Université de Bordeaux

Compter les points sur une courbe elliptique

Rapport TER

Master Cryptologie et sécurité informatique

Jérémie Coulaud

Tuteur : M. Jean-Marc Couveignes

2018-2019

Table des matières

1 Introduction

La cryptographie sur courbe elliptique introduite en 1985 a révolutionné la cryptographie à clé publique permettant une alternative efficace aux classiques RSA et Diffie Hellman. Les courbes elliptiques sont énormément utilisées pour les signatures (ECDSA), notamment pour le Bitcoin, et supportées dans la majorité des applications TLS, SSH. La sécurité d'un cryptosystème basé sur les courbes elliptiques dépend du nombre de points rationnels de la courbe. Il est donc intéressant de trouver des algorithmes efficaces pour déterminer ce nombre dans le but de construire des courbes elliptiques utilisables en cryptographie.
Le but de ce rapport est de présenter des algorithmes pour compter les points sur une courbe elliptique et de les implémenter pour tester leur efficacité. Après une brève introduction à la théorie de base des courbes elliptiques, on va décrire l'algorithme de Schoof, qui est polynomial mais peu efficace en pratique. Dans la suite on va essayer de comprendre les améliorations faites à cet algorithme par Atkins et Elkies conduisant à un algorithme toujours polynomial mais beaucoup plus efficace.
L'ensemble des codes est disponible sur le github `https://github.com/bialx/Elliptic-Curve-`
Le langage choisi est SAGE, basé sur Python. Il contient un large éventail de fonctions mathématiques prédéfinies, notamment pour les courbes elliptiques ce qui permet de se concentrer sur les algorithmes et non sur une implémentation d'objets mathématiques.

2 Introduction aux courbes elliptiques

On se contentera de donner une explication succincte de ce qu'est une courbe elliptique ainsi que différentes propriétés associées.

Définition 1. *On définit une courbe elliptique sur un corps K dans un plan par une équation de Weierstrass de la forme :*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Les coefficients $a_{1\leqslant i\leqslant 6}$ sont des éléments du corps K.

Dans le cas où le corps K a une caractéristique différente de 2 ou 3 on peut, via des changements de variable se ramener à l'équation courte suivante :

$$y^2 = x^3 + ax + b$$

Définition 2. *Soit $E : y^2 = x^3 + ax + b$ une courbe sur K on pose :*

$$\Delta = -16 * (4a^3 + 27b^2) \quad et \quad j(E) = \frac{(-48a)^3}{\Delta}$$

Δ est appelé le discriminant de E, et $j(E)$ son j-invariant. La courbe E est une courbe elliptique si et seulement si $\Delta \neq 0$.

Définition 3. *L'ensemble des points de la courbe E est noté $E(K)$ avec :*

$$E(K) = \left\{(x, y) \in \mathbb{K} \,|\, y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 = 0_E\right\} \cup O$$

Où O est le point à l'infini

On peut définir une loi de groupe abélien $\oplus$ sur $E(K)$ de neutre le point à l'infini O. On a pour tout $P = (x, y)$ dans $E(K)$ donnée sous forme réduite :

(1) $O \oplus (x, y) = (x, y) \oplus O = (x, y)$

(2) $\ominus(x, y) = (x, -y)$

(3) Soit $P, Q, R \in E(K)$, si ces trois points sont alignés alors $P \oplus Q \oplus R = 0$

Maintenant qu'on a donné une structure de groupe abélien à $E(K)$ on peut donner des formules explicites d'addition de points sur la courbe.

Proposition 1. *Soit* $P = (x_1, y_1), Q = (x_2, y_2)$.

$$P \oplus Q = (x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3 - y_1))$$

En posant :

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & si \quad P \neq Q, -Q \\ \frac{3x^2 + a}{2y_1} & si \quad P = Q, y_1 \neq 0 \end{cases} \tag{1}$$

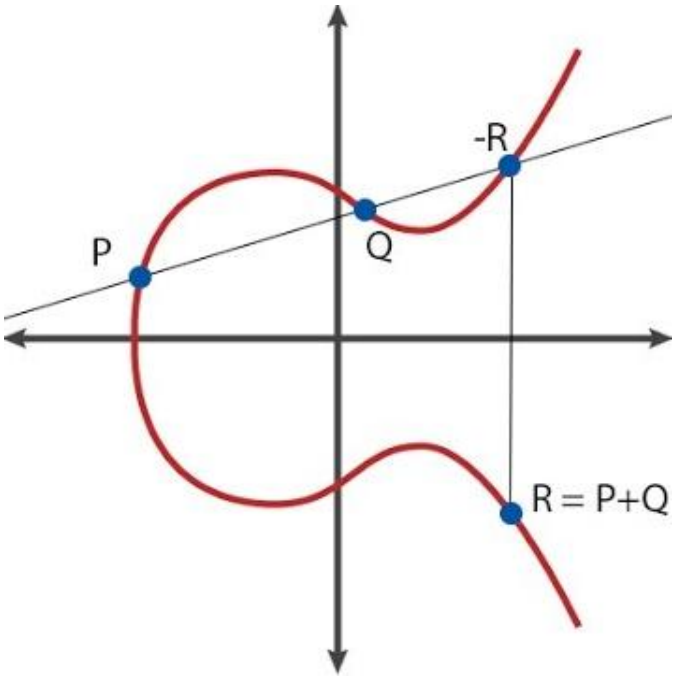


FIGURE 1 – vision graphique de l'addition de deux points sur la courbe

On dispose de formules d'addition pour deux points sur une courbe elliptique on peut donner un sens à la multiplication scalaire d'un point comme $lP = \underbrace{P + \ldots + P}_{l \text{ fois}}$. On va noter cette multiplication scalaire par :

$$\begin{array}{rccc} [l]_E & : & E(\mathbb{K}) & \to & E(\mathbb{K}) \\ & & P & \mapsto & lP \end{array}$$

Ce qui nous permet de définir l'ensemble des points de l-torsion comme le noyau de $[l]$. On note $E[l]$ cet ensemble.

$$E[l] = \left\{ P \in E(\overline{\mathbb{K}}) \mid [l]P = 0_E \right\}$$

3 Compter les points sur une courbe

On va considérer dans la suite que la caractéristique du corps utilisé pour définir nos courbes elliptiques est plus grande que 3. On peut donc écrire notre courbe elliptique sur $\mathbb{F}_p$ sous sa forme réduite $y^2 = x^3 + ax + b$

3.1 Algorithme naïf

On note $E : y^2 = f(x)$, compter les points de E revient donc pour chaque valeur de $x \in \mathbb{F}_p$ à regarder si $f(x)$ est un carré modulo p. On calcule donc le symbole de Legendre de $f(x)$, on a les cas suivants :

- $\left(\frac{f(x)}{p}\right) = -1$, $f(x)$ n'est pas un carré modulo p, on ne trouve aucun point appartenant à la courbe.
- $\left(\frac{f(x)}{p}\right) = 0$, $f(x)$ est divisible par p, on trouve 1 point sur la courbe.
- $\left(\frac{f(x)}{p}\right) = 1$, $f(x)$ est un carré modulo p, on trouve 2 points sur la courbe.

Au final en considérant le point à l'infini on peut calculer le nombre de points de E :

$$\#E(\mathbb{F}_p) = 1 + \sum_{x \in \mathbb{F}_p} \left(\left(\frac{f(x)}{p}\right) + 1\right)$$

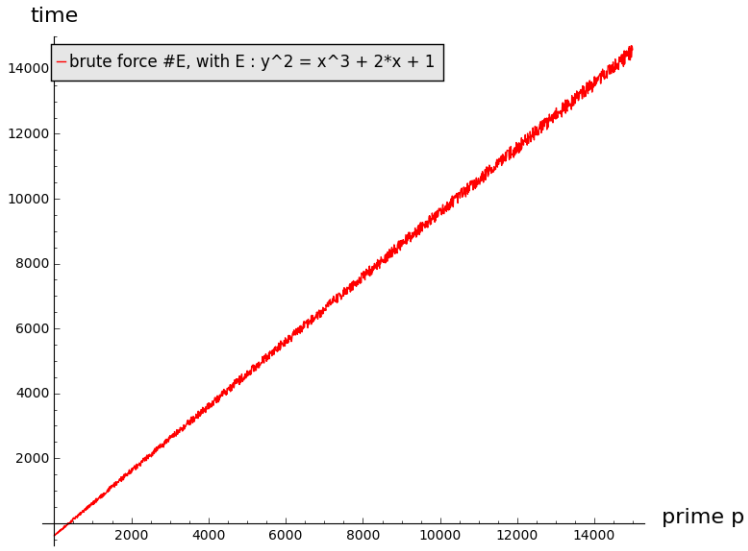
Soit :

$$\#E(\mathbb{F}_p) = 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{f(x)}{p}\right) \tag{2}$$

On peut implémenter cette fonction brute force très simplement sous SAGE en utilisant `kronecker(a,p)` qui calcule le symbole de Legendre de a modulo p.

```
def brute_force(p, a,b):
    R.<x> = PolynomialRing(GF(p))
    f = x**3 + a*x + b
    return 1+p+sum([kronecker(f(i),p) for i in range(p)])
```

La complexité est en la taille de p. Cette méthode est pratique quand p est petit mais devient impraticable s'il est trop grand. On va illustrer la non efficacité de cet algorithme lorsque p devient trop grand par le graphique suivant :



On retrouve bien l'aspect linéaire en p évoqué précédemment.

3.2 Shanks

Il s'agit d'un algorithme Baby steps-giant steps de complexité exponentielle qu'on ne traitera pas ici, mais plus d'informations peuvent être trouvées ici [5].

3.3 Schoof

Soit E une courbe elliptique définie sur $\mathbb{F}_p$ avec p premier > 3 sous sa forme réduite

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

On rappelle le théorème de Hasse-Weil :

Théorème 1. *$\#E(\mathbb{F}_p) = p + 1 - t$ avec $|t| \leqslant 2\sqrt{p}$ trace de l'endomorphisme de Frobenius de E.*

Pour trouver le nombre de points de E il faut donc déterminer t. L'idée de Schoof est de calculer t modulo de petits nombres premiers puis d'utiliser le théorème des restes chinois.

Avant de développer l'algorithme il est nécessaire de donner d'autres définitions.

Définition 4 (Frobenius)**.** *Soit E une courbe elliptique défini sur $\mathbb{F}_p$, l'endomorphisme de Frobenius est défini par*

$$\begin{array}{rccc} \phi_p & : & E(\mathbb{F}_p) & \to & E(\mathbb{F}_p) \\ & & (x, y) & \mapsto & (x^p, y^p) \end{array}$$

On peut définir le polynôme caractéristique de cet endomorphisme par $\phi_p^2 - t\phi_p + p = 0$. On peut déjà faire une première remarque, connaissant t la trace de l'endomorphisme de Frobenius, E une courbe elliptique sur $\mathbb{F}_p$, il est facile de déterminer le cardinal de $E(\mathbb{F}_{p^n})$ pour tout $n > 1$. En effet on a :

$$\#E(\mathbb{F}_{p^n}) = p^n + 1 - (r_1^n + r_2^n)$$

où r_1, r_2 sont les racines de $x^2 - tx + p$ dans $E(\mathbb{F}_{p^n})$, le polynôme caractéristique du Frobenius.

Démonstration. Les racines de $x^2 - tx + p$ sont r_1 et r_2, on a donc $\mathrm{Tr}(\phi_p) = t = r_1 + r_2$.
On remarque de plus que l'on a $\phi_{p^n} = (\phi_p)^n$. Car $\phi_{p^n}(x, y) = (x^{p^n}, y^{p^n}) = \phi_p(x^{p^{n-1}}, y^{p^{n-1}})$, et par récurrence on trouve le résultat.
Ainsi :

$$\begin{aligned}\mathrm{Tr}(\phi_{p^n}(x, y)) &= \mathrm{Tr}(\phi_p(x, y)^n) \\ &= r_1^n + r_2^n\end{aligned}$$

On peut maintenant appliquer le théorème de Hasse-Weil pour trouver le résultat attendu. □

L'équation du polynôme caractéristique du Frobenius reste en particulier vraie sur les points de l-torsion, ce qui nous sera utile par la suite. Ainsi nous avons :

$$\phi_p^2(P) + [p_l]P = [t_l]\phi_p(P) \quad \forall P \in E[l] \tag{3}$$

Avec $t_l \equiv t \pmod{l}$, $p_l \equiv p \pmod{l}$ et $0 \leqslant t_l, p_l < l$.
Il nous faut maintenant introduire les polynômes de division d'une courbe elliptiques E.

Définition 5. *Soit une courbe elliptique $E : y^2 = x^3 + ax + b$ défini sur $\mathbb{K}$. On appelle $f_n(X)$ le n-ième polynôme de divisons défini sur $\mathbb{Z}[x]$ de manière récursive par :*

$$\begin{aligned}f_0(x) &= 0 \\ f_1(x) &= 1 \\ f_2(x) &= 1 \\ f_3(x) &= 3x^4 + 6ax^2 + 12bx - a^2 \\ f_4(x) &= 2x^6 + 10ax^4 + 40bx^3 - 10a^2x^2 - 8abx - 2(a^3 + 8b^2)\end{aligned}$$

On pose $F(X) = 4x^3 + 4ax + 4b$, et on a :

$$\left\{\begin{array}{lll} f_{2n} & = f_n(f_{n+2}f_{n-1}^2 - f_{n-2}f_{n+1}^2) & \\ f_{2n+1} & = \left\{\begin{array}{lll} F^2 f_{n+2}f_n^3 - f_{n-1}f_{n+1}^3 & si & n \text{ est pair} \\ f_{n+2}f_n^3 - f_{n-1}f_{n+1}^3 F^2 & si & n \text{ est impair} \end{array}\right. \end{array}\right. \tag{4}$$

Ces polynômes sont de degré au plus $\frac{(n^2-1)}{2}$ si n est pair, ou bien au plus $\frac{(n^2-2)}{2}$ si n est impair.

On peut utiliser les polynômes de division pour calculer la multiplication scalaire d'un point de la courbe E. On a les formules suivantes :

Théorème 2. *Soit E une courbe elliptique défini sur $\mathbb{K}$, un point P sur cette courbe et $m \in \mathbb{N}^*$.*

$$[m]P = \left\{\begin{array}{ccc} O_E & si & P \in E[m] \\ \left(\frac{\phi_m(x,y)}{\psi_m^2(x,y)}, \frac{\omega_m(x,y)}{\psi_m^3(x,y)}\right) & sinon & \end{array}\right. \tag{5}$$

En posant :

$$\psi_m = \left\{\begin{array}{cc} 2yf_m & si\ m \text{ est pair} \\ f_m & sinon \end{array}\right.$$

et

$$\begin{cases} \phi_m & = x\psi_m^2 - \psi_{m-1}\psi_{m+1} \\ \psi_m\omega_m & = \psi_{2m} \end{cases}$$

On peut aussi réécrire $[m]P$ *sous cette forme :*

$$[m]P = \begin{cases} O_E & si \quad P \in E[m] \\ \left(x - \frac{\psi_{m-1}(x,y)\psi_{m+1}(x,y)}{\psi_m^2(x,y)}, \frac{\psi_{2m}(x,y)}{\psi_m^4(x,y)}\right) & sinon \end{cases} \tag{6}$$

Démonstration. On veut démontrer l'équation 6.
On note $[m]P = (x_1, y_1)$ On a alors :

$$y_1 = \frac{\omega_m}{\psi_m^3} = \frac{\psi_{2m}}{\psi_m}\frac{1}{\psi_m^3} = \frac{\psi_{2m}}{\psi_m^4}$$
$$x_1 = \frac{\phi_m}{\psi_m^2} = \frac{x\psi_m^2 - \psi_{m-1}\psi_{m+1}}{\psi_m^2} = x - \frac{\psi_{m-1}\psi_{m+1}}{\psi_m^2}$$

□

On remarque que $[m]P = (x_1, y_1)$ avec x_1 uniquement fonction de x. En effet, en utilisant les formules de récurrence, si m est pair $\psi_m^2 = 4y^2 f_m$, mais en utilisant l'équation de E on peut remplacer y^2 par une fonction de x, et les polynômes de division sont en fonction de x. De plus comme m est pair les polynômes $\psi_{m+1} et \psi_{m-1}$ sont uniquement en fonction de x. Dans le cas où m est impair on a l'inverse, le numérateur fait apparaître un y^2 qu'on peut remplacer par une fonction de x comme précédemment et le dénominateur est une fonction de x. Quand à y_1, son degré en y est au plus 1. Si m est pair ψ_m^4 fait apparaître un y^4 qui s'exprime en fonction de x, et ψ_{2m} amène un y. On montre la même chose quelque soit la parité de m et $2m$. On peut ainsi exprimer $[m]P$ comme un polynôme en x, y. Mais ce n'est pas la seule particularité de ces polynômes utiles pour notre algorithme. En effet $P = (x_1, y_1)$ est un point de l-torsion si et seulement si x_1 est une racine du l-ième polynôme de division f_l. De plus P est sur la courbe E. Les points de l-torsion sont donc solutions du système d'équation :

$$E(x, y) = y^2 - x^3 - ax - b = 0, \quad f_l(x) = 0 \tag{7}$$

L'équation 3 peut donc se récrire en utilisant les points de l-torsion. On va maintenant faire des calculs dans l'anneau

$$\mathbb{W} = \frac{\mathbb{F}_p[x, y]}{(f_l(x), E(x, y))}$$

L'idée de l'algorithme de Schoof est donc de tester pour des valeurs $\tau_l \in \{0, \ldots, l-1\}$ si l'équation suivante est vraie dans cet anneau :

$$(x^{p^2}, y^{p^2}) + [p_l](x, y) = [\tau_l](x^p, y^p), \quad P = (x, y) \tag{8}$$

L'unique solution que l'on trouve est t_l.On répète l'opération pour d'autres l premiers jusqu'à avoir assez de t_l pour appliquer le théorème des restes chinois et retrouver la valeur de t. L'intérêt de travailler avec des points de l-torsion est ainsi de pouvoir borner la taille des polynômes en jeu dans l'équation du Frobenius, en $O(l^2)$, ce qui est plus efficace que de uniquement travailler avec des fonctions rationnelles.
On va maintenant détailler l'algorithme étape par étape.

3.3.1 Choix de l'ensemble de premiers

L'idée de Schoof est de calculer la trace de l'endomorphisme de Frobenius modulo de petits premiers. Il faut choisir un ensemble $S = (l_1, \dots l_n)$ tel que $\prod l_i > 4\sqrt{p}$. En effet on rappelle que $|t| \leqslant 2\sqrt{p}$, on veut s'assurer de bien pouvoir reconstruire notre solution quand on va utiliser les restes chinois. Pour construire S on va juste ajouter des nombres premiers à une liste en utilisant la fonction `next_prime` de SAGE tant que le produit des éléments de S est plus petit que $4\sqrt{p}$. Au final cela revient à la boucle suivante ou N et `prime` sont initialisés à 1 et `list_l` correspond à S :

```
while N <= 4*sqrt(p):
        prime = next_prime(prime)
        if prime == p:
                continue
        N = N*prime
        list_l.append(prime)
```

3.3.2 Cas $l = 2$

Ce cas est particulier et doit être traité à part car on suppose nos premiers impairs. On a $t_2 \equiv 0 \pmod 2$ si et seulement si $E(\mathbb{F}_p)$ a un élément d'ordre 2, c'est à dire un point de 2-torsion. Or les points de 2-torsion sont de la forme $(x, 0)$. En effet, si $2P = 0$, pour $P \in E(\mathbb{F}_l)$ alors $P = -P$, et cela correspond aux points de l'axe des abscisses. Ainsi t_2 est congru à 0 si et seulement si $x^3 + ax + b$ à une racine dans $\mathbb{F}_p$. Mais les racines de $\mathbb{F}_p$ sont entièrement déterminées par le polynôme $x^p - x$, nos deux polynômes partageraient alors une racine commune. La façon utilisée ici pour décider si $t_2 \equiv 0 \pmod 2$ est donc de calculer si $\gcd(x^p - x, x^3 + ax + b) \neq 1$.

3.3.3 Calcul des polynômes de division

Il existe dans SAGE une fonction permettant de calculer les polynômes de division, `polynomial_polynomial(n)` renvoyant le n-ième polynôme de division. Mais comme au cours de l'algorithme on doit souvent utiliser différents polynômes de division et de plus ils sont calculés via des formules de récurrence. Il ne me paraissait pas forcement très judicieux d'utiliser la fonction de SAGE qui allait de toute façon très certainement à chaque appel devoir recalculer via les formules de récurrence le polynôme souhaité. L'idée étant de stocker tous ces polynômes dans un dictionnaire, une fois la phase de pré calcul faite on peut donc accéder à tous les polynômes de division que l'on souhaite en temps constant au cours de l'exécution de l'algorithme de Schoof. Et en utilisant la fonction de SAGE j'ai tout de même vérifié via une fonction de test que je trouvais bien les même polynômes qu'avec ma fonction.

3.3.4 Calcul de $(x^{p^2}, y^{p^2}) + [p_l](x, y)$

Comme vu en introduction aux courbes elliptiques la somme de deux points appartenant à la courbe dépend de plusieurs cas. Est-ce que nos points sont distincts ou ont la même coordonnée en x ? En effet les formules à appliquer seront différentes selon que l'on est dans un cas ou l'autre. Il est plus probable d'avoir des coordonnées différentes, on va donc travailler dans ce cas là. Si l'algorithme trouve un t_l vérifiant l'équation on est dans le bon cas, s'il n'en trouve aucun, cela veut dire que l'on se trouve dans l'autre cas.

Cas : $(x^{p^2}, y^{p^2}) \neq \pm[p_l](x,y)$ On peut utiliser notre formule usuelle d'addition sur courbe elliptique pour calculer $(x_1, y_1) = (x^{p^2}, y^{p^2}) + (x_{\bar{p_l}}, Y_{\bar{p_l}})$ en notant $(x_{\bar{p_l}}, y_{\bar{p_l}}) = [p_l](x,y)$.
Soit $\lambda = \frac{y_{\bar{p_l}} - y^{p^2}}{x_{\bar{p_l}} - x^{p^2}}$, on a alors :

$$(x_1, y_1) = (\lambda^2 - x^{p^2} - x_{\bar{p_l}},\ \lambda(x^{p^2} - x_1) - y^{p^2}) \tag{9}$$

On va aussi noter $(x_\tau, y_\tau) = [\tau](x,y)$. On doit maintenant tester en faisant varier la valeur de τ entre 0 et $\frac{l-1}{2}$ si $x_1 = x_\tau$. Si c'est le cas on se retrouve avec deux possibilités comme valeur de y_1, soit on a le point soit on a son opposé. On regarde donc si $y_1 = y_\tau$, on a alors $t_l = \tau$, sinon on a $t_l = -\tau$

Cas : $(x^{p^2}, y^{p^2}) = \pm[p_l](x,y)$ Là encore deux cas possibles. On va d'abord regarder lorsque $(x^{p^2}, y^{p^2}) = [p_l](x,y)$.
Soit $P = (x,y)$, on rappelle l'équation caractéristique de l'endomorphisme Frobenius restreinte aux points de l-torsion :

$$\phi_p^2(P) + [p_l]P = [t_l]\phi_p(P) \qquad (\text{mod } l) \tag{10}$$

On a alors dans notre cas $2[p_l]P = [t_l]\phi_p(P) \pmod{l}$. De plus on a $\phi_p^2 = [p_l]P$. En combinant ces deux égalités on trouve :

$$[t_l^2 p_l]P = [t_l^2]\phi_p^2 = [t_l]\phi([t_l]\phi_p) = [(2p_l)^2]P \qquad (\text{mod } l) \tag{11}$$

Ainsi p_l est un carré modulo l. On pose $p_l = \omega^2 \pmod{l}$. On va maintenant calculer $\omega\phi P$, si $[p_l]P = \omega\phi P$ alors $t_l = 2\omega$ sinon $t_l = -2\omega$.
Si p_l n'est pas un carré modulo l on est dans le cas $(x^{p^2}, y^{p^2}) = -[p_l](x,y)$ et on a alors $t_l = 0$ d'après l'équation caractéristique.

3.3.5 Algorithme

On donne une version complète en pseudo-code de l'algorithme de schoof :

Algorithm 1 Schoof

Require: E une courbe elliptique de la forme $y^2 = x^3 + ax = b$ sur $\mathbb{F}_p$
Ensure: Le nombre de points de E

Choisir un ensemble de premier S tel que $\prod_{l \in S} l \leqslant 4\sqrt{p}$
if $\gcd(x^q - x, x^3 + ax + b) \neq 1$ **then**
  $t_2 = 0$
else
  $t_2 = 1$
end if
for all $l \in S$ **do**
  Calculer le polynôme de division f_l, on fera les calculs dans l'anneau $\mathbb{F} = \frac{\mathbb{F}_l[X,Y]}{(f_l(X), y^2 - x^3 - ax - b)}$)
  On pose $p_l = p \mod (l)$
  Calculer (x^p, y^p), (x^{p^2}, y^{p^2}), $[p_l](x, y) = (x_{p_l}, y_{p_l})$
  if $x^{p^2} \neq x_{p_l}$ **then**
    Calculer $(X, Y) = (x^{p^2}, y^{p^2}) + (x_{p_l}, y_{p_l})$
    for all $1 \leqslant \tau \leqslant l - 1$ **do**
      if $X = x_\tau^p$ **then**
        if $Y = y_{p_l}^q$ **then**
          $t_l = \tau$
        else
          $t_l = -\tau$
        end if
      end if
    end for
  else
    if p est un carré modulo l **then**
      Calculer w tel que $p \equiv w^2 \pmod{l}$
      Calculer $[w](x^p, y^p)]$
      if $[w](x^p, y^p)] = (x^{p^2}, y^{p^2})$ **then**
        $t_l = 2w$
      end if
      if $[w](x^p, y^p)] = (x^{p^2}, -y^{p^2})$ **then**
        $t_l = -2w$
      else
        $t_l = 0$
      end if
    else
      $t_l = 0$
    end if
  end if
end for

Implémentation Il a fallu commencer par implémenter des fonctions de base sur les courbes elliptiques pour additionner, calculer une multiplication scalaire. Les algorithmes d'addition ne nécessitent pas d'explications, juste une application des formules. Pour la multiplication scalaire on avait deux choix, utiliser les polynômes de division ou de l'exponentiation rapide. La méthode utilisant les polynômes a amené des problème d'implémentation, notamment pour faire les calculs dans les bons anneaux, donc je suis resté avec une version plus simple d'exponentiation. L'idée de cet algorithme est d'utiliser la décomposition binaire de n pour calculer nP. Pour récupérer cette décomposition on utilise la fonction `digits(base)` qui retourne une liste avec les bits de poids faible à gauche de l'écriture binaire de n. On va à chaque itération doubler le point que l'on a, on calcule donc $2(2(\ldots 2(P))$. Il est quand même nécessaire lorsque l'on rencontre un 1 dans la décomposition binaire de n de faire une addition en plus de doubler. Au final on obtient l'algorithme suivant avec comme entrée n et P :

```
nbits = n.digits(2)
i = len(nbits)-2
Q = P
while i >= 0:
    Q = double(Q,a)
    if nbits[i]:
      Q = add(P,Q,a)
    i -= 1
return Q
```

Sans donner ici le détail complet de l'algorithme de Schoof sous SAGE il est intéressant de détailler certains points de l'implémentation. Le premier soucis rencontré fut la construction de l'anneau dans lequel on doit travailler. On crée tout simplement un anneau de polynôme dans le corps K pour commencer :

```
R.<x> = PolynomialRing(K)
```

En notant `poly_divi_l` le l-ième polynome de division on peut construire l'anneau quotient :

```
B.<x2> = R.quotient(ideal(poly_divi_l))
```

Cependant on n'est pas certain que ce polynôme soit irréductible, et ainsi créer un corps fini. Si ce n'est pas le cas et que l'on doit calculer l'inverse d'un élément non inversible, on rencontrera une erreur. Il est donc nécessaire de tester si le polynôme de division est irréductible et s'il ne l'est pas de le remplacer par un de ses diviseurs. Ce qu'on illustre avec la condition suivante :

```
if not poly_divi_l.is_irreducible():
        for poly, deg in list(factor(poly_divi_l)):
                if deg > 1:
                poly_divi_l  = poly
                break
```

Une fois que cela est fait on peut, en notant $f = x^3 + ax + b$, créer l'anneau final :

```
B.<x2> = R.quotient(ideal(poly_divi_l))
C.<y> = PolynomialRing(B)
W = C.quotient(y**2 - f)
```

Les calculs pour obtenir les différents t_l modulo l sont juste une succession de test comme décrit dans l'algorithme en pseudo-code. Il faut juste faire attention à bien faire les calculs dans le bon anneau. Une fois que l'on a calculé une liste `list_t` contenant tous les t modulo l calculés par l'algorithme on peut utiliser la fonction de SAGE `crt(list_t,list_l)` renvoyant la valeur de t final en utilisant les restes chinois.

3.3.6 Exemple

On va travailler avec la courbe E suivante, $E : y^2 = x^3 + 2x + 1$ sur $\mathbb{F}_{19}$.

l = 2 On calcule $\gcd(x^{19} - x, x^3 + 2x + 1)$ et on trouve 1, donc $t \equiv 1 \pmod 2$

l=3 On a $p^2 = 361$ et $p_l \equiv 1 \pmod 3$, on pose ainsi $p_l = 1$. L'équation du Frobenius devient :

$$(x^{361}, y^{361}) + (x, y) = [\tau_l](x^{19}, y^{19}) \quad (x, y) \in E[3],\ \tau_l = \pm 1 \tag{12}$$

Le troisième polynôme de division est $f_l = 3x^4 + 12x^2 + 12x - 4$ mais ce polynôme n'est pas irréductible dans $\mathbb{F}_{19}$, on va plutôt travailler avec un de ses diviseurs : $x^3 + 8x^2 + 16$.
Il nous faut maintenant calculer la partie gauche de cette équation. On a $x^{361} = 3x^2 + 2x + 5 \pmod{f)_l}$ et $y^{361} = y(y^2)^{180} = y(4x^2 + 9x + 16)$. On est dans le cas ou $x^{361} \neq x$. On doit maintenant utiliser nos formules de sommations pour calculer $(x_3, y_3) = (x^{361}, y^{361}) + (x, y)$:

$$\begin{aligned} x_3 = (\frac{y(4x^2 + 9x + 16) - y}{3x^2 + 2x + 5 - x})^2 - 3x^2 - 2x - 5 - x \quad &= (y(4x^2 + 9x + 15)(x^2 + 15x + 6))^2 - 3x^2 - 3x - 5 \\ = 11 - 3x^2 - 3x - 5 \\ = 16x^2 + 16x + 6 \end{aligned}$$

et,

$$\begin{aligned} y_3 &= \frac{y(4x^2 + 9x + 16) - y}{3x^2 + 2x + 5 - x}(4x^2 + 9x + 16 - 16x^2 - 16x - 6) - y(4x^2 + 9x + 16) \\ &= y(4x^2 + 9x + 15)(x^2 + 15x + 6)(-12x^2 - 7x + 10) - y(4x^2 + 9x + 16) \\ &= y(13x^2 + 13x + 14) \end{aligned}$$

Maintenant nous devons calculer pour τ variant de 0 à 2 si $(x_3, y_3) = \tau(x^p, y^p) = (x_\tau^p, y_\tau^p)$. Pour $\tau = 1$ on trouve $x_3 = x_\tau^p$ et $y^p = -y_\tau^p$. Ainsi $t_3 = -\tau = -1 = 2 \pmod 3$.

3.3.7 Expérimentations

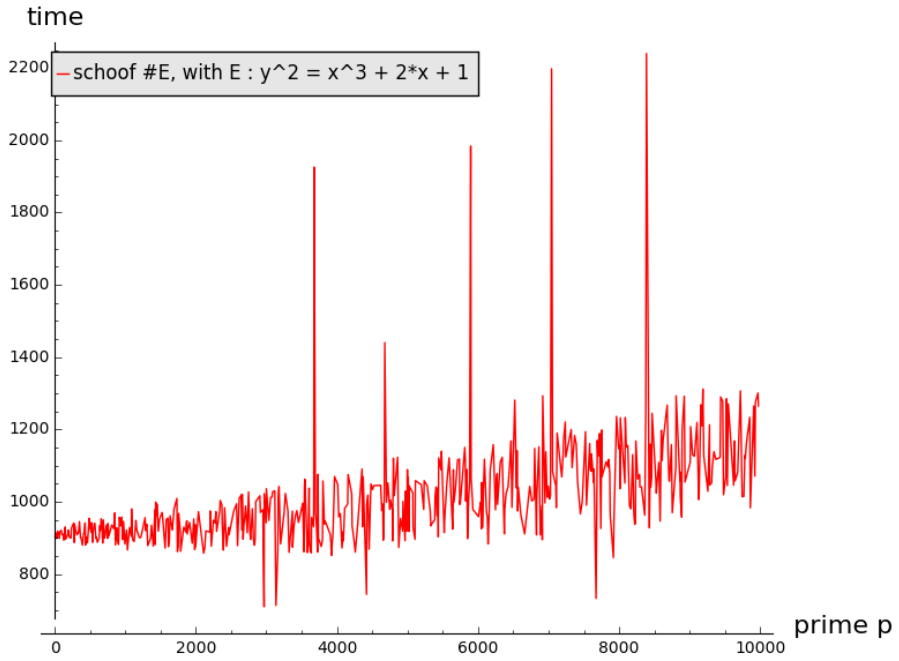
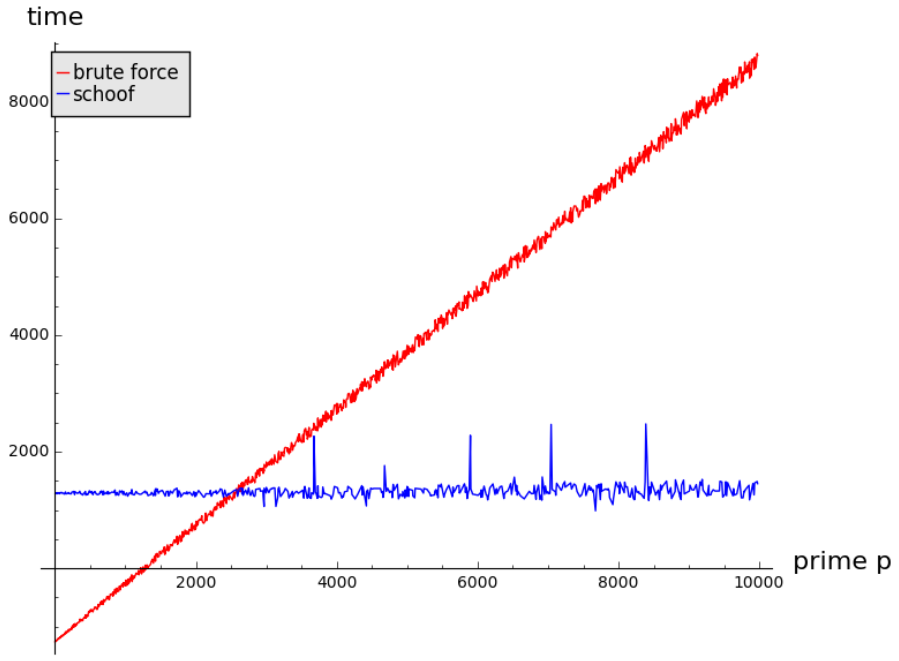


FIGURE 2 – Temps d'exécution de l'algorithme de Schoof

FIGURE 3 – Comparaison temps exécution entre les algorithmes de Schoof et de brute force



On peut tirer deux informations intéressantes de ce graphique. D'une part on a confirmation que l'algorithme de Schoof est bien plus efficace que la méthode naïve dès que p devient grand, mais on voit graphiquement que pour $p < 2500$ il est plus efficace d'utiliser l'algorithme naïf.

Il est intéressant de se demander quel est l'espace mémoire utilisé lors de l'exécution de l'algorithme de schoof, et vers quel quel p premier on commence à atteindre des limites. Pour cela on va utiliser la plateforme Plafrim pour faire nos calcul sur une machine Intel(R) Xeon(R) Gold 6142 CPU (Skylake), un processeur à 16 cœur, avec un fréquence de $2,60$ GHz.
On a testé notre algorithme de Schoof sur une courbe du standard du NIST pour tester les limites de l'algorithme dans un cas réaliste. La courbe choisi est P-192 de paramètres :

$$p = 6277101735386680763835789423207666416083908700390324961279$$
$$a = -3$$
$$b = 1679885593$$

On a du rajouter une option `sparsed=True` lors de la création de nos anneaux de polynômes pour permettre a SAGE de travailler avec des polynômes de très haut degré car sinon il renvoyait une erreur de type overflow. De plus le cas $l = 2$ demande de calculer $\gcd(x^p - x, x^3 - ax - b)$, ce qui sans algorithme efficace demande beaucoup trop de temps. Plutôt que d'essayer d'accélérer ce calcul le choix a été fait de simplement sauter le nombre premier 2. De plus dans notre algorithme on teste si le l-ième polynôme de division est irréductible ou non, s'il ne l'est pas on le remplace par l'un de ses facteurs. On utilise la fonction de SAGE `factor()` qui renvoie la factorisation de ce polynôme. Or si le degré devient important la factorisation de notre polynôme va devenir plus longue que l'exécution complète de l'algorithme. On a uniquement besoin d'un seul facteur, pas de la décomposition entière, on pourrait donc optimiser cette étape.
L'algorithme a tourné sur cette courbe pendant environ 25 heures. L'ensemble de premiers l considéré est :

```
('liste des premier: ', [5, 11, 17, 23, 31, 41, 47, 59, 67,
73, 83, 97, 103, 109, 127, 137, 149, 157])
```

Il faut noter que quand le cas $l = 2$ a voulu être écarté une fausse manipulation lors d'un copier coller a eu lieu. En fait on appelle deux fois la fonction `next_prime()`, et donc comme on le voit dans la liste, un nombre premier sur deux est sauté, ce qui fait qu'on se retrouve avec un l maximum de 157 au lieu de 79 en théorie. La phase de pré-calcul a du calculer des polynômes de division jusqu'au 157 ième. Au bout de 25 heures l'algorithme a calculé les traces modulo l jusqu'à $l = 73$. En terme de mémoire, cela a pour le moment demandé 24 Go. Il faut nuancer ces résultats par le fait que l'algorithme fait des calculs inutiles comme évoqué précédemment et travaille sur de mauvais paramètres.
On a pu relancer les calculs pendant 9 heures sur la même machine, mais cette fois ci avec un code correct. On a l'ensemble de premiers suivants :

```
('liste des premiers: ', [3, 5, 7, 11, 13, 17, 19,
 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79])
```

Sur ce temps de calcul on a pu calculer les traces modulo ces nombres premiers jusqu'à 73. L'algorithme a utilisé environ 4 Go.

3.3.8 Complexité

La complexité de l'algorithme de Schoof est en $O(\log q^8)$ opérations élémentaires. La partie coûteuse de l'algorithme réside dans les calculs dans l'anneau $R = \frac{\mathbb{F}_p[x,y]}{(f_l(x), y^2 - x^3 - ax - b)}$. Les éléments de R sont de degré $O(l^2)$

L'algorithme est plus efficace qu'une version naïve quand la taille de p augmente mais reste néanmoins limitée pour p trop grand. En effet le degré des polynômes de division en $O(l^2)$ empêche une performance optimale pour des tailles de p trop importante. Les polynômes de division deviennent rapidement très grands, ce qui pose aussi un problème de mémoire. Une amélioration de cet algorithme a été proposée par Elkies-Atkin pour produire un algorithme plus performant, le SEA.

3.4 Algorithme SEA

Il va être nécessaire d'introduire plus de théorie avant de s'attaquer à l'algorithme en lui même. L'idée finale est d'utiliser un polynôme de degré plus petit que les polynômes de division utilisés dans l'algorithme de Schoof, les polynômes modulaires. De plus pour comprendre certaines parties de l'algorithme SEA il va être nécessaire d'introduire un peu de théorie d'analyse complexe.

3.4.1 Analyse complexe

La théorie des courbes elliptiques est conséquente sur le corps des complexes mais nous allons uniquement donner des définitions et formules utiles, qu'on pourra retrouver ici [3] [1], sans rentrer dans le détail de preuves et résultats non pertinents pour la compréhension de l'algorithme SEA. On peut légitimement se demander le rapport entre courbe elliptique et analyse complexe. Nous allons donc donner quelques définitions avant d'expliciter ce lien.

Définition 6. *Une fonction holomorphe est une fonction à valeur complexe, définie et dérivable en tout point d'un sous ensemble ouvert du plan complexe $\mathbb{C}$. Une fonction est dite méromorphe si elle est holomorphe dans tout le plan complexe $\mathbb{C}$, sauf éventuellement sur un ensemble de points isolés dont chacun est un pôle pour la fonction.*

Proposition 2. *Soit $n \in \mathbb{N}$, $r \in \mathbb{R}$, $(w_i)_{1 \leqslant i \leqslant r}$ une partie libre du $\mathbb{R}$-espace vectoriel $\mathbb{R}^n$. Tout sous groupe discret non nul Γ de $\mathbb{R}^n$ peut s'écrire sous la forme :*

$$\Gamma = \mathbb{Z}\omega_1 + \ldots + \mathbb{Z}\omega_r$$

Un tel groupe Γ est un réseau de $\mathbb{R}^n$ de rang r.

Dans le cas particulier de $\mathbb{C}$, ses sous-groupes discrets non nuls et non isomorphes à $\mathbb{Z}$ sont de la forme $\mathbb{Z}\omega_1 + \mathbb{Z}w_2$ avec $\omega_1, \omega_2 \in \mathbb{C}$ et $Im(\frac{w_2}{w_1}) \neq 0$. En pratique on va souvent considérer $\tau = \frac{\omega_2}{\omega_1}$ pour avoir $\Gamma = \mathbb{Z} + \tau\mathbb{Z}$.

Définition 7. *On appelle tore le quotient $T = \mathbb{C}/\Gamma$. C'est le quotient du groupe $(\mathbb{C}, +)$ par le réseau $\Gamma = \mathbb{Z}\omega_1 + \mathbb{Z}w_2$*
Une fonction elliptique de période Γ est une fonction méromorphe f sur $\mathbb{C}$ telle que $f(z+\omega) = f(z),\ \forall \omega \in \Gamma$.

Définition 8. *Soit Γ un réseau de $\mathbb{C}$, on introduit la $\wp$-fonction de Weierstrass par la série suivante :*

$$\wp(z) = \frac{1}{z^2} + \sum_{w \in \Gamma, \omega \neq 0} \frac{1}{(z-\omega)^2} - \frac{1}{\omega^2}$$

On a une convergence absolue donc uniforme sur tout compact disjoint de Γ *de cette série. De plus cette fonction est elliptique. On peut aussi définir sa dérivée* $\wp'$ *qui de même est elliptique :*

$$\wp' = -2 \sum_{\omega \in \Gamma} \frac{1}{(z-\omega)^3}$$

On peut montrer qu'avec $\Gamma = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ *cette fonction* ρ *est doublement périodique de période* ω_1, ω_2.

Les théorèmes suivants permettent de faire le lien entre ces fonctions d'analyse complexe et les courbes elliptiques.

Théorème 3. *Soit* $E/\mathbb{C}$ *une courbe elliptique sous forme réduite sur le corps des complexes. Il existe alors un réseau* Γ *tel que l'application suivante soit une bijection :*

$$\begin{array}{ccc} \mathbb{C}/\Gamma & \to & E \\ z+\Gamma & \mapsto & \begin{cases} (\wp(z), \frac{\wp'(z)}{2}) & z \notin \Gamma \\ O & z \in \Gamma \end{cases} \end{array}$$

Théorème 4. *La fonction* $\wp$ *satisfait l'équation différentielle suivante :*

$$\wp'(z)^2 = 4\wp^3(z) + A\wp + B$$

où les coefficients A, B *sont totalement déterminés en analysant les développements limités en* 0 *de* $\wp, \wp'$ *et* $\wp'^2$ *et dépendent du réseau* Γ.

3.4.2 Isogénie

Soit K un corps fini, on va noter dans la suite E/K la courbe elliptique définie sur ce corps et on travaillera uniquement avec des courbes sous forme réduite.

Définition 9. *Soit* E_1/K *et* E_2/K *deux courbes elliptiques. Si* E_1 *et* E_2 *ont le même* j*-invariant alors elles sont isomorphiques sur* $\bar{K}$.

En d'autres termes, dire que deux courbes sont isomorphes signifie qu'il existe un changement de variable permettant de passer de l'une à l'autre. Avec un isomorphisme entre nos courbes, on peut s'intéresser à la structure des groupes des points rationnels de nos courbes, et à leurs relations.

Proposition 3. *Soit* E_1/K *et* E_2/K *deux courbes elliptiques isomorphes. On a alors un morphisme de groupe entre* $E_1(K)$ *et* $E_2(K)$.

Définition 10. *Soit deux courbes elliptiques* E_1 *et* E_2, $T_1 = \mathbb{C}/\Gamma_1$ *et* $T_2 = \mathbb{C}/\Gamma_2$ *les deux tores associés. Un morphisme de* E_1 *vers* E_2 *est une application holomorphe* μ *de* T_1 *vers* T_2 *qui soit un morphisme de groupe. Si ce morphisme est non constant alors on dit que c'est une isogénie.*

Deux courbes elliptiques E_1/K et E_2/K sont isogènes s'il existe une isogénie $\psi : E_1 \mapsto E_2$. Le degré de l'isogénie est le degré du noyau de ψ.

3.4.3 Polynôme modulaire

Ce sont les polynômes que l'on va utiliser dans l'algorithme SEA. Comme vu dans les parties précédentes on peut associer à chaque courbe elliptique $E/\mathbb{C}$ un invariant τ (on définissait un réseau $\Gamma = \mathbb{Z} + \tau\mathbb{Z}$), on a aussi $p = \mathrm{e}^{2\pi i\tau}$. De plus on peut exprimer le j-invariant de E comme une série entière $j(p)$. Dans la suite on va noter $j(\tau) = j_E$.

La définition des polynômes modulaires est un peu plus compliquée qu'une simple formule, il nous faut définir, pour tout entier n, l'ensemble suivant :

$$S_n^* = \left\{ \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \mid a, b, d \in \mathbb{Z}, 0 \leqslant b \leqslant d, \gcd(a, b, d) = 1 \right\}$$

On peut définir la quantité suivante pour $\alpha \in S_n^* : j \circ \alpha = j(\frac{a\tau+b}{d})$ On peut maintenant donner une définition des polynômes modulaires.

Définition 11. *Soit n un nombre premier, le n-ième polynôme modulaire est :*

$$\Phi_n(x, j) = \prod_{\alpha \in S_n^*} (x - j \circ \alpha)$$

On peut noter que ce polynôme est à coefficients dans $\mathbb{Z}$, symétrique et de degré $n + 1$ en chaque variable. De plus les coefficient de x^{n+1} et y^{n+1} sont 1. Ce qui nous permet de faire le lien entre isogénie et polynôme modulaire avec la définition suivante.

Définition 12. *Soit $E_1/\mathbb{C}$ et $E_2/\mathbb{C}$ deux courbes elliptiques de j-invariant respectivement j_{E_1} et j_{E_2}. Le l-ième polynôme modulaire vérifie $\Phi_l(j_{E_1}, j_{E_2}) = 0$ si et seulement si il existe une isogénie entre E_1 et E_2 dont le noyau est cyclique de degré l.*

Ce théorème nous donne une méthode pour trouver toutes les isogénies à une courbe elliptique $E/\mathbb{F}_p$. Soit l un nombre premier différent de p, on va chercher les racines de $\Phi_l(j_E, y) \in \mathbb{F}_p[y]$ dans $\mathbb{F}_p$. Quand on trouve une racine, on trouve un j-invariant d'une courbe E_2 isogène avec E.

On donne un exemple de polynôme modulaire, pour $l = 3$ on a :

$$\begin{aligned}
\Phi_3(x, y) = {} & x^4 - x^3y^3 + y^4 \\
& +2232(x^3y^2 + x^2y^3) \\
& -1069956(x^3y^2 + x^2y^3 \\
& +36864000(x^3 + y^3) \\
& +2587918086x^2y^2 \\
& +8900222976000(x^2y + xy^2) \\
& +452984832000000(x^2 + y^2) \\
& -770845966336000000xy \\
& +1855000000000(x + y)
\end{aligned}$$

On peut noter que les coefficients augmentent très rapidement. Un algorithme efficace pour calculer ces polynômes dans le cas où l est premier est le calcul via volcan d'isogénie [4]

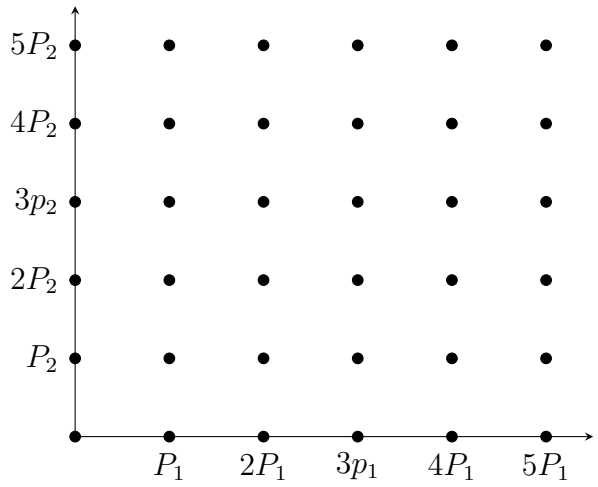
3.4.4 Algorithme SEA

Avec ce bagage théorique en plus, on va pouvoir commencer à détailler l'algorithme SEA (schoof-Elkies-Atkins), on suppose dans la suite que E est une courbe elliptique sous sa forme réduite $y^2 = x^3 + ax + b$ sur $\mathbb{F}_p$ avec p premier . On rappelle que le polynôme caractéristique de l'endomorphisme de Frobenius ϕ sur $\mathbb{F}_l$ est $\chi_l(x) = x^2 - t_l x + p_l$ où t_l est la trace du Frobenius modulo l, et p_l la caractéristique du corps sur lequel on définit E réduite modulo l. On va se demander si χ_l est irréductible sur $\mathbb{F}_l$. Or ce a une racine dans $\mathbb{F}_l$ si et seulement si son déterminant $\Delta_\chi = t_l^2 - 4p_l$ est un carré dans $\mathbb{F}_l$, ce qui nous amène à la classification suivante :

Définition 13. *Si $\Delta_\chi = t_l^2 - 4p_l$ est un carré non nul dans $\mathbb{F}_l$ alors on dit que l est un premier de Elkies, sinon c'est un premier de Atkins.*

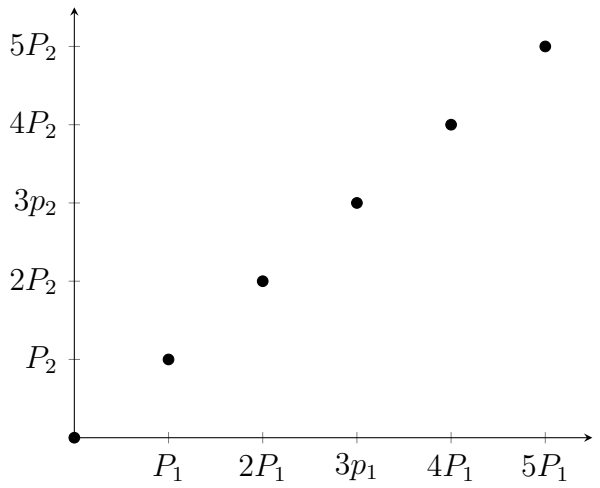
On peut représenter graphiquement l'ensemble des points de l-torsions. On sait que l'on a l^2 points de l-torsion et que l'on peut décrire $E[l]$ par son l-ième polynôme de division f_l de degré en $O(l^2)$.

Représentation de $E[6]$



L'idée de l'algorithme SEA est de ne plus travailler sur cet ensemble $E[l]$ mais sur l'un de ses sous groupe contenant l points et décrit par un polynôme g_l divisant f_l mais cette fois de degré en $O(l)$.

Représentation d'un sous groupe de $E[6]$



On peut noter que $E[l]$ contient $l+1$ sous groupe cyclique d'ordre l. La question est donc : quel sous groupe prendre ? On va s'intéresser à un sous espace propre de la restriction de ϕ à $E[l]$ de dimension 1 (comme visible sur le graphique précédent on prend une droite) qu'on note C_λ. La valeur propre associée à ce sous espace propre est bien sur λ tel que $\phi(P) = \lambda P$ pour $P \in C_\lambda$. On décrit C_λ par g_l.
On peut décrire $\phi_{|E[l]}$ par une matrice 2×2, la question de trouver des sous espaces propres est liée à la question de diagonalisation de cette matrice.

Atkins a montré qu'a partir des polynômes modulaires on peut déterminer si Δ_χ est un carré. Il en résulte le théorème suivant :

Théorème 5. *Soit E une courbe elliptique définie sur $\mathbb{F}_p$ de j-invariant non nul. On note $\Phi_l(x, j) = h_1(x) \dots h_s(x)$ la factorisation du l-ième polynôme modulaire sur $\mathbb{F}_p[x]$ en produit de polynôme irréductible. On a alors les possibilités suivantes pour les degrés de $h_1, \dots, h_s$:*

(1) $(1, l)$ ou $(1, \dots, 1)$, dans les deux cas $\Delta_\chi \equiv 0 \pmod{l}$, et on pose respectivement $r = l$ et $r = 1$.

(2) $(1, 1, r, \dots, r)$, dans ce cas Δ_χ est un carré dans $\mathbb{F}_l$, r divise $l - 1$ et ϕ_l agit sur $E[l]$ comme la matrice diagonale $\begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}$ avec $\lambda, \mu \in \mathbb{F}_l^*$.

(3) $(r, \dots, r)$ pour $r > 1$, dans ce cas Δ_χ n'est pas un carré dans $\mathbb{F}_l$, r divise $l + 1$ et χ_l est irréductible sur $\mathbb{F}_l$.

Dans tous les cas r est l'ordre de ϕ_p dans le groupe projectif linéaire $PGL(\mathbb{F}_l)$ et la trace du Frobenius satisfait :

$$t^2 \equiv p(\xi + \xi^{-1})^2 \pmod{l} \tag{13}$$

Où ξ est une r-ième racine primitive de l'unité dans $\bar{\mathbb{F}}_l$.

On rappelle que le $PGL(E)$ d'un espace vectoriel E sur un corps K est le groupe quotient $GL(E)/Z(E)$. Où $GL(E)$ est le groupe général linéaire de E, le groupe des automorphismes de E muni de la composition des applications. Et $Z(E)$ le centre de $GL(E)$, c'est à dire l'ensemble des éléments de E qui commutent avec tous les autres.
Pour déterminer dans quel cas du théorème l'on se trouve, on va étudier la factorisation de $\Phi_l(x, j)$ (j est le j-invariant de notre courbe elliptique E). Pour se faire on va calculer le degré de $\gcd(\Phi_l(x, j), x^p - x)$, d'après le théorème précédent si ce gcd est constant alors l est un premier d'Atkins, sinon c'est un premier d'Elkies.

Premier d'Elkies On se place dans le cas où l est un premier d'Elkies. Le Frobenius est diagonalisable, de valeurs propres λ, μ, car $\Delta_{\chi_l} = t_l^2 - 4p$ est un carré modulo l (on rappelle que $\chi_l(x) = x^2 - t_l x + p_l$). Ainsi on a :

$$\chi_l(x) = (x - \lambda)(x - \mu)$$

Donc $t_l \equiv \lambda + \mu \equiv \lambda + \frac{p_l}{\lambda} \pmod{l}$ (car $p_l = \lambda\mu$). On peut calculer la trace modulo l en trouvant une valeur propre du Frobenius. La première question est de se demander si $\lambda \neq \mu$, si on est dans ce cas alors $t_l = 2\lambda = 2\sqrt{p_l}$ et on se ramène à un cas expliqué dans la partie sur l'algorithme de Schoof. Le cas le plus intéressant est donc quand $\lambda \neq \mu$. Par définition d'une valeur propre on sait qu'il existe deux points $P_1, P_2 \in E[l]$ non triviaux tel que $\phi_p(P_1) = [\lambda]P_1$ et $\phi_p(P_2) = [\lambda]P_2$. Soit C_1 et C_2 les deux sous groupes cycliques d'ordre l générés par P_1 et P_2, ils sont stables par l'action du Frobenius, c'est à dire $\phi_p(C_1) = C_1$. On peut alors construire le polynôme g_l de degré $\frac{(l-1)}{2}$ et divisant le l-ième polynôme de division :

$$g_l(x) = \prod_{\pm P \in C_1^*} (x - x_p), \quad \text{où } P = (x_p, y_p) \tag{14}$$

On peut à partir de ce polynôme utiliser l'algorithme de Schoof pour trouver la trace du Frobenius modulo l. On commence par noter que vu que λ est valeur propre, pour tout point de C_1 on a l'équation $\phi(P) = [\lambda]P$. On peut appliquer nos formules de multiplication scalaire énoncée dans notre partie sur les polynomes de division. L'abscisse du point $[\lambda]P$ est :

$$x - \frac{\psi_{\lambda-1}\psi_{\lambda+1}}{\psi_\lambda^2}$$

Avec $\phi(P) = (x^p, y^p)$ on trouve :

$$x^p = x - \frac{\psi_{\lambda-1}\psi_{\lambda+1}}{\psi_\lambda^2}$$

On pose $h(x) = \psi_\lambda^2(x^p - x) + \psi_{\lambda-1}\psi_{\lambda+1}$, qu'on réduit modulo g_l. Il faut trouver une valeur propre λ tel que $\gcd(h, g_l) \neq 1$. Une fois que l'on a trouvé une valeur qui correspond on peut calculer $t_l = \lambda + \frac{p_l}{\lambda} \pmod{l}$

Il faut maintenant montrer comment construire ce polynôme g_l. L'idée va être de trouver dans un premier temps le premier coefficient de g_l (qui est moins la somme des racines de g_l). On sait que le degré du polynôme $\gcd(\Phi(x, j), x^p - x)$ est plus grand que 0 comme l est un premier de Elkies et que ses racines sont dans $\mathbb{F}_p$. Notons par exemple $\tilde{j}$ l'une de ses racines. On sait donc qu'il existe une isogénie entre E et la courbe elliptique de j-invariant $\tilde{j}$ qu'on note $\tilde{E} : y^2 = x^3 + \tilde{a_4}x + \tilde{a_6}$. Le noyau de cette isogénie est l'un des $l + 1$ sous groupe cyclique de $E[l]$. On va utiliser cette isogénie pour retrouver le premier coefficient de g_l. On ne détaillera pas les calculs permettant d'arriver à ce résultat, ils s'appuient fortement sur l'analyse complexe évoqué plus haut. Plus d'informations peuvent être trouvées ici [2].

Premier d'Atkin On va étudier le degré r de ϕ_p dans $PGL_2(\mathbb{F}_l)$. Il faut dans un premier temps déterminer le plus petit $i > 1$ tel que :

$$\gcd(\Phi_l(x, j), x^{p^i} - x) = \Phi_l(x, j) \tag{15}$$

On recherche donc la plus petite extension $\mathbb{F}^{p^i}$ de $\mathbb{F}_p$ contenant toutes les racines de $\Phi_l(x, j)$. Ce i est alors le degré de ϕ_p, et on peut montrer qu'il suffit de tester les i divisant $l+1$ et qui satisfont $(-1)^{\frac{l+1}{i}} = (\frac{p}{l})$. On rappelle qu'on a une relation entre les racines primitives de l'unité et la trace d'après l'équation 13. On va construire un ensemble de valeurs possibles pour la trace modulo l. Soit λ et μ deux valeurs propres de ϕ_p, alors $\gamma = \frac{\lambda}{\mu}$ est une r-ième racine primitive de l'unité et $\gamma \in \mathbb{F}_{l^2}$. On peut aussi écrire, pour d qui n'est pas un carré de $\mathbb{F}_l$, $\mathbb{F}_{l^2} \cong \mathbb{F}_l[\sqrt{d}]$. On peut alors énumérer toute les valeurs de γ. Soit g un générateur de $\mathbb{F}_{l^2}$ alors $\gamma = g^{\frac{l^2-1}{r}}$ est une r-ième racine de l'unité, pour trouver les autres on calcule $\gamma_i = \gamma^i$. Montrons maintenant comment utiliser ces γ_i pour calculer notre ensemble de trace. Sans détailler, l'idée est d'écrire γ_i sous la forme $\gamma_i = g_{i_1} + g_{i_2}\sqrt{d}$ avec g_{i_1}, g_{i_2} dans $\mathbb{F}_l$, mais on peut aussi montrer que pour $x_1, x_2 \in \mathbb{F}_l$ on a $\lambda = x_1 + x_2\sqrt{d}$ et $\mu = x_1 - x_2\sqrt{d}$. En utilisant la relation $\gamma_i = \frac{\lambda}{\mu}$. Les variables x_i sont inconnues mais pas les g_i, au final on a la relation $x_1^2 = \frac{p(g_{i_1}+1)}{2}$. Si x_i^2 n'est pas un carré dans $\mathbb{F}_l$ on passe au γ_i suivant, sinon sachant que $t \equiv \lambda + \mu = 2x_1$ on ajoute à notre ensemble de trace $2x_1$ et $-2x_1$.

Retrouver t On ne peut pas simplement utiliser les restes chinois comme pour l'algorithme de Schoof. En effet la trace que l'on trouve en utilisant les premiers d'Atkins est un ensemble de traces possibles, on n'a pas unicité. Il y a plusieurs possibilités, la plus simple mais peut être moins efficace est de considérer plus d'entier l pour calculer notre trace modulo l. L'autre idée est d'utiliser un algorithme baby-step/giant-step qui sera plus efficace.

3.4.5 Algorithme pseudo-code

On donne une description en pseudo-code de l'algorithme SEA.

Algorithm 2 Schoof-Elkies-Atkins

Require: E une courbe elliptique de la forme $y^2 = x^3 + ax = b$ sur $\mathbb{F}_p$
Ensure: Le nombre de points de E
  Calculer le j-invariant de E
  Choisir un ensemble de premier S tel que $\prod_{l \in S} l \leqslant 4\sqrt{p}$
  for all $l \in S$ **do**
    if $deg(\gcd(\Phi_l(x, j), x^p - x) = 0)$ **then**
      Calculer r avec le théorème 5
      Calculer d tel que $(\frac{d}{l}) = -1$, et $0 < d < l$
      Calculer g tel que g soit un générateur de $\mathbb{F}_l[\sqrt{d}]^*$
      Calculer $S_2 = \left\{ g^{\frac{i(l^2-1)}{r}} \mid \gcd(i, r) = 1 \right\}$
      for all $\gamma \in S_2$ **do**
        écrit $\gamma_i = g_{i_1} + \sqrt{d} g_{i_2}$
        Calculer $z = \frac{p(g_{i_1}+1)}{2} \pmod{l}$
        if $(\frac{z}{l}) = 1$ **then**
          Calculer $x = \sqrt{z} \pmod{l}$
          $t_l = (2x, -2x)$
        end if
      end for
    else
      Calculer g_l un diviseur de f_l
      Trouver λ valeur propre du Frobenius tel que $\gcd(\psi_\lambda^2(x^p - x) + \psi_{\lambda-1}\psi_{\lambda+1}, g_l(x)) \neq 1$
      Calculer $t_l = \lambda + \frac{\lambda}{p} \pmod{l}$
    end if
  end for
  retrouver t en utlisiant les t_l
  retourner $p + 1 - t$

Complexité L'algorithme SEA a une complexité heuristique de $O(\log^{2+2\epsilon} p)$ en temps, et $O(\log^2 p)$ en espace.

4 Conclusion

Au cours de cette étude on a vu divers algorithmes permettant de compter le nombres de points sur une courbe elliptique défini sur un corps $\mathbb{F}_p$ avec p premier. Tout d'abord un algorithme de complexité exponentielle, avec la méthode brute-force qui consiste simplement à vérifier si pour tout élément de $\mathbb{F}_p$, ce dernier appartient à la courbe. Peu efficace quand p devient trop important il est néanmoins plus rapide que ses homologues polynomiaux quand on se restreint à de petits premiers p. Par la suite avec l'introduction de l'endomorphisme du Frobenius et des polynômes de division, on a pu détailler l'algorithme de Schoof qui lui est de complexité polynomiale. Une implémentation a été faite sous SAGE nous permettant de tester l'algorithme sur des cas pratiques. Il est apparut que même si l'algorithme demande un nombre polynomial d'opérations élémentaires, il est relativement lent du fait de la taille des polynômes de division, de degré en $O(l^2)$. C'est avec cette idée de travailler avec des polynômes de plus petit degré que Elkies et Atkins ont proposé des améliorations à cet algorithme en utilisant plutôt des polynômes de degré $O(l)$, les polynômes modulaires. Cela a rendu l'algorithme nommé SEA (Schoof-Elkies-Atkins) très efficace, même lorsque que p devient très grand. Pour expliquer l'algorithme il a fallut introduire de la théorie d'analyse complexe, expliciter les isogénies de courbe elliptique et les polynômes modulaires. Un algorithme en pseudo-code a seulement été proposé, sans détailler pleinement toutes les étapes.

Références

[1] Jean-Marc Couveignes. Courbes elliptiques. https ://www.math.u-bordeaux.fr/ jcouveig/cours/CSI-ELL.pdf.

[2] Ben Galin. *Schoof-Elkies-Atkins Algorithm.* PhD thesis, Standford University, 2007.

[3] Tom M.Apostol. *Modular Functions and Dirichlet Series in Number Theory.* 1976.

[4] Andrew V. Sutherland Reinier Broker, Kristin Lauter. Modular polynomials via isogeny volcanoes. *Mathematics of Computation 81*, 2010.

[5] René Schoof. Counting points on elliptic curves over finite fields. *Journal de Théorie des Nombres de Bordeaux 7*, 1995.