Compter les points sur une courbe elliptique

Jérémie Coulaud

7 février 2019

Table des matières

1 Introduction **3**

2 Introduction aux courbes elliptiques **3**

3 Compter les points sur une courbe **4**
3.1 Algorithme naif 4
3.2 Shanks 5
3.3 Schoof 5
3.3.1 Choix de l'ensemble de premiers 7
3.3.2 Cas $l = 2$ 7
3.3.3 Calcul des polynômes de division 8
3.3.4 Calcul de $(x^{p^2}, y^{p^2}) + [p_l](x, y)$ 8
3.3.5 Algorithme 9
3.3.6 Exemple 10
3.3.7 Expérimentations 12
3.3.8 Complexité 13
3.4 Algorithme SEA 13
3.4.1 Analyse complexe 13
3.4.2 Isogénie 14
3.4.3 Polynôme modulaire 14

1 Introduction

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. L'ensemble des codes sont 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'objet mathématiques.

2 Introduction aux courbes elliptiques

La cryptographie sur courbe elliptique introduite en 1985 a révolutionné la cryptographie à clé publique permettant une alternative efficace aux classique 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. On se contentera de donner une explication succincte de ce qu'est une courbe elliptique ainsi que différents 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 coefficient $a_{1\leq i\leq 6}$ sont des éléments du corps K.

Dans le cas ou le corps K a une caractéristique différent 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 :

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

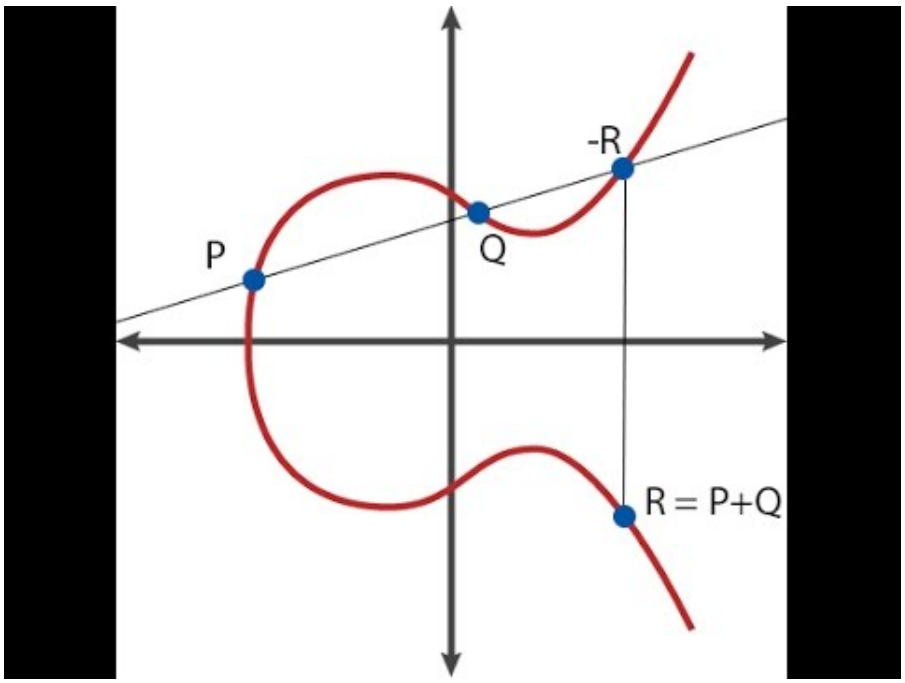


FIGURE 1 – vision graphique de l'addition de deux points sur la courbe

Maintenant que l'on dispose de formule 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.

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

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 naif

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 :

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

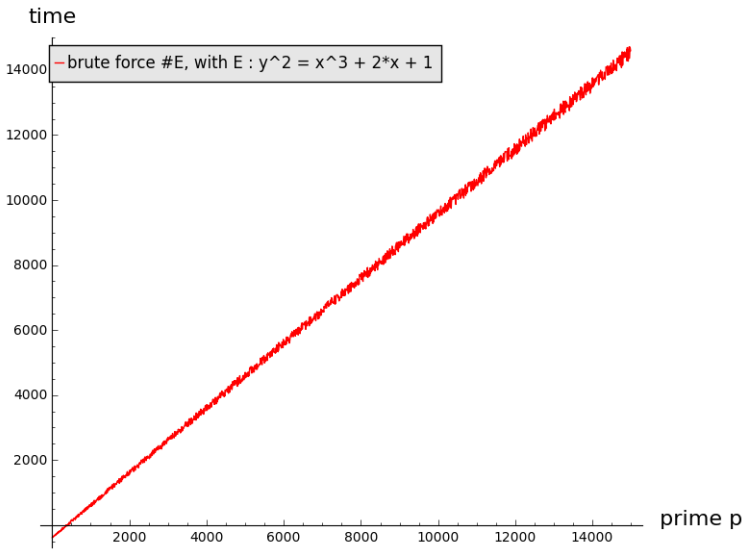
Soit :

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

On peut implémenter cette fonction brute force tres 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.

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 $|t| \leq 2\sqrt{p}$ 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 $Tr(\phi_p) = t = r_1 + r_2$.
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 :

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

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 $0 \leq t_l, p_l < l$.
Il nous faut maintenant introduire les polynôme 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 définit 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 :

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

Ces polynômes sont de degrés 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 $m \in \mathbb{N}^*$.*

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

En posant :

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

et

$$\left\{ \begin{array}{ll} \phi_m & = x\psi_m^2 - \psi_{m-1}\psi_{m+1} \\ \psi_m\omega_m & = \psi_{2m} \end{array} \right.$$

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

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

Démonstration. On veut démontrer 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 à 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 polynôme utile 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 $\mathbb{W} = \frac{\mathbb{F}_l[x,y]}{(f_l(x), E(x,y))}$. 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 vrai 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 $S = (l_1, \ldots l_n)$ tel que $\prod l_i > 4\sqrt{p}$. En effet on rappelle que $|t| \leq 2\sqrt{p}$, 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 est initialisé à 1 et `list_l` correspond à S :

```
while N <= 4*sqrt(p):
        if prime == p:
                pass
        prime = next_prime(prime)
        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 $E(\mathbb{F}_p)$ à un élément d'ordre 2, c'est à dire un point de 2-torsion. Or les points de 2 torsions 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 déterminée par le polynôme $x^p - x$, nos deux polynômes partageraient alors une racine commune. La façon utilisé ici pour décider si $t_2 \equiv 0 \pmod 2$ est donc de calculer si $\gcd(x^p - x, x^3 + ax + b) \neq 1$ (qu'on calcule par exponentiation rapide).

3.3.3 Calcul des polynômes de division

Il existe dans SAGE une fonction permettant de calculer les polynôme de division, `polynomial_polynomial(n)` renvoyant le n-ième polynôme de division. Mais comme au cours de l'algorithme on doit souvent utiliser différents polynôme de division et de plus ils sont calculé 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 où ont la même coordonnée en x. En effet les formules à appliquer seront différentes selon que l'on soit dans un cas ou l'autre. Il est plus probable d'avoir des coordonnées différentes, on va donc travailler dans ce cas la. 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 $l - 1$ 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 : $(x^{p^2}, y^{p^2}) = \pm[p_l](x, y)$ La encore deux cas possible. 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 :

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

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 :

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

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 on a alors $t_l = 0$ d'après l'équation caractéristique.

3.3.5 Algorithme

On donne une version complète en langage naturel 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 $\prod_{l \in S} l \leq 4\sqrt{p}$
  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 $1 \leq \tau \leq l - 1$ **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 fallut commencer par implémenter des fonctions de base sur les courbes elliptique pour additionner, calculer une multiplication scalaire. Les algorithmes d'additions ne nécessite 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, donc je suis resté avec un version plus simple d'exponentiation. L'idée de cet algorithme est d'utilisation 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 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 entre 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 détails complet de l'algorithme 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 est pas certains que ce polynôme soit divisible, 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 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(reversed(factor(poly_divi_l))):
                if poly.degree() > 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)
```

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 ces 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 $(x_3, y_3) = (x^{361}, y^{361}) + (x, y)$:

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

et,

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

(reprendre le calcul de y_3, le résultat final est juste mais erreur dans les calcules intermédiaire !

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

3.3.7 Expérimentations

Faire les test de l'algo + comparaison avec brute force/methode native sage Inserer les graphiques + commenter

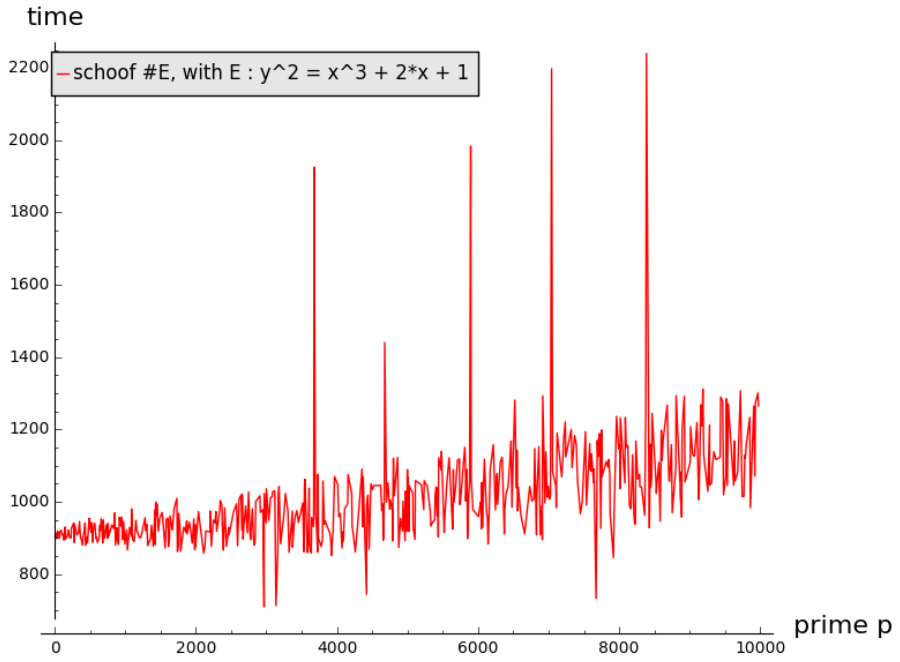
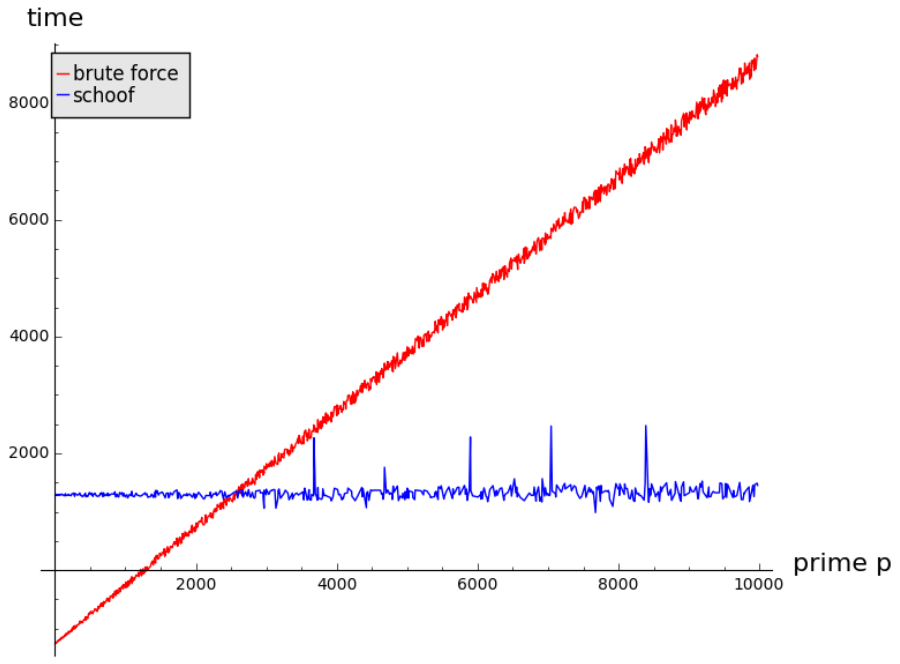


FIGURE 2 – Temps d'exécution de l'algorithme de Schoof

FIGURE 3 – Comparaison temps exécution entre les algorithme 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.

3.3.8 Complexité

La complexité de l'algorithme de Schoof est en $O(\log q^8)$. La partie conteuse de l'algorithme réside dans les calculs dans l'anneau $R = \frac{\mathbb{F}_l[x,y]}{(f_l(x), y^2 - x^3 - ax - b)}$. 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 limité 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 grand, ce qui pose un problème de mémoire. Une amélioration de cet algorithme a été proposé 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 modulaire. Pour les définir il va être nécessaire d'introduire un peu de théorie d'analyse complexe.
Soit K un corps fini, on va noter dans la suite E/K la courbe elliptique définie sur ce corps et on ne travaillera uniquement avec des courbes sous forme réduite.

Définition 6. *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 isomorphe 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 rationnelles de nos courbes, et à leurs relations.

Proposition 2. *Soit E_1/K et E_2/K deux courbes elliptiques isomorphe. On a alors un morphisme de groupe entre $E_1(K)$ et $E_2(K)$.*

3.4.1 Analyse complexe

La théorie des courbes elliptiques est importe sur le corps des complexes mais nous allons uniquement donner des définitions et formules utiles sans rentrer dans le détails de preuve et résultats non pertinent pour la compréhension de l'algorithme SEA. On peut légitimement se demander le rapport entre courbe elliptique et analyse complexe, dans un premier temps il est nécessaire d'introduire un peu de théorie d'analyse complexe et rappeler quelques définitions avant de faire le lien entre les deux.

Définition 7. *Une fonction holomorphe est une fonction à valeur complexes, définies et dérivable en tout points 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 3. *Soit $n \in \mathbb{N}$, $r \in \mathbb{R}$, $(w_i)_{1 \leq i \leq r}$ 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 isomorphe à $\mathbb{Z}$ sont de la forme $\mathbb{Z}\omega_1 + \mathbb{Z}w_2$ avec $\omega_1, \omega_2 \in \mathbb{C}$ et $Im(\frac{w_2}{w_1}) \neq 0 = .$

Définition 8. *On appelle tore le quotient $T = \mathbb{C}/\Gamma$. Cest 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 9. *Soit Γ un réseau de $\mathbb{C}$, on introduit la série :*

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

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

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

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

Le théorème suivant permet de faire le lien entre ces fonctions d'analyse complexe et les courbes elliptique.

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 :*

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

3.4.2 Isogénie

3.4.3 Polynôme modulaire