur la clôture algébrique : ils peuvent donc être identifiés aux j-invariants. Une arête relie j_1 à j_2 s'il existe une isogénie $\mathbb{F}_q$-rationnelle de degré ℓ reliant E_1 à E_2, avec $j(E_1) = j_1$ et $j(E_2) = j_2$. On se permettra d'indexer des sommets par des courbes, bien que cette description ne soit pas unique : seule leur classe d'isomorphisme est bien définie. Ce graphe est non orienté du fait de l'existence de l'isogénie duale. D'après la section 2, les voisins d'un sommet j sont les racines du polynôme modulaire $\Phi_\ell(j, X)$. On utilise le polynôme classique Φ_ℓ pour naviguer dans le graphe, ce qui ne pose pas de problème puisque ℓ est supposé petit.

On s'intéresse ici à la composante connexe dans laquelle vit la courbe E : comme ℓ est éligible à la méthode d'Elkies, ce graphe n'est pas un point, et comme E n'est pas isogène à $j = 0$ et $j = 1728$, ses arêtes sont simples. De plus, E est ordinaire. Dans ce cas, Kohel [Koh96] a montré que ces graphes peuvent être de plusieurs types, que l'on appellera ici *cyclique simple*, *dégénéré simple*, *cyclique volcanique*, *dégénéré volcanique* et *point volcanique* (voir les dessins page 30). L'arité d'un sommet d'un graphe volcanique est soit 1, soit $\ell + 1$, et l'on appelle *hauteur* d'un sommet la plus petite distance de celui-ci à un sommet d'arité 1.

Un chemin dans le graphe de ℓ-isogénies correspond donc à une chaîne d'isogénies de degré ℓ. Si cette chaîne est de longueur r, on montre facilement que le noyau de la composition est un groupe cyclique d'ordre ℓ^r si et seulement si le chemin ne fait pas de demi-tour dans le graphe : un demi-tour correspond à la composition d'une isogénie et de sa duale, ce qui fait apparaître un facteur $[\ell]$ dont le noyau n'est pas cyclique.

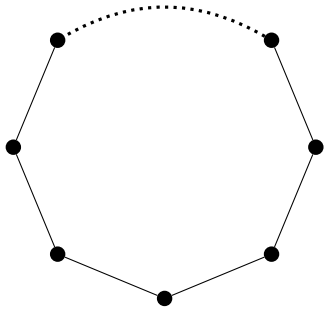


FIGURE 3 – Graphe d'isogénies cyclique simple

FIGURE 4 – Graphe d'isogénies dégénéré simple

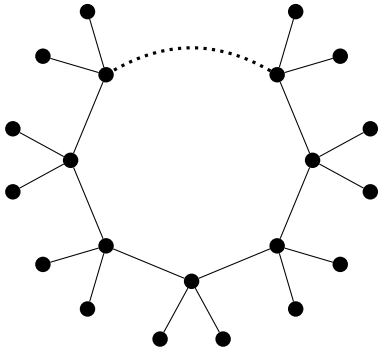


FIGURE 5 – Graphe d'isogénies cyclique volcanique ($\ell = 3$, profondeur 1)

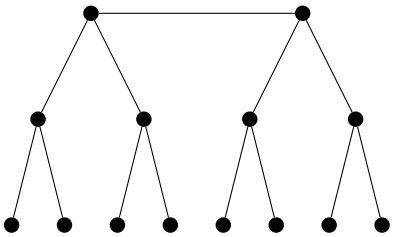


FIGURE 6 – Graphe d'isogénies dégénéré volcanique ($\ell = 2$, profondeur 2)

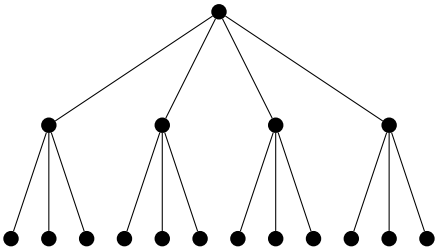


FIGURE 7 – Graphe d'isogénies point volcanique ($\ell = 3$, profondeur 2)

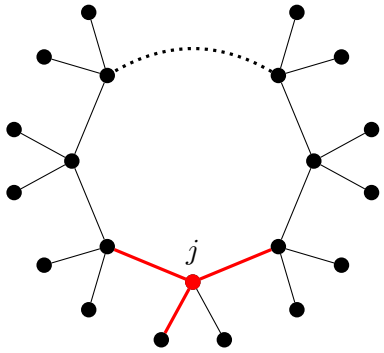


FIGURE 8 – Calcul de hauteur dans le graphe d'isogénies

Recherche de la plus longue chaîne. Lorsque l'on a décidé de calculer la trace du Frobenius modulo ℓ^r, la première étape est donc de trouver un chemin de longueur r dans le graphe d'isogénies. Dans le cas cyclique simple, il suffit de choisir à chaque étape le voisin distinct du précédent ; dans le cas dégénéré simple, c'est impossible et on peut uniquement calculer la trace modulo ℓ. Dans les cas volcaniques, il faut faire attention au voisin que l'on choisit.

On va calculer la hauteur des sommets visités dans le graphe : partant d'un point, on parcourt simultanément trois chemins de directions différentes sans retour en arrière : le premier qui atteint un sommet d'arité 1 donne la hauteur du sommet initial. Sur la figure 8, j est de hauteur 1.

On peut également donner un moyen, moins coûteux, de savoir si le pas $j \to j'$ est descendant, connaissant la hauteur h de j : il est (strictement) descendant si et seulement s'il est impossible de prolonger ce pas par un chemin de longueur h sans retour en arrière. Il n'y a pas de moyen aussi simple de différencier les pas montants des pas horizontaux : pour savoir si un pas est montant, on vérifie d'abord qu'il n'est pas descendant puis on calcule la hauteur du point d'arrivée. Cela est justifié par le fait qu'il existe plus de pas descendants que montants à partir d'un point donné.

Une amélioration possible. *Lorsque l'on sait que j est de hauteur h et que $j \to j'$ est montant ou horizontal, on a la propriété suivante : $j \to j'$ est montant si et seulement si l'on peut prolonger ce pas par deux chemins de longueur $h+1$ et de directions différentes. On économiserait ainsi un chemin par rapport au calcul de hauteur.*

L'algorithme 2 permet alors de trouver la plus longue chaîne partant de E dans le graphe. Dans le cas cyclique (dégénéré ou non), cet algorithme ne termine pas : dans l'implémentation réelle, on arrête bien sûr le calcul lorsque le chemin a atteint une longueur prédéterminée. Rappelons que l'on n'utilise pas cet algorithme dans le cas cyclique simple.

Une amélioration possible. *D'après un théorème de Tate, deux courbes isogènes sur $\mathbb{F}_q$ ont le même nombre de points sur ce corps. Par conséquent, afin de déterminer la trace modulo ℓ^r, la chaîne choisie n'a pas nécessairement besoin d'avoir la courbe E comme point de départ. Dans le cas dégénéré volcanique, choisir un autre point de départ peut permettre de rallonger la chaîne. Cela nécessiterait de calculer des équations de courbes pour lesquelles l'isogénie duale est normalisée, et non plus l'isogénie directe.*

L'algorithme 2 décrit réellement l'algorithme implémenté. Des optimisations sont possibles : par exemple, si dans l'étape 1 on constate qu'il existe un pas horizontal, celui-ci sera jeté puis recalculé à la première entrée dans l'étape 2. De même, dans le cas d'un point volcanique, on voit à la fin de l'étape 1 qu'il n'existe pas de chemin horizontal, donc on peut éviter de faire un calcul à l'étape 2.

Entrées : Une courbe elliptique $E/\mathbb{F}_q$, un petit premier ℓ éligible à la méthode d'Elkies
Sorties : Un plus long chemin $E \to j_1 \to j_2 \to \dots$ dans le graphe de ℓ-isogénies
$h \leftarrow$ hauteur de E;
$C \leftarrow$ chemin de longueur 0 partant de E;
`/* Étape 1: rechercher un sommet de hauteur maximale */`
tant que *c'est possible* **faire**
  Ajouter un pas montant à C, sachant la hauteur actuelle h;
  $h \leftarrow h + 1$;
fin
`/* Étape 2: parcourir le cycle, s'il existe */`
tant que *c'est possible* **faire**
  Ajouter un pas non descendant à C, sachant la hauteur actuelle h;
fin
`/* Étape 3: s'il n'y a pas de cycle, redescendre */`
tant que *c'est possible* **faire**
  Ajouter un pas à C ; `/* N'importe lequel */`
fin
retourner C ;

Algorithme 2 : Recherche de la plus longue chaîne partant de E

Composition des isogénies. On est maintenant muni d'un chemin

$$E \to j_1 \to \cdots \to j_r$$

dans le graphe d'isogénies. Il s'agit maintenant d'en déduire une valeur propre du Frobenius modulo ℓ^r. La première étape est de calculer des équations pour des courbes $E_1, \dots E_r$ telle que chaque isogénie de la chaîne est normalisée, en utilisant la proposition 4.1. On peut alors en déduire les noyaux $K_0, \dots K_{r-1}$ de chaque isogénie $\phi_i : E_i \to E_{i+1}$, en notant $E_0 = E$. Chaque K_i est un polynôme de degré $\frac{\ell-1}{2}$ en X si ℓ est impair, et de degré 1 si $\ell = 2$.

L'idée est la suivante : pour tout i de 0 à $r-1$, on va calculer le noyau de la composition $\phi_i \circ \cdots \circ \phi_0$. On obtient alors un sous-groupe cyclique de E d'ordre ℓ^{i+1}. Sur ce sous-groupe, on calcule la valeur propre du Frobenius à partir de la valeur propre modulo ℓ^i calculée précédemment, dans une procédure de relèvement à la Hensel.

Proposition 4.5 (Expression de l'isogénie à partir du noyau)**.** *Soit*

$$\phi : (x, y) \mapsto (\phi_x(x), \phi_y(x, y))$$

une isogénie de noyau donné par le polynôme $K(X)$ sur la courbe $E : y^2 = x^3 + ax + b$. Alors :

1. *Si $\ell = 2$, en notant $K(X) = X - x_0$, on a*

$$\phi_x = \frac{X^2 - x_0 X + 3x_0^2 + a}{X - x_0}.$$

2. *Si ℓ est impair, on a $\phi_x = \frac{N}{D}$ avec*

$$N = 4(X^3 + aX + b)(K'^2 - KK'') - (6X^2 + 2a)KK' + (nX - 2s)K^2,$$
$$D = K^2$$

où n désigne le degré de K et s la somme des racines de K.

Démonstration. Voir [Koh96], section 2.4. □

Par exemple, pour calculer le noyau de $\phi_1 \circ \phi_0$, on cherche un polynôme P tel que $P(X) = 0$ si et seulement si $K_1(\phi_{0,x}(X)) = 0$. En écrivant $\phi_{0,x} = \frac{N}{D}$ et $K_1 = \sum_{i=0}^{n} a_i X^i$, on pose donc

$$P = \sum_{i=0}^{n} a_i N^i D^{n-i}.$$

Lorsque ℓ est impair, le degré de P est $\ell\frac{\ell-1}{2}$. En réalité, ce n'est pas le noyau entier de $\phi_1 \circ \phi_0$, qui serait de degré $\frac{\ell^2-1}{2}$: il manque le facteur K_0. Néanmoins, travailler avec un polynôme de degré inférieur est un bénéfice. Ainsi, on calcule le noyau de $\phi_1 \circ \phi_0$ en calculant la préimage du noyau de ϕ_1 par ϕ_0. Pour calculer chaque sous-groupe cyclique d'ordre ℓ^i, on effectue $i - 1$ tels calculs de préimage.

Remarque. *Dans le cas $\ell = 2$, chaque K_i est le carré du « véritable » polynôme de noyau. Par conséquent, il faut effectuer l'opération suivante : après la première prise de préimage comme ci-dessus, on obtient un polynôme de degré 2. Ce polynôme est le carré d'un polynôme de degré 1 par lequel il doit être remplacé avant de continuer la chaîne.*

Une amélioration possible. *Afin d'éviter des calculs de composition, on pourrait calculer la valeur propre modulo ℓ en utilisant K_{r-1}, puis la valeur propre modulo ℓ^2 en utilisant la composition $\phi_{r-1} \circ \phi_{r-2}$, etc. Ainsi, un seul calcul de préimage est nécessaire pour relever la valeur propre de $\mathbb{Z}/\ell^i\mathbb{Z}$ à $\mathbb{Z}/\ell^{i+1}\mathbb{Z}$, au lieu de i. En pratique, ce surcoût est négligeable.*

Relèvement de Hensel. Après chaque calcul de sous-groupe d'ordre ℓ^i sur E, représenté par un polynôme K, on doit répondre à la question suivante : quel est la valeur propre v du Frobenius sur K connaissant $v_0 = v \mod \ell^{i-1}$? La valeur propre peut s'écrire

$$v = v_0 + k\ell^{i-1} \quad \text{avec } 0 \leq k \leq \ell - 1.$$

Comme ℓ est petit, on adopte une variante de l'algorithme naïf du calcul de valeur propre : si P désigne le point générique et F le Frobenius modulo K, on calcule tout d'abord

$$Q = [v_0]P \quad \text{et} \quad R = [\ell^{i-1}]P$$

puis on recherche k tel que l'égalité $Q + [k]R = F$ soit vérifiée, par additions successives de R. Notons qu'il suffit de tester l'égalité des abscisses, car v_0 n'est jamais nul.

On met en œuvre ici les techniques de calcul du Frobenius exposées à la section 4.2. En pratique, c'est l'étape de relèvement qui est la plus coûteuse par rapport aux compositions d'isogénies et aux calcul des chemins dans le graphe. Le coût du calcul de la valeur propre modulo ℓ^r est celui de $O(\ell)$ multiplications de polynôme de degré $\ell^{r-1}\frac{\ell-1}{2}$.

4.4 Relèvement modulo ℓ^2 : le cas des premiers moyens

Soit E une courbe elliptique et ℓ un nombre premier impair pour lequel on vient d'appliquer la méthode d'Elkies « classique ». On dispose alors d'un polynôme K de degré $\frac{\ell-1}{2}$ qui représente un sous-groupe cyclique de E de cardinal ℓ, sur lequel on connaît la valeur propre du Frobenius, v. On suppose ici que le graphe de ℓ-isogénies est cyclique simple : c'est le seul cas d'intérêt la méthode d'Elkies « classique » avec un exposant idéal 1 d'après la section 3.4.

L'idée est la suivante : si K admet un facteur de degré d, alors en composant avec une seconde isogénie de degré ℓ, on peut obtenir un polynôme

de degré $d\ell$. Celui-ci est le polynôme de noyau d'un certain sous-groupe cyclique de ℓ^2-torsion. On peut donc utiliser ce polynôme pour calculer la valeur propre du Frobenius modulo ℓ^2.

De plus, pour détecter la présence de facteurs de K de petit degré, tenter systématiquement de factoriser ce polynôme n'est pas nécessaire :

Proposition 4.6. *Avec les notations ci-dessus, la factorisation de $K(X)$ sur $\mathbb{F}_q$ est de la forme*

$$K = Q_1 \cdots Q_n$$

où les Q_i sont du même degré d. De plus, d est le plus petit entier ≥ 1 tel que $v^d = \pm 1 \mod \ell$.

Cela se montre aisément par le lien entre définition sur une certaine extension de corps d'une part, et invariance par une puissance du Frobenius d'autre part.

En pratique, le calcul de d peut être très naïf (on calcule les puissances successives de v). Pour calculer un des facteurs Q_i, on adapte la méthode de factorisation de Cantor–Zassenhaus. On peut de dispenser de l'étape de factorisation en degrés égaux, et puisque l'on cherche uniquement un facteur, on garde la partie de plus petit degré à chaque tentative de scission.

Calcul de la seconde isogénie. Contrairement à la section précédente, on tente ici de minimiser les calculs sur les équations modulaires. Lors de la méthode d'Elkies classique, comme le graphe d'isogénies est cyclique simple, on a trouvé deux voisins j_1 et j_{-1} de la courbe E dans le graphe de ℓ-isogénies : quitte à renommer, on suppose que K est le noyau de l'isogénie $E \to j_1$.

L'idée est alors d'utiliser la chaîne $j_{-1} \to E \to j_1$ pour relever la valeur propre modulo ℓ^2, sur le même principe qu'en 4.3. On utilise toutefois une version différente du relèvement de Hensel (voir ci-dessous).

Une amélioration possible. *Ici, il faudrait calculer une équation pour E_{-1} telle que l'isogénie $E_{-1} \to E$ est normalisée. Comme on ne dispose pas de cela, on calcule actuellement une équation (quelconque) pour E_{-1}, puis on recalcule une équation pour E et E_1 de sorte que les isogénies soient normalisées. Le calcul de K est donc perdu au lieu d'être réutilisé.*

En pratique, on connaît aussi l'autre valeur propre $\frac{q}{v}$ du Frobenius : on connaît donc également le degré d' des facteurs de l'*autre* sous-groupe rationnel d'ordre ℓ de E. Dans le cas où $d' < d$, on utilise la chaîne renversée $j_1 \to E \to j_{-1}$.

Relèvement de Hensel. On se pose ici la même question de relèvement qu'en 4.3. Toutefois, ℓ n'est pas nécessairement petit, donc on remplace l'al-

gorithme naïf par une méthode à la Shanks, analogue aux méthodes présentées en 4.2. On écrit

$$v = v_0 + k\ell \quad \text{avec } 0 \leq k \leq \ell - 1.$$

On tente d'écrire $k = \frac{i}{j}$ avec $0 \leq i \leq I$, $1 \leq j \leq J$: c'est possible dès que $IJ > \ell$. Si P désigne le point générique et F le Frobenius, on doit alors résoudre

$$F = [v_0 + i/j]P$$

c'est à dire

$$[j](F - [v_0]P) = [i]P.$$

On utilise donc l'analogue de la méthode des polynômes de division en remplaçant F par $F - [v_0]P$. Pour cela, il faut ramener ce point projectif à la situation $z = 1$, ce qui coûte une division. Cela suppose que $F - [v_0]P$ est non nul ; dans le cas contraire, on conclut immédiatement que $v = v_0$.

Une amélioration possible. *Actuellement, on prend* $I = J = \left\lceil \sqrt{\ell} \right\rceil$. *Comme le calcul des multiples est plus cher pour* $F - [v_0]P$ *que pour le point générique, il faudrait plutôt choisir* $I > J$ *comme précédemment.*

5 La méthode d'Atkin

Comme annoncé dans la section 3, on peut utiliser la factorisation d'une équation modulaire de niveau ℓ évaluée en une courbe E pour obtenir des informations sur la trace du Frobenius modulo ℓ. Elles ne sont toutefois que partielles : on obtient seulement un ensemble de valeurs possibles pour la trace. On expose dans cette section les algorithmes utilisés pour le calcul de cet ensemble de valeurs.

5.1 Calcul du degré de décomposition

On suppose à présent que l'on se trouve dans le cas 4 de la proposition 3.1. On notera pour simplifier $A = A_\ell(j(E), X)$. La factorisation de A sur $\mathbb{F}_q$ est donc

$$A = Q_1 \cdots Q_n$$

où chaque Q_i est de degré r, pour un certain $r > 1$ à déterminer. On insiste sur le fait que l'on ne calcule pas cette factorisation : on entre dans la méthode d'Atkin au moment où l'on constate que A n'a pas de facteur linéaire. Selon la proposition 3.1, déterminer r suffit pour donner un ensemble restreint de valeurs possibles pour la trace du Frobenius modulo ℓ.

L'idée utilisée pour calculer r est la suivante : r est le plus petit entier ≥ 1 tel que

$$X^{q^r} = X \mod A.$$

Selon 3.1, on peut construire une liste R de valeurs qui contient le r cherché : ce sont les diviseurs de $\ell + 1$ vérifiant

$$(-1)^{\frac{\ell+1}{r}} = \left(\frac{q}{\ell}\right).$$

Parfois, R peut être très petite, notamment lorsque $\ell + 1$ a peu de diviseurs, ou lorsque $\left(\frac{q}{\ell}\right) = -1$ et $\ell + 1$ est divisible par une « grande » puissance de 2.

Pour calculer les itérés du Frobenius, on utilise la remarque suivante : si $F_i = X^{q^i} \mod A$ et $F_j = X^{q^j} \mod A$, alors

$$F_i \circ F_j = X^{q^{i+1}} \mod A.$$

Cela provient directement du fait que la mise à la puissance q est une opération $\mathbb{F}_q$-linéaire. En pratique, on utilise l'algorithme de composition modulaire de Brent et Kung, disponible dans Flint et NTL.

Afin de trouver la bonne valeur de r en minimisant le nombre de compositions modulaires à effectuer, on cherche une chaîne d'additions parcourant tous les éléments de R. La solution retenue est la suivante. On trie tout d'abord R dans l'ordre croissant, et l'on traite chaque r l'un après l'autre en stockant les itérés F_i du Frobenius que l'on a calculés. Pour toute nouvelle valeur de r, on cherche le plus grand i tel que F_i est déjà calculé. Si $i < 2r$,

on stocke $F_{2i} = F_i \circ F_i \mod A$ dans la table, et l'on redemande le calcul de F_r. Sinon, on demande le calcul de F_{r-i} et l'on calcule ensuite $F_r = F_i \circ F_{r-i} \mod A$. Dès que l'un des F_r est égal à X, on stoppe le calcul et l'on renvoie cette valeur.

Un exemple : si $R = [2, 3, 7, 11, 50]$, on va effectuer

$$\begin{array}{ccc} 1+1 \to 2; & 2+1 \to 3; & 3+3 \to 6; \\ 6+1 \to 7; & 3+1 \to 4; & 7+4 \to 11; \\ 11+11 \to 22; & 22+22 \to 44; & 44+6 \to 50. \end{array}$$

Une amélioration possible. *Cette solution donne-t-elle le nombre minimal de compositions modulaires à effectuer pour parcourir toutes les valeurs de r ?*

5.2 Calcul des valeurs possibles pour la trace

Comme r est un diviseur de $\ell + 1$, donc de $\ell^2 - 1$, les racines primitives r-ièmes de l'unité sont des éléments de $\mathbb{F}_{\ell^2}$. Concrètement, on trouve un élément $\delta \in \mathbb{F}_\ell$ qui n'est pas un carré dans $\mathbb{F}_\ell$. On représente le corps $\mathbb{F}_{\ell^2}$ sous la forme $\mathbb{F}_\ell[\sqrt{\delta}]$. On cherche ensuite un générateur multiplicatif g dans cette extension : une racine primitive r-ième est alors $\zeta_0 = g^{\frac{\ell^2-1}{r}}$. Les autres racines primitives r-ièmes sont les ζ_0^i, pour tout $1 \leq i \leq r-1$ premier à r. Pour la recherche de δ et g, on utilise des méthodes naïves (on énumère les éléments jusqu'à trouver un non-carré ; pour déterminer si x est d'ordre multiplicatif $\ell^2 - 1$, on vérifie que $x^d \neq 1$ pour tout diviseur d de $\ell^2 - 1$).

Soit une racine primitive r-ième $\zeta = z_1 + z_2\sqrt{\delta}$, avec $z_1, z_2 \in \mathbb{F}_\ell$. Soit $\lambda = x_1 + x_2\sqrt{\delta}$ une des valeurs propres du Frobenius sur $E[\ell]$, qui est un élément de $\mathbb{F}_{\ell^2}$. L'autre valeur propre, conjuguée à λ, est supposée être $\mu = \zeta\lambda$. Alors, suivant [Gal07] :

$$q = \lambda\mu = x_1^2 - \delta x_2^2$$

et

$$\zeta = \frac{\lambda}{\mu} = \frac{x_1^2 + \delta x_2^2}{q} + \frac{2x_1x_2}{q}\sqrt{\delta}$$

d'où l'on déduit

$$x_1^2 = q(z_1 + 1)/2 \mod \ell.$$

Si cette quantité est un carré dans $\mathbb{F}_\ell$, on en calcule une racine carrée x_1, et on ajoute à la liste des traces possibles les quantités $2x_1$ et $-2x_1$. On vérifie également que

$$t^2 - 4q = 2q(z_1 - 1)$$

n'est pas un carré modulo ℓ, ce qui est équivalent à vérifier que

$$x_2^2 = q(z_1 - 1)/2\delta$$

est bien un carré modulo ℓ.

L'ensemble obtenu contient nécessairement la trace du Frobenius modulo ℓ : en effet, π_E^r est scalaire sur $E[\ell]$, ce qui implique

$$\lambda^r = \mu^r,$$

donc λ et μ diffèrent d'une racine r-ième de l'unité. Celle-ci est nécessairement primitive puisque le Frobenius n'a pas de droite vectorielle stable dans $E[\ell]$ définie sur une sous-extension stricte de $\mathbb{F}_{p^r}$.

Une amélioration possible. *Existe-t-il d'autres restrictions pour les valeurs possibles de t ? Toute réduction de la liste des candidats est un gain important lors du crible.*

6 La méthode de Schoof

Par beaucoup d'aspects, l'implémentation de la méthode de Schoof est similaire à celle de la méthode d'Elkies. Rappelons que l'on applique cette méthode à ℓ^k lorsque la courbe E ne dispose pas de sous-groupe cyclique d'ordre ℓ. L'entier k est choisi maximal tel que $2k$ soit inférieur à l'exposant idéal pour ℓ.

La méthode de Schoof fait intervenir trois étapes : le calcul du polynôme de division d'indice ℓ^k, le calcul du Frobenius et du Frobenius au carré sur le sous-groupe $E[\ell^k]$, et enfin le calcul de la trace du Frobenius sur $E[\ell]$.

Calcul du polynôme de division. On utilise ici le polynôme f_{ℓ^k} introduit à la section 4.2. Il s'agit exactement du polynôme de division si ℓ est impair, et si $\ell = 2$, il est avantageux d'utiliser ce polynôme à la fois univarié et de plus petit degré. On le calcule à partir des formules de récurrences données dans cette même section.

Une amélioration possible. *Actuellement, on calcule tous les polynômes de division d'indice 1 à ℓ^k à cette étape. Le coût supplémentaire est faible, mais facilement évitable.*

Calcul du Frobenius. On utilise ici la méthode optimisée présentée à la section 4.2 pour calculer $F_X = X^q$ et $F_Y = Y^{q-1}$ modulo f_{ℓ^k}. Une fois ce calcul effectué, on calcule le carré du Frobenius à l'aide d'une composition modulaire : plus précisément, on calcule

$$X^{q^2} = F_X \circ F_X \mod f_{\ell^k}, \quad Y^{q^2-1} = F_Y(F_Y \circ F_X) \mod f_{\ell^k}$$

par deux compositions modulaires et une multiplication.

Calcul de la trace. Rappelons que l'exposant idéal r pour ℓ vérifie $r \geq 2k$: l'entier ℓ^k est donc relativement petit. Pour cette raison, on utilise un algorithme naïf pour le calcul de la trace modulo ℓ^k. Si P, F, F_2 désignent respectivement le point générique, le Frobenius et le Frobenius au carré, on calcule

$$T = F_2 + [q]P$$

puis l'on cherche t tel que $[t]F = T$. Comme à la section 4.2, on utilise un test d'égalité au signe près qui permet de considérer les t compris entre 0 et $\frac{\ell^k-1}{2}$, pour $\frac{\ell^k-1}{4}$ additions sur la courbe en moyenne.

7 Le crible final

7.1 Principe du crible

Dans l'algorithme SEA, on considère de petits premiers ℓ_i les uns après les autres. Pour certains, on applique la méthode d'Elkies ou de Schoof : on obtient une unique possibilité t_i modulo $\ell_i^{r_i}$ pour la trace t du Frobenius (le plus souvent, $r_i = 1$). Pour d'autres, on applique la méthode d'Atkin : on obtient, pour chaque premier ℓ_i, un ensemble L_i de valeurs possibles pour t mod ℓ_i, et on pose $r_i = 1$. On arrête la collecte de données lorsque l'inégalité

$$\prod \ell_i^{r_i} > 4\sqrt{q}$$

est vraie.

Par le théorème chinois, on dispose alors de $N = \prod \mathrm{Card}(L_i)$ possibilités pour $t \mod \prod \ell_i^{r_i}$. Cela détermine uniquement la valeur de t dans $\mathbb{Z}$ vu les bornes de Hasse. Pour trouver la valeur de la trace, on peut procéder de la manière suivante : on choisit un point $P \in E(\mathbb{F}_q)$ aléatoirement. Pour chaque valeur candidate c, on calcule $[q+1-c]P$: si c est la bonne réponse, alors ce point doit être nul. En pratique, on obtient très souvent une unique valeur possible pour t. Le coût de cet algorithme est celui de N multiplications scalaires sur E par un entier de l'ordre de $\sqrt{q}$, si $[q+1]P$ a été calculé une fois pour toutes.

On peut toutefois faire mieux en adoptant une stratégie type pas de bébé–pas de géants. On sépare les données en trois parties :

— La partie « Elkies », qui donne $t_E,\ M_E$ tels que

$$t = t_E \mod M_E.$$

— Une première partie Atkin, qui donne M_1 et un ensemble L_1 tels que

$$t \in L_1 \mod M_1.$$

— Une seconde partie Atkin, qui donne M_2 et un ensemble L_2 tels que

$$t \in L_2 \mod M_2.$$

Les entiers M_E, M_1 et M_2 sont premiers entre eux. On sait donc que l'on peut écrire t sous la forme

$$t = t_E + M_E(r_1 M_2 + r_2 M_1) \tag{7}$$

pour certains r_1, r_2. Les valeurs possibles pour r_1 modulo M_1 (resp. r_2 modulo M_2) sont déterminées à partir de L_1, t_E et les entiers M_E, M_1, M_2 : pour chaque $t_1 \in L_1$, la valeur correspondante de r_1 est

$$r_1 = \frac{t_1 - t_E}{M_E M_2} \mod M_1.$$

De plus, si $0 \leq t_E < M_E$ et si l'entier r_1 est choisi tel que

$$|r_1| \leq \frac{M_1}{2}, \tag{8}$$

alors on a nécessairement $|r_2| \leq M_2$: en effet,

$$\begin{aligned} |r_2| &\leq \frac{|t| + |t_E| + M_E M_2 |r_1|}{M_E M_1} \\ &< \frac{2\sqrt{q}}{M_E M_1} + \frac{1}{M_1} + \frac{M_2}{2} \\ &\leq M_2 + \frac{1}{M_1} \end{aligned}$$

puisque $M_E M_1 M_2 > 4\sqrt{q}$ par hypothèse. Cela conclut car toutes les quantités sont entières.

Par conséquent, on va procéder de la manière suivante. On calcule les valeurs possibles pour les entiers r_1, r_2 satisfaisant les congruences (8) et les bornes

$$|r_1| \leq \frac{M_1}{2}, \quad |r_2| \leq M_2.$$

On obtient deux listes R_1, R_2. Le point P étant choisi au préalable, on calcule $P_1 = [q+1-t_E]P$ et $Q_1 = [-M_2 M_E]P$. Puis, pour chaque $r_1 \in R_1$, on calcule $P_1 + [r_1]Q_1$ et l'on stocke le couple $(r_1, P_1 + [r_1]Q_1)$ dans une table. Cette étape représente les pas de bébé.

Une fois cela fait, on calcule $Q_2 = [M_1 M_E]P$. Pour chaque $r_2 \in R_2$, on calcule $[r_2]Q_2$, et l'on recherche ce point dans la table précédente. Si le couple (r_1, r_2) donne une collision, alors c'est un candidat.

En pratique, afin de bien contrôler l'espace mémoire utilisé, on ne stocke pas des entiers multiprécision. On stocke les r_1 sous la forme d'entiers de 64 bits, et l'on calcule un haché de Q_1 sous la forme d'un entier 32 bits, que l'on stocke dans la table. Une collision de hachés n'implique pas forcément une collision des points : on effectue une multiplication scalaire pour vérifier que (r_1, r_2) est bien une solution.

Une amélioration possible. *Cet approche ne fonctionne pas si les valeurs de r_1, r_2 dépassent 64 bits, ce qui peut arriver en pratique si l'on dispose de beaucoup de premiers de type Atkin pour lesquels la liste L_i est courte. Cela engendre un « abort ».*

On obtient en général un seul candidat, et la valeur de la trace est donnée par l'égalité (7). Dès que l'on trouve un deuxième candidat, on arrête l'algorithme SEA en déclarant que le cardinal n'a pas été trouvé.

Une amélioration possible. *Dans ce cas (non rencontré malgré de nombreux essais), on pourrait soit retenter le crible avec un nouveau point P (l'échec peut s'expliquer par le fait que P était un point de torsion de petit ordre), ou encore mieux, ajouter des données Elkies ou Atkin avant de retenter le crible.*

Le coût de cet algorithme est celui de $n_1 + 2n_2$ multiplications scalaires sur E par un entier de l'ordre de $\sqrt{q}$, où n_1 (resp. n_2) désigne la taille de R_1 (resp. R_2). On a $n_1 n_2 = N$: on voit ici le gain apporté par rapport au premier algorithme.

Lors de la séparation des données, on tente donc d'atteindre $n_1 \simeq 2n_2$. Concrètement, on trie les données de la méthode d'Atkin (ℓ_i, L_i) dans l'ordre de $\mathrm{Card}(L_i)$ décroissant. On les répartit ensuite dans l'ordre : on ajoute le prochain élément à la famille 1 si l'égalité $n_1 < 2n_2$ est actuellement vraie, sinon on l'ajoute à la famille 2.

Une amélioration possible. *Cette stratégie de répartition semble convenir dans la plupart des cas en pratique, mais une garantie théorique serait souhaitable.*

7.2 Calcul des multiples d'un point

Dans le crible exposé en 7.1, le coût principal est celui des multiplications scalaires : étant donnée une liste d'entiers R, il s'agit de calculer tous les

$$[r]Q \quad \text{ou} \quad P + [r]Q$$

pour des points fixés $P, Q \in E(\mathbb{F}_q)$. Pour cela, on peut éviter de calculer une multiplication scalaire pour chaque r par la technique suivante, inspirée de [Atk92].

On trie R dans l'ordre croissant, et l'on calcule l'intervalle maximal $I > 0$ entre deux éléments successifs de R. On choisit des entiers $b > 1,\ n \geq 1$ tels que $b^n > I$. Par conséquent, si r' succède à r dans R, on peut écrire

$$r' - r = \sum_{i=0}^{n-1} a_i b^i$$

avec $0 \leq a_i < b$.

On précalcule les multiples $[kb^i]P$ pour tous $0 \leq k < b$ et $0 \leq i < n$. On écrit donc

$$[r']Q = [r]Q + [a_0]P + [a_1 b]P + \cdots + [a_{n-1} b^{n-1}]P,$$

ce qui permet de calculer $[r']Q$ en n additions de points sur la courbe E. La seule multiplication scalaire effectuée, en dehors des précalculs, est l'initialisation de $[r]Q$ pour le premier élément de la liste R.

En pratique, on se fixe au préalable une borne B qui est la taille maximale des tables que l'on accepte de précalculer. On choisit ensuite n minimal tel que $B^n > I$, puis on pose $b = \left\lceil \sqrt[n]{I+1} \right\rceil$. Le plus souvent (voir section 8), n est inférieur à 3.

7.3 Limitations pratiques

Le crible, malgré les optimisations possibles, est un algorithme exponentiel. En pratique, il faut éviter à tout prix de se lancer dans un crible dont le coût est hors de contrôle.

On se donne donc une borne S pour la taille maximale du crible, c'est à dire la quantité N de la section 7.1. En pratique, cette borne est dépassée lorsque l'on collecte toutes les informations que l'on peut obtenir par la méthode d'Atkin.

Lors de la collecte de données de type Atkin, on fait donc le choix suivant : lorsque N devient supérieur à S, on jette le premier de type Atkin le moins intéressant (ou plusieurs), jusqu'à ce que l'inégalité $N \leq S$ soit vraie.

Pour estimer l'intérêt d'une donnée (ℓ_i, L_i), on peut faire la remarque suivante : on veut atteindre une certaine valeur pour $\prod \ell_i$, sans dépasser une certaine valeur pour $\prod \mathrm{Card}(L_i)$. Par conséquent, on peut quantifier l'intérêt de (ℓ_i, L_i) par la quantité

$$\frac{\log \ell_i}{\log \mathrm{Card}(L_i)}.$$

On supprime alors la donnée dont l'intérêt est le plus faible.

Une autre stratégie serait de s'interdire d'appliquer la méthode d'Atkin dès que N devient proche de S. On choisit de rejeter cette solution, qui ne permet pas de profiter des grands premiers de type Atkin qui offrent parfois une grande quantité d'information sur la trace.

On peut toutefois éviter de consacrer trop de temps sur les grands premiers de type Atkin : lorsque le degré de décomposition r est trop grand, l'information obtenue est de peu d'intérêt, et r grandit habituellement avec ℓ_i. Par conséquent, on définit préalablement une borne pour les valeurs de ℓ_i et r à considérer (voir section 8).

Une amélioration possible. *Pour un premier ℓ donné, lorsque l'on applique la méthode d'Atkin, on peut prévoir la répartition statistique attendue pour le degré de décomposition r. En effet, on constate que les valeurs propres du Frobenius sont équitablement réparties dans $\mathbb{F}_{\ell^2}$, en tenant compte des restrictions (4) pour la valeur de r.*

Cela peut-il être utilisé pour proposer des limitations pertinentes aux calculs qui dépendraient de ℓ et q ?

8 Étude des performances pratiques

Dans cette section, on mesure l'impact des différents paramètres que l'on peut ajuster dans l'algorithme SEA tel qu'il a été décrit dans la section précédente. Concrètement, on travaille avec un fichier de « stratégie » qui contient ces différents paramètres et que l'on donne en entrée du programme.

8.1 Résumé des points de contrôle sur l'algorithme

L'exposant idéal (section 3.4). Il s'agit, pour un premier ℓ donné, de déterminer un entier k tel que l'on veut manipuler des polynômes de degré jusqu'à ℓ^k. Le coût est ensuite polynomial en ℓ^k, et l'on souhaite aboutir à un algorithme polynomial en $\log q$: on se donne donc un réel $f \geq 0$, et l'on choisit k maximal tel que

$$\ell^k \leq f \log q.$$

(avec un minimum de $k = 1$). Dans la stratégie, f est donné par le champ `ideal_exp_mulf`.

On contrôle également l'entrée dans le relèvement modulo ℓ^2 lorsque ℓ est Elkies simple (section 4.4). Pour cela, on se donne un second réel g, et l'on accepte de calculer modulo un polynôme de degré ℓd si

$$\ell d \leq g f \log q$$

où f est donné ci-dessus. g est appelé `second_step_mulf`.

Le crible (section 7). Tout d'abord, on se donne une borne pour la taille du crible $N = \prod \mathrm{Card}(L_i)$, donnée par le champ `max_sieve_size`. Par ailleurs, on peut contrôler la taille des tables de multiples (la borne B de la section 7.2) : c'est le rôle du champ `max_sieve_precomp`. Cela contrôle le coût du crible en lui-même.

On peut également contrôler les calculs effectués dans la méthode d'Atkin préliminaire au crible. On cesse de considérer les premiers d'Atkin à partir de la borne `max_atkin_prime`, et on ne considère pas de degré de décomposition au-dessus de la borne `max_atkin_splitting_degree`.

Le calcul de la valeur propre (section 4.2) Il s'agit d'une part du seuil de passage entre l'algorithme naïf et la méthode des pas de bébés–pas de géants de Shanks (`eigenvalue_bsgs_threshold`) et du seuil entre ce dernier algorithme et la méthode utilisant les polynômes de division (`eigenvalue_divpolys_threshold`).

D'autre part, on peut contrôler la répartition entre le Frobenius et le point générique dans les deux méthodes à la Shanks : on choisit un réel h tel que l'on met $h\sqrt{N}$ sur le point générique, et $\sqrt{N}/h$ sur le Frobenius. Le paramètre h est donné par le champ `eigenvalue_bsgs_ratio`.

8.2 Recherche de la stratégie optimale

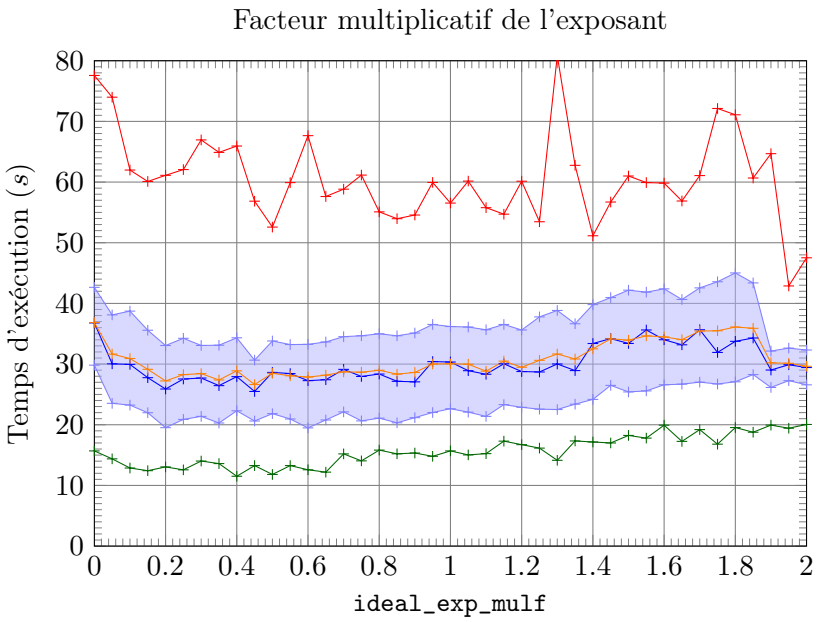
L'objectifs des résultats expérimentaux qui suivent est de déterminer de « bonnes valeurs » pour les paramètres détaillés ci-dessus dans le cas de corps finis **dont l'ordre mesure 512 bits**. Plus précisément, on tente de valider expérimentalement la stratégie empirique suivante :

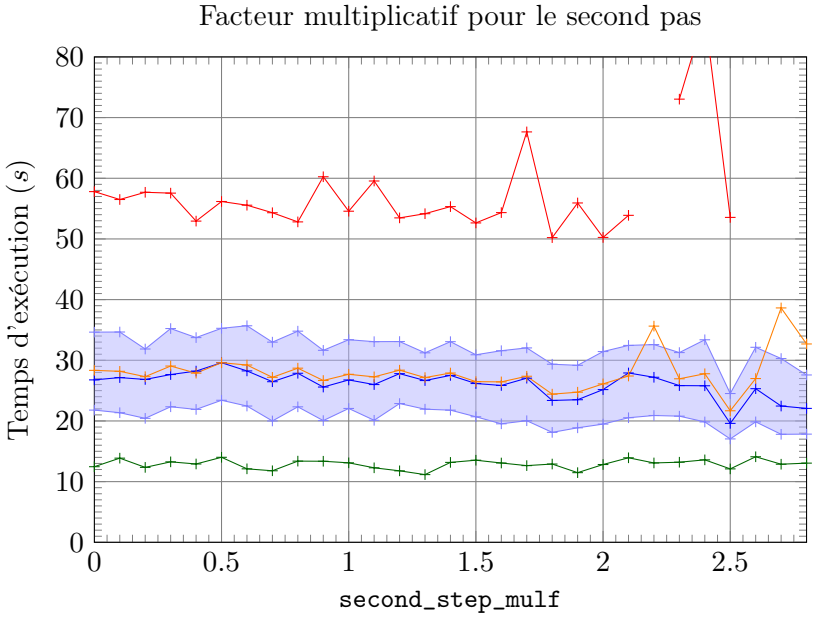
— `ideal_exp_mulf` $= 0.3$
— `second_step_mulf` $= 1.0$
— `max_sieve_size` $= 5 \cdot 10^8$
— `max_sieve_precomp` $= 10000$
— `max_atkin_prime` $= 250$
— `max_atkin_splitting_degree` $= 60$
— `eigenvalue_bsgs_threshold` $= 20$
— `eigenvalue_divpolys_threshold` $= 20$
— `eigenvalue_bsgs_ratio` $= 1.15$.

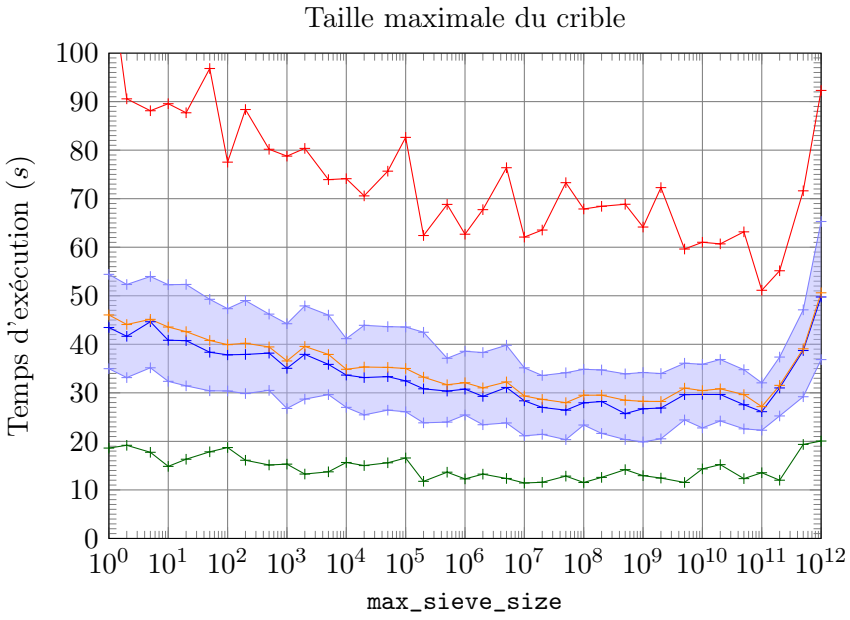
On fait alors varier chacun des champs indépendamment dans une certaine plage de valeurs. Les autres paramètres ayant leurs valeurs par défaut ci-dessus, avec les exceptions suivantes :

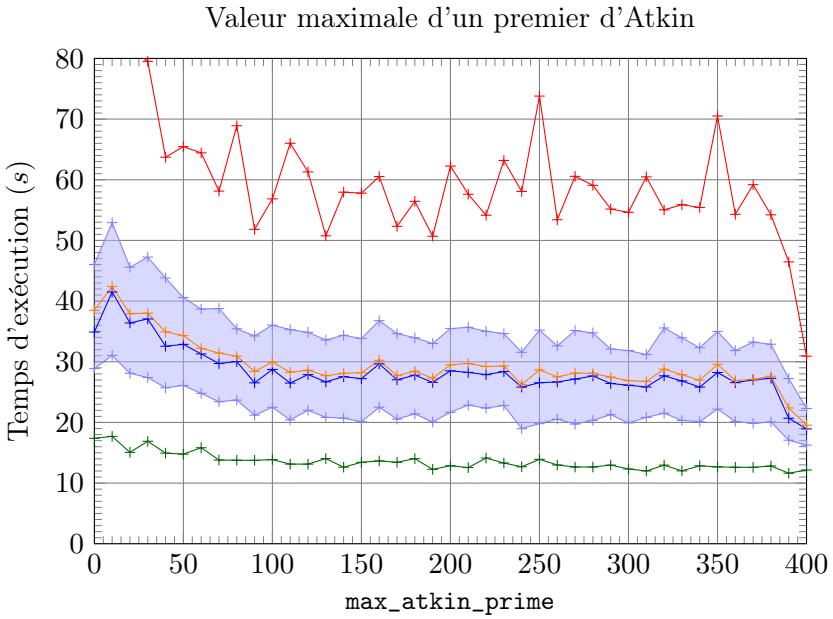
— Pour la mesure de `max_sieve_size`, `max_atkin_prime` et `max_atkin_splitting_degree`, chacun des deux autres paramètres est mis à une grande valeur.
— Pour la mesure de `eigenvalue_divpolys_threshold`, la quantité `eigenvalue_bsgs_threshold` est mis à une valeur faible, et vice-versa.

Pour chacune des stratégies obtenues, on choisit 200 courbes au hasard sur des corps finis variables de 512 bits, et l'on mesure le temps d'exécution du programme sur ces courbes. On représente le minimum (en vert), la médiane (en bleu) et une zone entre 25% et 75% comprenant donc la moitié des courbes (bleutée), le maximum (en rouge) et enfin le temps d'exécution moyen (en orange). Certaines valeurs maximales aberrantes ont été otées du graphique. Les mesures ont été réalisées sur une machine munie d'un processeur Intel Core i3-6100 @ 3.70GHz.

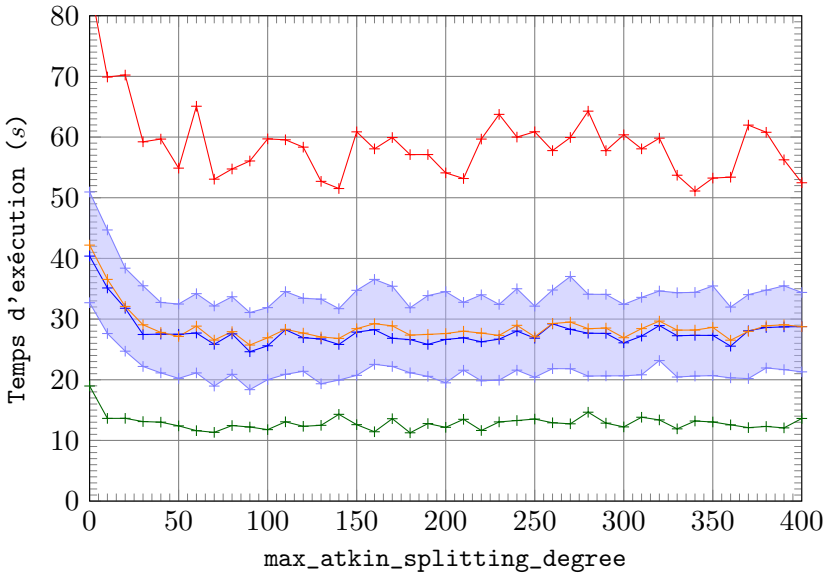

Facteur multiplicatif de l'exposant
Temps d'exécution (s)
0
10
20
30
40
50
60
70
80
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
ideal_exp_mulf


Facteur multiplicatif pour le second pas
Temps d'exécution (s)
0
10
20
30
40
50
60
70
80
0
0.5
1
1.5
2
2.5
second_step_mulf

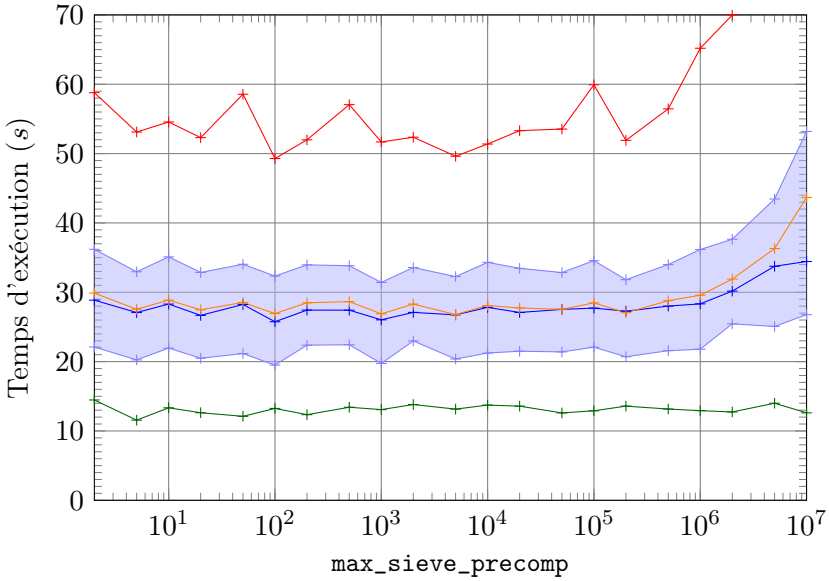

Taille maximale du crible
Temps d'exécution (s)
max_sieve_size


Valeur maximale d'un premier d'Atkin
Temps d'exécution (s)
max_atkin_prime

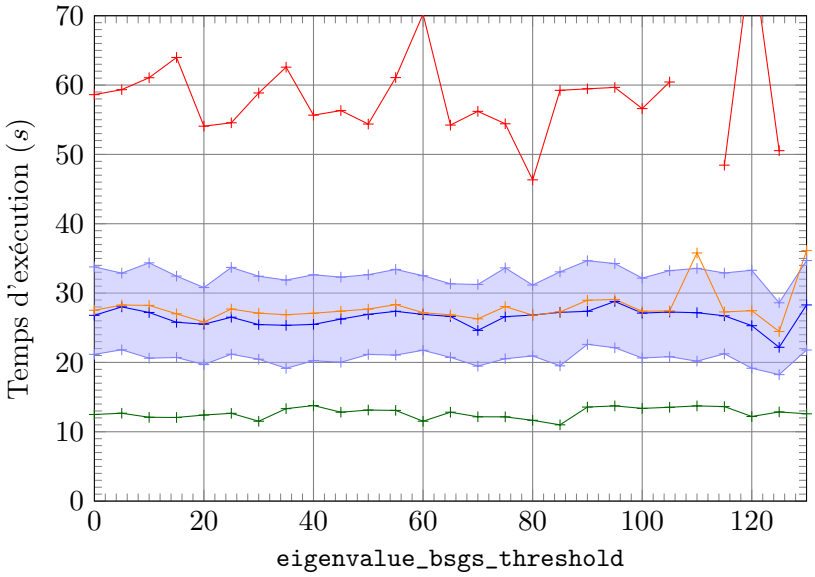
Degré de décomposition maximal pour les premiers d'Atkin



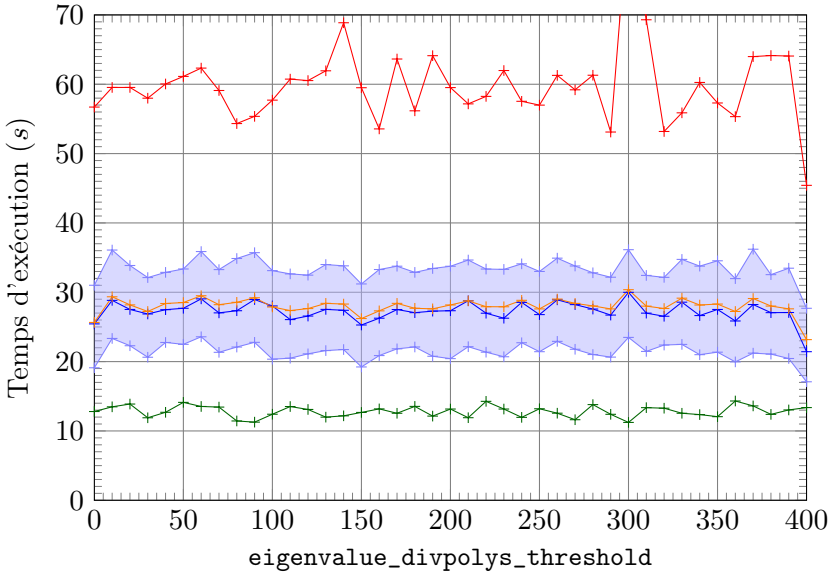
Taille des précalculs lors du crible

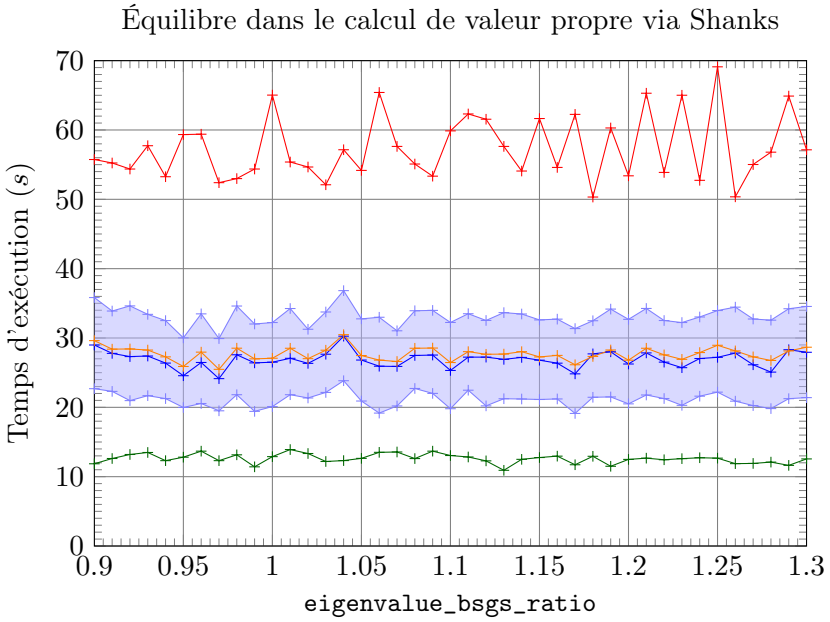


Seuil d'utilisation de BSGS pour le calcul de valeur propre



Seuil d'utilisation des polynômes de division





Conclusions. Les paramètres contrôlant le crible font apparaître le même motif : à des valeurs faibles, le crible n'est pas utilisé autant qu'il devrait, ce qui se traduit par une recherche plus longue de premiers d'Elkies. Cela a un impact notable sur les courbes les moins favorables. On voit qu'il est important de limiter la taille du crible : 10^9 semble une bonne valeur pour celle-ci. On peut limiter le degré de décomposition maximal à environ 80, en revanche la borne `max_atkin_prime` ne semble pas pertinente. À l'intérieur du crible lui-même, il est important de limiter la taille des tables précalculées, et `max_sieve_precomp` $= 1000$ semble convenable.

Le facteur `ideal_exp_mulf` est important pour tirer le meilleur parti des courbes favorables : 0.4 semble un bon choix. Le facteur `second_step_mulf` est plus anecdotique, une valeur de 1.3 semble accélérer les performances sur les meilleures courbes, mais une valeur supérieure à 2 engendre des coûts parfois démesurés.

Enfin, les paramètres contrôlant le calcul de valeur propre ont peu d'influence. Les seuils de $\ell \geq 30$ pour l'algorithme des pas de bébés et de $\ell \geq 90$ pour l'utilisation des polynômes de division, associés à un `ratio` de 1.13 environ, semblent satisfaisants.

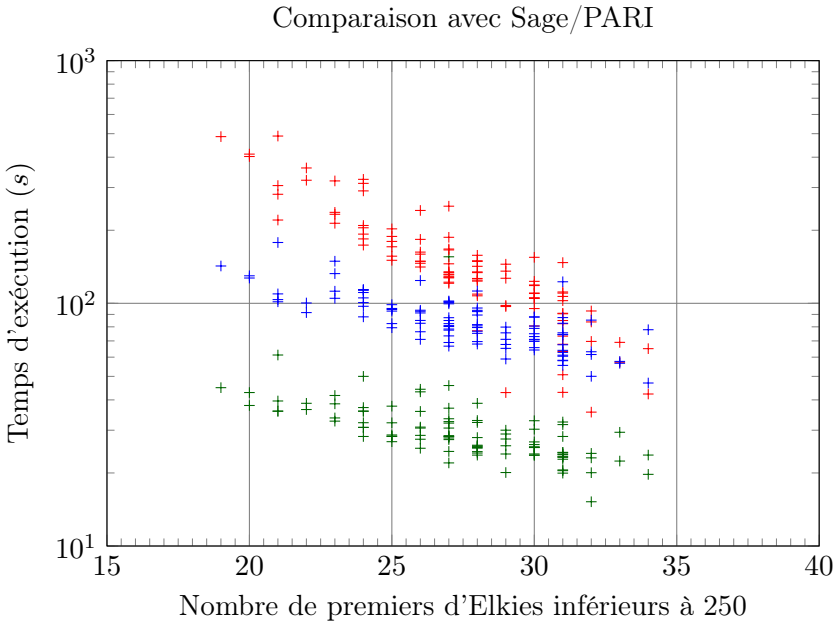
La stratégie définitive est donc la suivante :

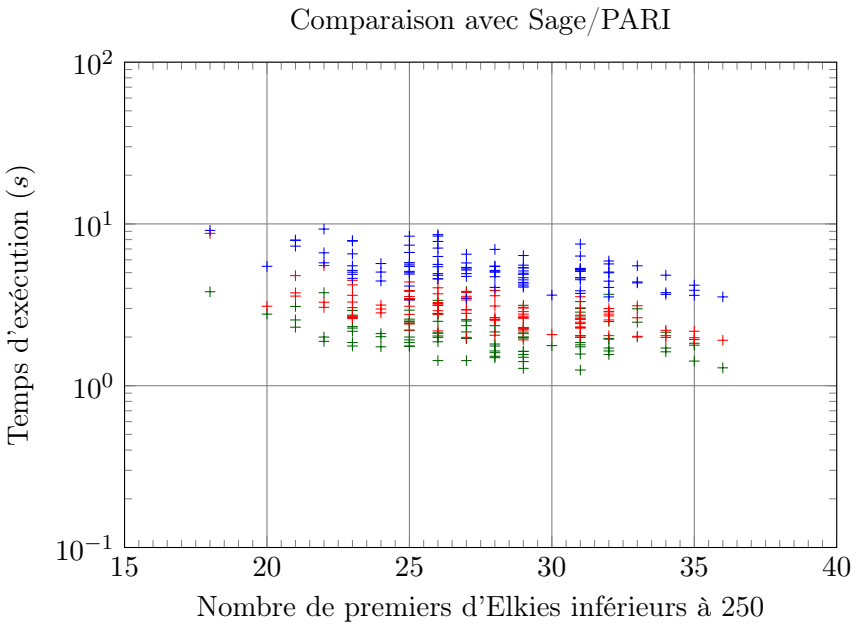
— `ideal_exp_mulf` $= 0.4$
— `second_step_mulf` $= 1.3$
— `max_sieve_size` $= 10^9$
— `max_sieve_precomp` $= 1000$
— `max_atkin_prime` $= 500$ (non utilisé)
— `max_atkin_splitting_degree` $= 80$

— `eigenvalue_bsgs_threshold` = 30
— `eigenvalue_divpolys_threshold` = 90
— `eigenvalue_bsgs_ratio` = 1.13.

8.3 Comparaison avec d'autres systèmes

On tente d'établir une comparaison entre notre implémentation et les fonctionnalités de comptage de points proposés par d'autres systèmes de calcul formel. La comparaison porte sur 100 courbes définies sur un corps premier de 512 bits choisies aléatoirement, et l'on représente le temps d'exécution en fonction du nombre de premiers d'Elkies de chaque courbe entre 2 et 250 pour notre implémentation (en vert), le système de calcul formel Sage qui utilise PARI [PAR17] pour ce calcul (en rouge), et le système Magma (en bleu).





Les mesures ont été réalisées sur une machine munie d'un processeur Intel Xeon E5-2680v3@2.5GHz.

La comparaison avec Sage/PARI est sans équivoque : l'algorithme SEA implanté en C est plus rapide que ce dernier d'un facteur 2 à 20. La dépendance de PARI envers le nombre de premiers d'Elkies est accentuée : cet écart pourrait s'expliquer en partie par le fait que PARI n'utilise pas suffisamment le crible d'Atkin. Magma est battu d'un facteur 2 à 3 environ.

L'avantage s'estompe pour des corps de 256 bits. Cela reflète les choix faits dans notre implémentation : par exemple, éviter les divisions de polynômes à tout prix (section 4.2) n'est sans doute pas rentable dans ce cadre. On mentionne toutefois que l'analyse de la meilleure stratégie n'a pas été faite pour 256 bits, ce qui pourrait apporter une amélioration mineure au temps d'exécution.

9 Description de l'implémentation

Cette section tient lieu de documentation « à la Flint » de l'implémentation de l'algorithme SEA en langage C : on donne la signature et une courte description des fonctions disponibles, en faisant références aux algorithmes pertinents. On donne également l'architecture des sources afin de faciliter la recherche dans cette documentation.

9.1 Architecture du projet

Le programme SEA se présente sous la forme d'une arborescence contenant entre autres des fichiers source (`./src/`), de données relatives aux polynômes modulaires (`./data/`), de fichiers de stratégie (`./tune/`) et de test (`./test/`) et enfin un répertoire pour les binaires (`./bin/`). Il peut être basé soit sur la bibliothèque Flint [HJP17], soit sur la bibliothèque NTL [Sho17], qui doit être fournie dans le dossier `./lib/` ou aux endroits standard.

La compilation est assurée par un makefile : la commande

```
make
```

déclenche la compilation basée sur NTL (par défaut) sans tests de correction, tandis que

```
make flint=true debug=true
```

compile le programme en se basant sur Flint avec ces tests. La commande `make` répond aux cibles standard `all`, `check` (voir ci-dessous), `clean` et `proper`. Dans le cas de NTL, qui est une librairie C++, la compilation des sources se fait en tant que code C++.

Deux versions de SEA peuvent être compilées : l'une concerne uniquement les corps finis premiers $\mathbb{F}_p$, l'autre peut traiter des corps non premiers mais n'est pas optimisée pour les corps premiers. La première est choisie par défaut ; pour choisir la seconde, il faut ajouter l'option

```
prime_field=false
```

à la commande `make`.

L'implémentation de l'algorithme SEA peut être découpée par thèmes : algorithme global, méthode d'Atkin, arithmétique de base, etc. La figure 9 représente les liens de dépendance entre les principaux blocs, les parties supérieures utilisant les parties inférieures (si un bloc A dépend d'un bloc B, il est implicite que A dépend également de toutes les dépendances de B). Pour chaque bloc, on indique le dossier qui contient le bloc source correspondant ainsi que la section qui en décrit les fonctions principales.

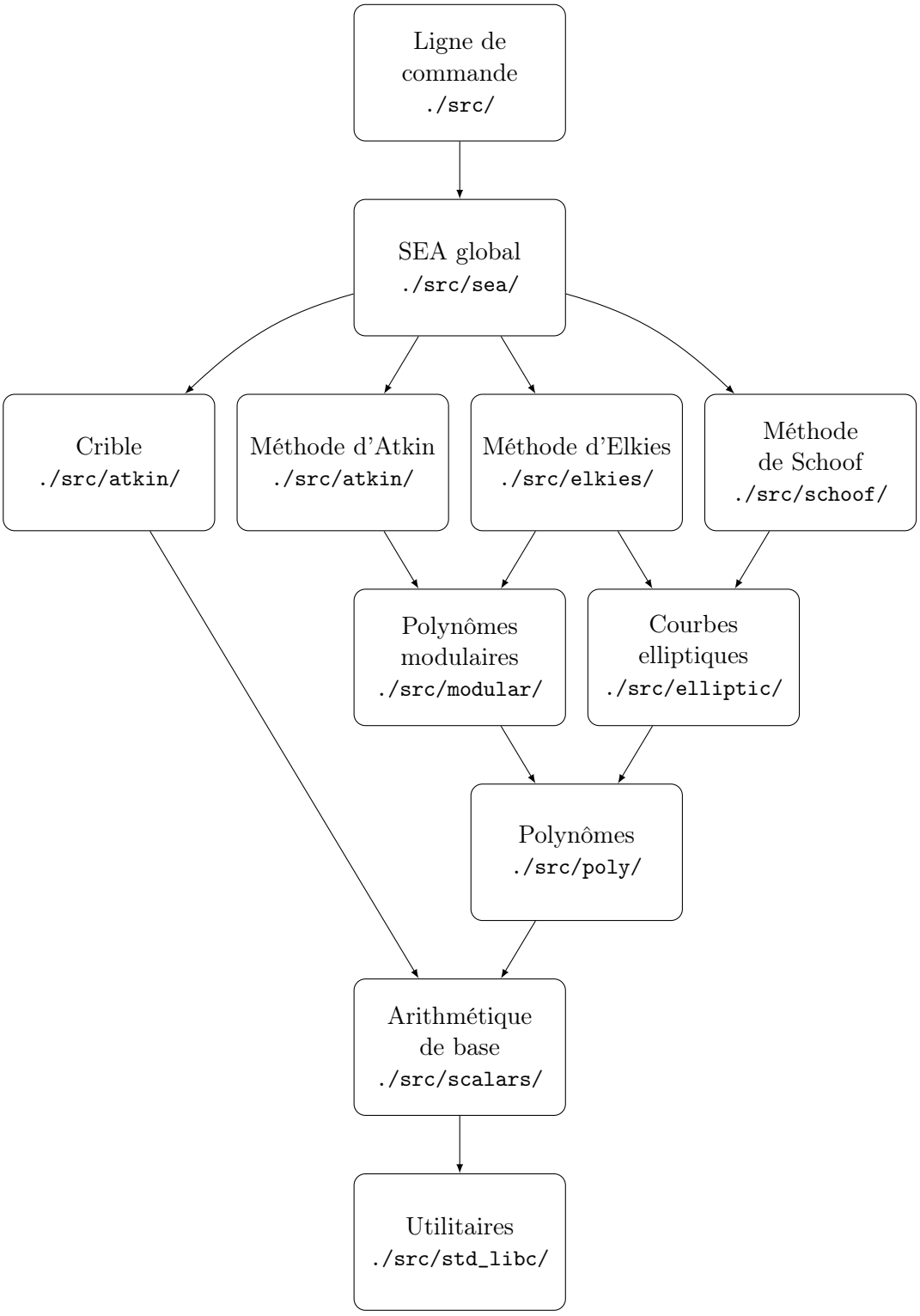


FIGURE 9 – Architecture simplifiée du code source

9.2 Contenu de `make check`

À l'appel de `make check`, plusieurs programmes de test sont compilés et exécutés. Le principe est d'appeler les fonctions testées, puis de comparer les résultats avec les valeurs correctes écrites en dur dans le test. Cela s'ajoute aux tests de validité internes à l'algorithme SEA. Voici les différents blocs testés, du plus bas niveau au plus haut niveau :

- `test_arith` teste les liens vers la bibliothèque sous-jacente pour l'arithmétique de base et les polynômes. Aucun test de validité n'est réellement fait : l'utilisateur est invité à exécuter également les programmes de test propres à la bibliothèque.
- `test_roots` teste le calcul de racines d'un polynôme via l'algorithme de Cantor–Zassenhaus « maison ».
- `test_modular` teste la lecture et l'évaluation des équations modulaires.
- `test_elliptic` teste les opérations sur une courbe elliptique pour les points génériques d'un sous-groupe ainsi que sur le corps $\mathbb{F}_q$.
- `test_kernel` teste le calcul de l'équation normalisée et du polynôme de noyau dans la méthode d'Elkies.
- `test_eigenvalue` teste les algorithmes de calcul de la valeur propre une fois le polynôme de noyau fixé.
- `test_volcano` teste la méthode d'Elkies étendue, avec les calculs dans le graphe d'isogénies et le relèvement de la valeur propre.
- `test_atkin` teste le calcul des traces dans la méthode d'Atkin ainsi que le crible qui suit.
- `test_sea100`, `test_sea256` et `test_sea512` testent l'algorithme SEA global sur un corps respectivement de 100, 256 et 512 bits.

La commande `make check` est équivalente à

```
make check data_path=./data
```

et la valeur de `data_path`, qui indique où chercher les équations modulaires nécessaires à certains tests, peut être modifiée par l'utilisateur.

Références

[Atk88] A. O. L. Atkin: *The Number of Points on an Elliptic Curve Modulo a Prime (1)*, 1988. www.lix.polytechnique.fr/Labo/Francois.Morain/AtkinEmails.

[Atk92] A. O. L. Atkin: *The Number of Points on an Elliptic Curve Modulo a Prime (2)*, 1992. www.lix.polytechnique.fr/Labo/Francois.Morain/AtkinEmails.

[Atk93] A. O. L. Atkin: *Several public email messages*, 1991-1993. www.lix.polytechnique.fr/Labo/Francois.Morain/AtkinEmails.

[BLS12] R. Bröker, K. Lauter et A. V. Sutherland: *Modular polynomials via isogeny volcanoes*. Mathematics of Computation, 81 :1201–1231, 2012.

[BMSS08] A. Bostan, F. Morain, B. Salvy et É. Schost: *Fast algorithms for computing isogenies between elliptic curves*. Mathematics of Computation, 77 :1755–1778, 2008.

[CDM96] J.-M. Couveignes, L. Dewaghe et F. Morain: *Isogeny cycles and the Schoof–Elkies–Atkin algorithm*, avril 1996.

[Cox89] D. A. Cox: *Primes of the form x^2+ny^2*. Wiley-Interscience, 1989.

[Elk97] N. D. Elkies: *Elliptic and modular curves over finite fields and related computational issues*, mars 1997. Preprint.

[Gal07] B. Galin: *Schoof–Elkies–Atkin algorithm*. Mémoire de maîtrise, Stanford University, décembre 2007.

[GM06] P. Gaudry et F. Morain: *Fast algorithms for computing the eigenvalue in the Schoof–Atkin–Elkies algorithm*. Dans J.-G. Dumas (rédacteur) : *ISSAC '06 : Proceedings of the 2006 international symposium on symbolic and algebraic computation*, pages 109–115. ACM Press, juillet 2006.

[Gra17] T. Granlund: *The GNU Multiple Precision Arithmetic Library*, 2017. Version 6.1.2, https://gmplib.org.

[HJP17] W. Hart, F. Johansson et S. Pancratz: *FLINT : Fast Library for Number Theory*, 2017. Version 2.5.2, http://flintlib.org.

[Koh96] D. Kohel: *Endomorphism rings of elliptic curves over finite fields*. Thèse de doctorat, University of California, Berkeley, décembre 1996.

[Mor95] F. Morain: *Calcul du nombre de points sur une courbe elliptique dans un corps fini : aspects algorithmiques*, 1995.

[Nek16] J. Nekovar: *Algèbre 2*, 2013–2016. Notes de cours disponibles à https://webusers.imj-prg.fr/~jan.nekovar.

[PAR17] The PARI Group, Univ. Bordeaux: *PARI/GP version* `2.8.0`, 2017. http://pari.math.u-bordeaux.fr/.

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

[Sch95] R. Schoof: *Counting points on elliptic curves over finite fields.* Journal de théorie des Nombres de Bordeaux, 7(1) :219–254, 1995.

[Sho17] V. Shoup: *NTL : a library for doing number theory*, 2017. Version 10.5.0, shoup.net/ntl.

[Sil86] J. H. Silverman: *The Arithmetic of Elliptic Curves.* Springer-Verlag, 1986.

[Sil94] J. H. Silverman: *Advanced Topics in the Arithmetic of Elliptic Curves.* Springer-Verlag, 1994.

[Sut10] A. V. Sutherland: *Database of classical modular polynomials*, 2010. math.mit.edu/~drew.

[vzGG03] J. von zur Gathen et J. Gerhard: *Modern Computer Algebra.* Cambridge University Press, 2003.

[Wat69] W. C. Waterhouse: *Abelian varieties over finite fields.* Ann. Sci. Éc. Norm. Sup., 4 :521–560, 1969.