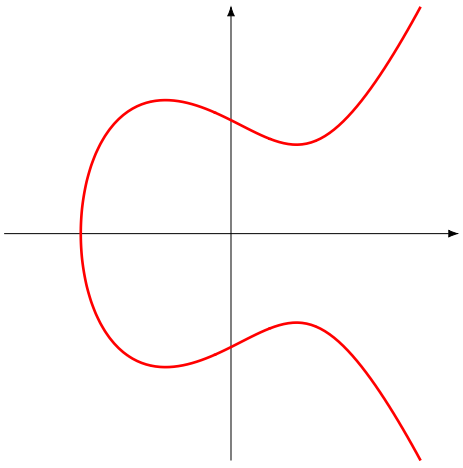
TIPE

# CRYPTOGRAPHIE SUR COURBES ELLIPTIQUES (ECC)

Baptiste Pasquier

2017-2018



*Lycée privé Sainte-Geneviève*

# Sommaire

# Introduction

Le mot cryptographie provient des racines grecques *kruptos* « caché » et *graphein* « écrire », la cryptographie est la science de la protection des messages. Elle occupe aujourd'hui une place prépondérante dans de nombreux domaines où le partage d'informations confidentielles est important : défense, commerce, vie privée...

Les premières méthodes de chiffrement remontent à l'Antiquité. L'une des plus connue est le Chiffre de César, un algorithme de substitution qui consiste en un décalage de l'alphabet pour rendre le message illisible. Bien évidemment, la sécurité est très faible puisqu'il n'y a que 25 clés possibles ! Ce n'est qu'avec l'avènement des communications puis de l'informatique au XX$^{e}$ siècle que la cryptographie est devenue une véritable science.

Aujourd'hui, il est nécessaire de distinguer deux notions : la cryptographie symétrique et la cryptographie asymétrique. La cryptographie symétrique utilise des algorithmes de chiffrement où la clé de chiffrement est identique à la clé de déchiffrement. Un des problèmes de ce système est de pouvoir transmettre la clé au destinataire de manière sécurisée.

La cryptographie à clé publique ou asymétrique, proposée par Whitfield Diffie et Martin Hellman en 1976, tente de résoudre ce problème grâce à un couple de clé publique et clé secrète ainsi que d'un certain problème mathématique. La cryptographie RSA inventée en 1977 et encore beaucoup utilisée aujourd'hui se base ainsi sur la difficulté de factoriser un très grand nombre.

L'utilisation des courbes elliptiques pour la cryptographie asymétrique a été proposée pour la première fois par Miller et Klobitz en 1985. Au lieu d'effectuer des calculs dans le groupe $\mathbb{Z}/n\mathbb{Z}$ comme pour le chiffrement RSA, on s'intéresse ici à des groupes associés à une courbe elliptique. Les principaux avantages sont la réduction de la taille des clés cryptographiques par rapport à d'autres méthodes de cryptographie et une hausse de la sécurité.

Nous essaierons donc de comprendre comment les courbes elliptiques fournissent un cadre utile pour la cryptographie.

# 1 Courbes elliptiques

## 1.1 Polynômes à plusieurs indéterminées

Nous introduisons dans cette première partie quelques notions sur les polynômes à plusieurs indéterminées que nous utiliserons par la suite.

Soit $\mathbb{K}$ un corps.

**Définition 1 :** *Monôme*

Soit $n \in \mathbb{N}^*$.

Un *monôme* en les indéterminées $X_1, \ldots, X_n$ est un produit de la forme :

$$X_1^{a_1} \cdots X_n^{a_n}$$

où $a_1, \ldots, a_n$ sont des entiers naturels.

Le *degré* d'un monôme est défini comme la somme :

$$\sum_{i \in [\![1\,,\,n]\!]} a_i$$

**Définition 2 :** *Polynôme*

Un *polynôme* en les indéterminées $X_1, \ldots, X_n$ à coefficient dans $\mathbb{K}$ est une combinaison linéaire finie de monômes d'indéterminées $X_1, \ldots, X_n$.

**Proposition 1**

L'ensemble des polynômes d'indéterminées $X_1, \ldots, X_n$ forme un anneau commutatif, noté $\mathbb{K}[X_1, \ldots, X_n]$.

- Admis -

**Définition 3 :** *Degré d'un polynôme*

Le degré d'un polynôme non nul $P$ est le maximum des degrés des monômes non nuls qui le composent.

**Définition 4 :** *Polynôme homogène*

Un polynôme $P$ est dit *homogène de degré $p$* si chacun des monômes le constituant est de degré $p$.

**Définition 5 :** *Polynôme irréductible*

Un polynôme $P \in \mathbb{K}[X_1, \ldots, X_n]$ est dit *irréductible* s'il ne peut pas s'écrire comme le produit non trivial de deux polynômes de $\mathbb{K}[X_1, \ldots, X_n]$

**Définition 6 :** *Dérivée partielle d'un polynôme*

Soit un polynôme $P \in \mathbb{K}[X_1, \ldots, X_n]$.

La *dérivée partielle* du polynôme $P$ par rapport à $X_i$ est la dérivée du polynôme $P$ en considérant les autres indéterminées comme des constantes.

On la note $\dfrac{\partial F}{\partial X}$.

## 1.2 Géométrie projective

Le cadre idéal pour l'étude des courbes elliptiques est celui de la géométrie projective dont nous allons donner quelques notions.

**Définition 7 :** *Espace projectif*

Soit $\mathbb{K}$ un corps commutatif. Pour $n \in \mathbb{N}^*$, on définit la relation d'équivalence $\mathcal{R}$ sur $\mathbb{K}^n \backslash \{(0,\dots,0)\}$ par :

$$\forall (u,v) \in (\mathbb{K}^n \backslash \{(0,\dots,0)\})^2, \quad u \,\mathcal{R}\, v \iff \exists \lambda \in \mathbb{K}^* : u = \lambda v$$

L'espace projectif $\mathbb{P}^{n-1}(\mathbb{K})$ est l'ensemble quotient $(\mathbb{K}^n \backslash \{(0,\dots,0)\})/\mathcal{R}$.

L'espace projectif $\mathbb{P}^2(\mathbb{K})$ est appelé le *plan projectif* sur $\mathbb{K}$.

Les coordonnées d'un point dans l'espace projectif sont appelées *coordonnées homogènes*. Les coordonnées homogènes d'un point ne sont pas uniques et sont définies à une constante multiplicative près.

Soit $(x,y) \in \mathbb{P}^n(\mathbb{K})$ tel que $x = [x_0,\dots,x_n]$ et $y = [y_0,\dots,y_n]$, dans ce cas,

$$x = y \iff \exists \lambda \in \mathbb{K}^*,\ \forall i \in [\![0\,,\,n]\!],\ y_i = \lambda x_i$$

Ainsi la relation $\mathcal{R}$ identifie les éléments d'une même droite vectorielle.

- Si $x_n \neq 0$, on peut écrire $x$ sous la forme $x = [x_0,\dots,x_{n-1},1]$. On peut alors construire une bijection entre l'ensemble $U$ des points dont la dernière coordonnée est non nulle et l'espace $\mathbb{K}^n$.

$$U = \{[x_0,\dots,x_n] \in \mathbb{P}^n(\mathbb{K}) \ \mid \ x_n \neq 0\}$$

On pose l'application

$$\varphi : \begin{cases} U & \longrightarrow & \mathbb{K}^n \\ [x_0,\dots,x_n] & \longmapsto & \left(\dfrac{x_0}{x_n},\dots,\dfrac{x_{n-1}}{x_n}\right) \end{cases}$$

L'application $\varphi$ est bijective, sa bijection réciproque est :

$$\varphi^{-1} : \begin{cases} \mathbb{K}^n & \longrightarrow & U \\ (x_0,\dots,x_{n-1}) & \longmapsto & [x_0,\dots,x_{n-1},1] \end{cases}$$

- Si $x_n = 0$, tous les points de la forme $[x_0,\dots,x_{n-1},0]$ correspondent à l'hyperplan d'équation $x_n = 0$. Cet hyperplan est appelé *hyperplan à l'infini.*

Ainsi, on peut considérer que l'espace projectif $P^n(\mathbb{K})$ est la réunion de l'espace affine $\mathbb{K}^n$ et de l'hyperplan à l'infini.

Dans le cas du plan projectif, on obtient :

$$\mathbb{P}^2(\mathbb{K}) = \mathbb{K}^2 \cup D_\infty$$

avec $D_\infty$ la droite à l'infini.

**Définition 8 :** *Courbe projective plane*

Une courbe projective plane $C$ est l'ensemble des points $[X, Y, Z]$ de $\mathbb{P}^2(\mathbb{K})$ qui vérifient une équation :

$$F(X, Y, Z) = 0$$

où $F$ est un polynôme homogène de degré $d \geq 1$.

Le *degré de la courbe* $C$ est définie comme le degré du polynôme $F$.

Si $F$ est un polynôme irréductible, la courbe $C$ est elle-même dite *irréductible*.

Si $d = 1$, $C$ est un droite. Si $d = 2$, $C$ est une conique. Si $d = 3$, $C$ est une **cubique**.

**Définition 9 :** *Point singulier*

Soient $C$ une courbe projective plane d'équation $F(X, Y, Z) = 0$ et $P = [x_P, y_P, z_P] \in C$.

P est un *point singulier* si :

$$\frac{\partial F}{\partial X}(x_P, y_P, z_P) = \frac{\partial F}{\partial Y}(x_P, y_P, z_P) = \frac{\partial F}{\partial Z}(x_P, y_P, z_P) = 0$$

**Définition 10 :** *Courbe non singulière*

Une courbe projective plane $C$ est *non singulière* (ou *lisse*) si tous les points de $C$ sont non singuliers.

## 1.3 Courbe elliptique

**Définition 11 :** *Courbe elliptique*

Une courbe elliptique $E$ sur un corps $\mathbb{K}$ est une cubique irréductible non singulière dans le plan projectif $\mathbb{P}^2(\mathbb{K})$, munie d'un point d'origine. Elle est définie par une équation $F(X, Y, Z) = 0$ où F un polynôme de degré 3, homogène et irréductible, à 3 variables dans $\mathbb{K}$ :

$$F(X, Y, Z) = aX^3 + bY^3 + cZ^3 + dX^2Y + eX^2Z + fY^2X + gY^2Z + hZ^2X + iZ^2Y + jXYZ = 0$$

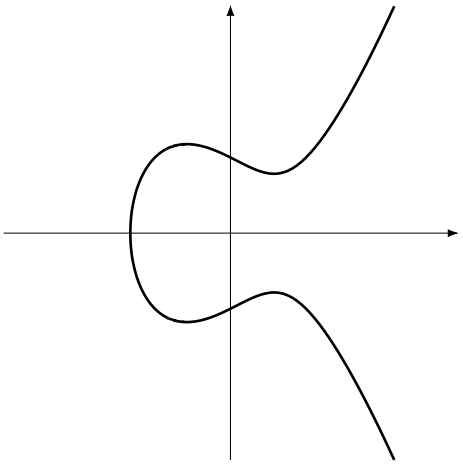


FIGURE 1 – Une courbe elliptique

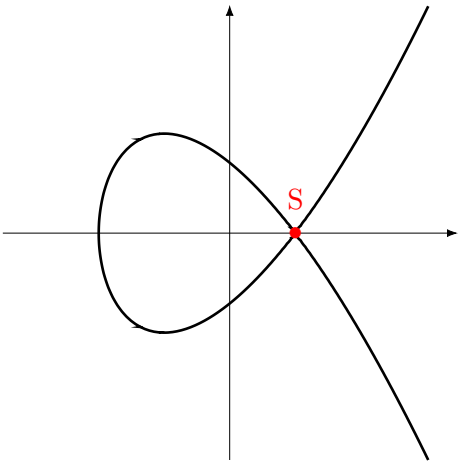



FIGURE 2 – Une courbe singulière

**Remarque :**

Avec les mêmes notations, l'homogénéité du polynôme assure que si un point $P$ est sur la courbe $E$, alors tous les points de sa classe d'équivalence le sont aussi :

Soit $P = [x_P, y_P, z_P] \in E$.

Pour $\lambda \in \mathbb{K}$, $F(\lambda x_P, \lambda y_P, \lambda z_P) = \lambda^3 F(x_P, y_P, z_P) = 0$.

**Définition 12 :**

Une courbe elliptique E est sous forme de Weierstrass si son équation est de la forme :

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

**Proposition 2**

Pour toute courbe elliptique E, il est possible de se ramener à une courbe sous forme de Weierstrass.

- Admis -

Dans un corps $\mathbb{K}$ de caractéristique différente de 2 ou 3, on peut ensuite se ramener à une forme plus simple :

$$Y^2Z = X^3 + aXZ^2 + bZ^3$$

grâce aux changements de variable :

$$Y \leftarrow \left\{Y - \frac{1}{2}(a_1X + a_3)\right\} \quad \text{et} \quad X \leftarrow \left\{X - \frac{a_1^2 + 4a_2}{12}Z\right\}$$

On considère à présent $\mathbb{K}$ un corps de caractéristique différente de 2 ou 3.

**Partie affine et point à l'infini :**

Soit $E$ une courbe elliptique d'équation $Y^2Z = X^3 + aXZ^2 + bZ^3$.

On pose

$$U = \{[x, y, z] \in E \ / \ z \neq 0\}$$

On a alors

$$U = \left\{(x, y, z) \in \mathbb{P}^2(\mathbb{K}) \ / \ z \neq 0 \ \wedge \ y^2z = x^3 + axz^2 + bz^3\right\}$$

On passe en coordonnées non homogènes $x \leftarrow \frac{x}{z}$ et $y \leftarrow \frac{y}{z}$ pour se ramener dans le plan affine :

$$U = \{(x, y) \in \mathbb{K}^2 \ / \ y^2 = x^3 + ax + b\}$$

Par ailleurs,

$$\begin{aligned}\left\{(x, y, z) \in \mathbb{P}^2(\mathbb{K}) \ / \ z = 0 \ \wedge \ y^2z = x^3 + axz^2 + bz^3\right\} &= \left\{(x, y, z) \in \mathbb{P}^2(\mathbb{K}) \ / \ x = 0 \ \wedge \ z = 0\right\} \\ &= \left\{P \in P^2(\mathbb{K}) \ / \ P = [0, \lambda, 0], \ \lambda \in \mathbb{K}^*\right\}\end{aligned}$$

Cet ensemble est équivalent dans le plan projectif au point $\mathcal{O} = [0, 1, 0]$.

On dit alors que $U$ est la *partie affine* de $E$ et que $\mathcal{O}$ est le *point à l'infini* de $E$.

**Proposition 3**

Une courbe donnée par une équation de Weierstrass est non singulière si et seulement si son discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$.

**Démonstration :**

On considère $E$ une courbe elliptique d'équation $F(X, Y, Z) = Y^2Z - X^3 - aXZ^2 - bZ^3 = 0$.

Montrons que le point à l'infini $\mathcal{O} = [0, 1, 0]$ n'est pas singulier.

On a

$$\frac{\partial F}{\partial Z} = Y^2 - 2aXZ - 3bZ^2$$

ainsi,

$$\frac{\partial F}{\partial Z}(0, 1, 0) = 1 \neq 0$$

Donc $\mathcal{O}$ n'est pas un point singulier de E.

Considérons maintenant les autres points. On utilise alors la partie affine en donnant une équation de $E$ sous la forme

$$F(X, Y) = Y^2 - X^3 - aX - b = 0$$

$$\begin{aligned}
\text{E est singulière} &\iff \forall P \in E,\ P \text{ est singulier} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} y_0^2 = x_0^3 + ax_0 + b \\ \frac{\partial F}{\partial Y}(P) = 0 \\ \frac{\partial F}{\partial X}(P) = 0 \end{cases} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} y_0^2 = x_0^3 + ax_0 + b \\ 2y_0 = 0 \\ 3x_0^2 + a = 0 \end{cases} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} y_0^2 = x_0^3 + ax_0 + b \\ 2y_0 = 0 \\ 3x_0^2 + a = 0 \end{cases} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} y_0^2 = x_0^3 + ax_0 + b \\ y_0 = 0 \\ x_0^2 = -\frac{a}{3} \end{cases} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} x_0^3 + ax_0 + b = 0 \\ y_0 = 0 \\ x_0^2 = -\frac{a}{3} \end{cases} \\
&\iff \forall P = (x_0, y_0),\ \begin{cases} y_0 = 0 \\ -\frac{a}{3}x_0 + ax_0 + b = 0 \\ x_0^2 = -\frac{a}{3} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\iff \forall P = (x_0, y_0), \begin{cases} y_0 = 0 \\ x_0 = -\frac{3b}{2a} \\ x_0^2 = -\frac{a}{3} \end{cases} \\
&\iff \frac{9b^2}{4a^2} + \frac{a}{3} = 0 \\
&\iff 27b^2 + 4a^3 = 0 \\
&\iff \Delta = 0
\end{aligned}$$

Ainsi, E est non singulière si et seulement si $\Delta \neq 0$.

❑

Après avoir étudié la construction des courbes elliptiques, nous donnons à présent 2 définitions équivalentes. Dans la suite, toute courbe elliptique correspondra indifféremment à l'une ou l'autre de ces définitions [1].

**Définition 13 :** *Définition projective : Courbe elliptique*

Une courbe elliptique $E$ est le sous-ensemble du plan projectif $\mathbb{P}^2(\mathbb{K})$ défini par :

$$E = \left\{ [x, y, z] \in \mathbb{P}^2(\mathbb{K}) \ / \ y^2 z = x^3 + axz^2 + bz^3 \right\}$$

avec $(a, b) \in \mathbb{K}^2$ tel que $\Delta = -16(4a^3 + 27b^2) \neq 0$.

**Définition 14 :** *Définition affine : Courbe elliptique*

Une courbe elliptique $E$ est l'ensemble défini par :

$$E = \{(x, y) \in \mathbb{K}^2 \ / \ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

avec $(a, b) \in \mathbb{K}^2$ tel que $\Delta = -16(4a^3 + 27b^2) \neq 0$, et $\mathcal{O}$ le *point à l'infini* de la courbe.

## 1.4 Structure de groupe

Une propriété remarquable des courbes elliptiques est que l'on peut leur adjoindre une structure de groupe commutatif.

### 1.4.1 Droites dans $\mathbb{P}^2(\mathbb{K})$

**Définition 15 :** *Droite projective*

Une *droite projective* est une courbe projective plane de degré 1. Une droite projective est donc l'ensemble des points $[x, y, z] \in \mathbb{P}^2(\mathbb{K})$ vérifiant une équation :

$$ax + by + cz = 0$$

avec $(a, b, c) \in \mathbb{K}^3$ et $(a, b, c) \neq (0, 0, 0)$.

Dans le plan affine, une droite verticale a pour équation $x = \lambda$, dans le plan projectif, cette équation devient $x = \lambda z$ par homogénéisation. Ainsi les droites verticales du plan projectif sont les droites projectives d'équation $x = \lambda z$.

1. Nous avons vu précédemment comment s'effectue le passage en coordonnées non homogènes dans le plan affine.

**Proposition 4 :** *Lemme*

Soit $D$ une droite projective.
Si $D$ est une droite verticale, alors le point $\mathcal{O} = [0, 1, 0]$ appartient à $D$.

**Démonstration :**

$D$ est une droite verticale, donc il existe $\lambda \in \mathbb{K}$ tel que l'équation de $D$ soit $x = \lambda z$. On a bien $0 = \lambda 0$ donc le point $\mathcal{O} = [0, 1, 0]$ appartient à $D$.

❑

**Remarque :**

La réciproque est fausse car le point $\mathcal{O}$ appartient aussi à la droite à l'infini d'équation $z = 0$ qui n'est pas verticale. On peut seulement dire que les droites projectives passant par le point $\mathcal{O} = [0, 1, 0]$ ont une équation de la forme $ax + cz = 0$.

**Proposition 5**

Soient $P = [x_P, y_P, z_P]$ et $Q = [x_Q, y_Q, z_Q]$ deux points *distincts* de $\mathbb{P}^2(\mathbb{K})$.
Il existe une **unique** droite projective $D$ passant par $P$ et $Q$.
La droite $D$ est l'ensemble des points $[x, y, z] \in \mathbb{P}^2(\mathbb{K})$ vérifiant l'équation :

$$\begin{vmatrix} x_P & x_Q & x \\ y_P & y_Q & y \\ z_P & z_Q & z \end{vmatrix} = 0$$

La droite $D$ a donc pour équation :

$$ax + by + cz = 0$$

avec $a = y_P z_Q - z_P y_Q$, $b = z_P x_Q - x_P z_Q$, $c = x_P y_Q - y_P x_Q$.

- Admis -

**Remarque :**

Nous retiendrons et utiliserons simplement les deux résultats suivants qui en découlent :

• Pour deux points distincts de $\mathbb{P}^2(\mathbb{K})$, il existe une unique droite passent par ces 2 points.

• Pour $P = [x_P, y_P, z_P]$ et $Q = [x_Q, y_Q, z_Q]$ deux points distincts de $\mathbb{P}^2(\mathbb{K})$, la droite projective passant par $P$ et $Q$ est verticale si et seulement si $x_P z_Q = z_P x_Q$. Dans le cas où $z_P = z_Q = 1$, la condition devient $x_P = x_Q$, ce qui est cohérent avec la notion de droite verticale dans le plan affine $\mathbb{K}^2$.

### 1.4.2 Tangente à une courbe elliptique

La tangente à une courbe étant difficile à construire en géométrie projective, nous donnons simplement une définition de celle-ci.

**Définition 16 :**

Soit $E$ une courbe elliptique d'équation $F(X,Y,Z) = Y^2Z - (X^3 + aXZ^2 + bZ^3) = 0$.

Soit $P = [x_P, y_P, z_P] \in E$.

La tangente à la courbe E au point $P$ est la droite d'équation :

$$\frac{\partial F}{\partial X}(x_P, y_P, z_P)x + \frac{\partial F}{\partial Y}(x_P, y_P, z_P)y + \frac{\partial F}{\partial Z}(x_P, y_P, z_P)z = 0$$

**Remarque :**

Par définition une courbe elliptique $E$ est non singulière, on a donc

$$\forall P \in E, \quad \left(\frac{\partial F}{\partial X}(P), \frac{\partial F}{\partial Y}(P), \frac{\partial F}{\partial Z}(P)\right) \neq (0,0,0)$$

Ainsi, la tangente à la courbe $E$ est bien définie en tout point de $E$.

### 1.4.3 Loi d'addition

La multiplicité d'intersection de deux courbes projectives est complexe à définir, mais nous allons en donner quelques notions.

**Définition 17 :** *Multiplicité d'intersection*

La multiplicité d'intersection est un entier non négatif.

Soient $C$ et $D$ deux courbes algébriques et $P$ un point, la multiplicité d'intersection de $C$ et $D$ en $P$ est strictement positive si et seulement si $P$ appartient aux deux courbes.

Par la suite, nous aurons simplement besoin de la multiplicité d'intersection d'une droite et d'une courbe elliptique.

Soient $C$ une courbe algébrique et $D$ une droite projective. On suppose que $P$ est un point de $C$ et $D$. Si la tangente à la courbe $C$ est distincte de la droite $D$, alors la multiplicité d'intersection de $C$ et $D$ en $P$ vaut 1, sinon elle est strictement supérieure à 1.

Pour notre utilisation, nous pouvons assimiler la multiplicité d'intersection d'une courbe projective et d'une droite projective à celle d'une courbe et d'une droite dans le plan réel :

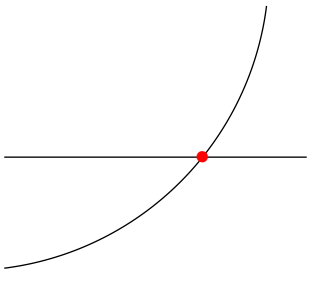


FIGURE 3 – Droite sécante
Multiplicité d'intersection 1

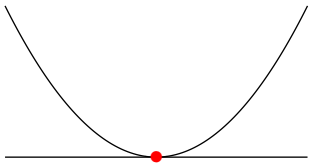


FIGURE 4 – Tangente simple
Multiplicité d'intersection 2

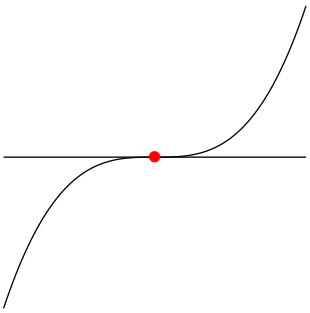


FIGURE 5 – Tangente inflexionnelle
Multiplicité d'intersection $\geq 3$

**Proposition 6 :** *Théorème de Bézout faible*

Soient $C$ et $D$ deux courbes projectives planes de degrés $m$ et $n$ définies sur un corps $\mathbb{K}$ telles que $C$ et $D$ n'ont pas de facteur irréductible commun. Dans ce cas, $C$ et $D$ ont au plus $m \times n$ points d'intersection.

- Admis -

**Remarque :**

La version forte du théorème de Bézout requiert en plus l'hypothèse que le corps $\mathbb{K}$ est **algébriquement clos**, et affirme dans ce cas que $C$ et $D$ ont exactement $m \times n$ points d'intersection, comptés avec leur multiplicité.

Pour notre étude, nous avons seulement besoin de la version faible et nous restons donc dans un corps $\mathbb{K}$, de caractéristique différente de 2 ou 3, algébriquement clos ou non.

**Proposition 7**

Soient $E$ une courbe elliptique et $D$ une droite projective.
Si la droite $D$ coupe la courbe $E$ en au moins deux points d'intersection (comptés avec leur multiplicité), alors elle coupe la courbe E en exactement trois points d'intersection (comptés avec leur multiplicité).

**Démonstration :**

Soit $E$ une courbe elliptique d'équation $F(X,Y,Z) = Y^2Z - (X^3 + aXZ^2 + bZ^3) = 0$. Il existe $(\alpha, \beta, \gamma)$ tel que $D$ ait pour équation $\alpha x + \beta y + \gamma z = 0$.

On note $P$ et $Q$ les deux premiers points d'intersection.

1er cas : si $\mathcal{O} \notin D$, on passe alors en coordonnées non homogènes. Soit $R = (x, y)$, R est un point d'intersection s'il vérifie le système :

$$\begin{cases} y^2 = x^3 + ax + b \\ \alpha x + \beta y + \gamma = 0 \end{cases}$$

Comme la droite $D$ n'est pas verticale ($\mathcal{O} \notin D$), on peut se ramener à l'équation $y = \lambda x + \nu$. On substitue alors $y = \lambda x + \nu$ dans l'équation $y^2 = x^3 + ax + b$ et on obtient un polynôme de degré 3. On connait déjà deux racines ce polynôme [a], et comme il est de degré 3, il existe une troisième racine. On a ainsi trouvé un troisième point d'intersection de la droite $D$ et de la courbe $E$. La version faible du théorème de Bézout nous montre qu'il n'existe pas d'autre point d'intersection.

2e cas : si $\mathcal{O} \in D$.

1er sous-cas : si $P = Q$,

La droite $D$ est alors tangente à la courbe $E$ au point $P$, le point $P$ a donc une multiplicité au moins égale à 2.

a) Si $P \neq \mathcal{O}$, d'après la version faible du théorème de Bézout, la multiplicité de $P$ est égale à 2.

b) Si $P = \mathcal{O}$, on admet que le point $\mathcal{O}$ a une multiplicité de 3.

2e sous-cas : si $P \neq Q$,

a) Si $P \neq \mathcal{O}$ et $Q \neq \mathcal{O}$, on a trouvé 3 points d'intersections distincts, et d'après la version faible du théorème de Bézout, ils ont chacun une multiplicité de 1.

b) Si $P = \mathcal{O}$, on admet que le droite $D$ est verticale. Le point $P$ a soit une multiplicité de 1, et on trouve facilement le troisième point d'intersection [b], soit une multiplicité de 2 d'après la version faible du théorème de Bézout.

❑

a. Deux racines distinctes si $P \neq Q$, ou une racine de degré 2 si $P = Q$ car la droite $D$ sera dans ce cas tangente à $E$.

b. Voir la démonstration de la proposition 10, ce point correspond en fait à l'opposé du point $P$.

**Notations :**

On considère une courbe elliptique $E$ ainsi que deux points $P$ et $Q$ de la courbe. On note :

$$(PQ) = \begin{cases} \text{la droite passant par } P \text{ et } Q \text{ si } M \neq Q, \\ \text{la tangente à la courbe } E \text{ en } P \text{ sinon.} \end{cases}$$

On note $P * Q$ le troisième point d'intersection de la droite $(PQ)$ avec E qui existe d'après la proposition 7.

**Définition 18 :** *Loi de composition* $+$

Soit $E$ une courbe elliptique.

On définit la loi de composition interne appelée **addition** $+ : E \times E \to E$ par :

$$+ : \begin{cases} E \times E & \longrightarrow & E \\ (M, N) & \longmapsto & \mathcal{O} * (M * N) \end{cases}$$

On notera $M + P$ pour $+(M, N)$.

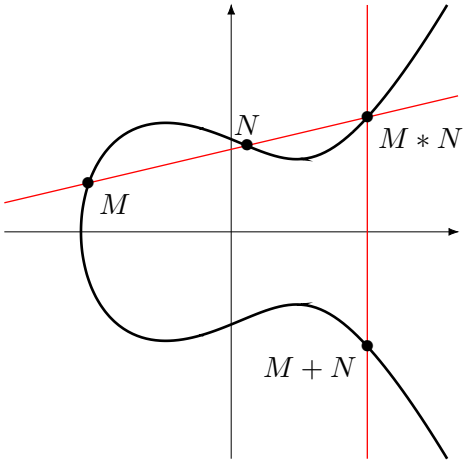



FIGURE 6 – Loi d'addition pour $M \neq N$

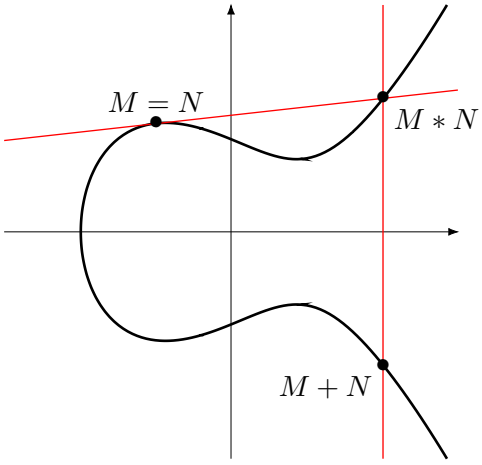



FIGURE 7 – Loi d'addition pour $M = N$

**Proposition 8 :** *Lemme*

Soit $E$ une courbe elliptique, on a

$$\mathcal{O} * \mathcal{O} = \mathcal{O}$$

**Démonstration :**

Soit $E$ une courbe elliptique d'équation $F(X,Y,Z) = Y^2Z - (X^3 + aXZ^2 + bZ^3) = 0$.

$\mathcal{O} * \mathcal{O}$ correspond au troisième point d'intersection de la tangente à la courbe $E$ au point $\mathcal{O}$ avec $E$.

D'après la définition 16, l'équation de la tangente à la courbe $E$ est :

$$\frac{\partial F}{\partial X}(\mathcal{O})x + \frac{\partial F}{\partial Y}(\mathcal{O})y + \frac{\partial F}{\partial Z}(\mathcal{O})z = 0$$

On note $I$ l'intersection de la courbe $E$ avec sa tangente au point $\mathcal{O} = [0,1,0]$.

Soit $P = (x_P, y_P, z_P) \in \mathbb{P}^2(K)$.

On rappelle que $\mathbb{P}^2(K)$ est l'ensemble quotient $(\mathbb{K}^3 \backslash \{(0,0,0)\})/\mathcal{R}$, ainsi $(x_P, y_P, z_P) \neq (0,0,0)$.

$$\begin{aligned}
P \in I &\iff \begin{cases} y_P^2 zP - (x_P^3 + ax_P z_P^2 + bz_P^3) = 0 \\ \dfrac{\partial F}{\partial X}(\mathcal{O})x_P + \dfrac{\partial F}{\partial Y}(\mathcal{O})y_P + \dfrac{\partial F}{\partial Z}(\mathcal{O})z_P = 0 \end{cases} \\
&\iff \begin{cases} y_P^2 z_P - (x_P^3 + ax_P z_P^2 + bz_P^3) = 0 \\ (-3X^2 - aZ^2)(\mathcal{O})x_P + (2YZ)(\mathcal{O})y_P + (Y^2 - 2aXZ - 3bZ^2)(\mathcal{O})z_P = 0 \end{cases} \\
&\iff \begin{cases} y_P^2 z_P - (x_P^3 + ax_P z_P^2 + bz_P^3) = 0 \\ z_P = 0 \end{cases} \\
&\iff \begin{cases} -x_P^3 = 0 \\ z_P = 0 \end{cases} \\
&\iff \begin{cases} x_P = 0 \\ z_P = 0 \end{cases} \\
&\iff P = [0, y_P, 0] \quad \text{avec } y_P \neq 0 \qquad\qquad ((x_P, y_P, z_P) \neq (0,0,0)) \\
&\iff P = \mathcal{O}
\end{aligned}$$

On a donc montré que $I = \{\mathcal{O}\}$. Ainsi, en utilisant la proposition 7, on obtient que $\mathcal{O} * \mathcal{O} = \mathcal{O}$.

❑

**Proposition 9 :** *Structure de groupe commutatif*

Soit $E$ une courbe elliptique.
La loi de composition interne $+$ définit une structure de groupe commutatif. L'élément neutre est $\mathcal{O}$. Le symétrique d'un point $M \in E$ est $-M = O * M$

**Démonstration :**

1) Montrons que $\mathcal{O}$ est l'élément neutre pour la loi $+$.

Soit $M \in E$,

$$\mathcal{O} + M = \mathcal{O} * (\mathcal{O} * M) = M$$

en effet, les 3 points d'intersection comptés avec leur multiplicité de la droite $(\mathcal{O}M)$ avec $E$ sont $\mathcal{O}$, $M$ et $(\mathcal{O}*M)$, donc les 3 points d'intersection comptés avec leur multiplicité de la droite $(\mathcal{O}(\mathcal{O}*M))$ avec $E$ sont $\mathcal{O}$, $(\mathcal{O} * M)$ et $M$.

2) Soit $M \in E$, montrons que le symétrique du point $M$ est $\mathcal{O} * M$.

$$\begin{aligned}(\mathcal{O} * M) + M &= \mathcal{O} * ((\mathcal{O} * M) * M) \\ &= \mathcal{O} * \mathcal{O} \\ &= \mathcal{O}\end{aligned}$$

En effet, les 3 points d'intersection comptés avec leur multiplicité de la droite $(\mathcal{O}M)$ avec $E$ sont $\mathcal{O}$, $M$ et $(\mathcal{O} * M)$, donc les 3 points d'intersection comptés leur multiplicité de la droite $((\mathcal{O} * M)M)$ avec $E$ sont $(\mathcal{O} * M)$, $M$ et $\mathcal{O}$.

De même,

$$\begin{aligned}M + (\mathcal{O} * M) &= \mathcal{O} * (M * (\mathcal{O} * M)) \\ &= \mathcal{O} * \mathcal{O} \\ &= \mathcal{O}\end{aligned}$$

3) Associativité : admise

L'associativité est relativement complexe à démontrer. Il est aussi possible de la démontrer par des calculs fastidieux grâce aux formules explicites que nous donnerons par la suite.

4) Commutativité

Soit $(P, Q) \in E^2$

$(P * Q)$ est le troisième point d'intersection de la droite $(PQ)$ avec $E$. Or on a $(PQ) = (QP)$ donc le troisième point d'intersection de la droite $(PQ)$ avec $E$ et aussi celui de $(QP)$ avec $E$. Ainsi $(P * Q) = (Q * P)$. Donc :

$$\begin{aligned}P + Q &= \mathcal{O} * (P * Q) \\ &= \mathcal{O} * (Q * P) \\ &= Q + P\end{aligned}$$

❑

Soit $(A, B, C) \in E^3$. Si $A$, $B$ et $C$ sont alignés, alors $A + B + C = \mathcal{O}$.

**Proposition 10**

Soit $E$ une courbe elliptique. Pour $M = (x, y) \in E$, on a $-M = (x, -y)$.
En particulier, dans le plan réel, la courbe E est symétrique par rapport à l'axe des abscisses.

**Démonstration :**

Soit $E$ une courbe elliptique d'équation $F(X,Y,Z) = Y^2Z - (X^3 + aXZ^2 + bZ^3) = 0$.

Soit $M = [x_M, y_M, 1] \in E$, montrons que le point $M' = [x_M, -y_M, 1] \in E$.

Comme $M \in E$, on a $y_M^2 = x_M^3 + ax_M + b$. On a aussi $(-y_M)^2 = x_M^3 + ax_M + b$ donc $M' \in E$.

Montrons que $M' = -M$, cela revient à montrer que $M' = (\mathcal{O} * M)$ d'après la proposition 9.

Montrons d'abord que $\mathcal{O} \in (MM')$.

1er cas : si $M = M'$

On a alors $y_M = -y_M = 0$ ($\mathbb{K}$ est de caractéristique différente de 2). La droite $(MM')$ correspond à la tangente à la courbe $E$ au point $M$, qui a pour équation :

$$\frac{\partial F}{\partial X}(M)x + \frac{\partial F}{\partial Y}(M)y + \frac{\partial F}{\partial Z}(M)z = 0$$

Or, $\frac{\partial F}{\partial Y}(M) = (2YZ)(M) = 0$.

1er sous-cas : si $\frac{\partial F}{\partial X}(M) = 0$, alors l'équation de la droite $(MM')$ est :

$$\frac{\partial F}{\partial Z}(M)z = 0$$

avec $\frac{\partial F}{\partial Z}(M) \neq 0$ car $M$ est un point non singulier (par définition de la courbe elliptique). Ainsi, on a bien $\mathcal{O} \in (MM')$.

2e sous-cas : si $\frac{\partial F}{\partial X}(M) \neq 0$, alors l'équation de la droite $(MM')$ est :

$$\frac{\partial F}{\partial X}(M)x = -\frac{\frac{\partial F}{\partial Z}(M)}{\frac{\partial F}{\partial X}(M)}z$$

Ainsi $(MM')$ est une droite verticale et $\mathcal{O} \in (MM')$.

2e cas : si $M \neq M'$

Comme les points $M$ et $M'$ sont distincts, d'après la remarque de la proposition 5, la droite $(MM')$ est verticale, donc $\mathcal{O} \in (MM')$.

Conclusion : Dans tous les cas, on a montré que $\mathcal{O} \in (MM')$. Or $\mathcal{O} \in E$, donc $\mathcal{O}$ est le troisième point d'intersection de la droite $(MM')$ avec $E$. On a ainsi $(M * M') = \mathcal{O}$ et $P + Q = \mathcal{O} * (M * M') = \mathcal{O} * \mathcal{O} = \mathcal{O}$. Donc $M' = (\mathcal{O} * M) = -M$.

❑

**Proposition 11**

Soit $A = (x_A, y_A) \in E$ et $B = (x_B, y_B) \in E$.
Si $x_A = x_B$ et $y_A \neq y_B$, alors $A + B = \mathcal{O}$.

**Démonstration :**

La droite $(AB)$ est verticale ($A \neq B$), donc le troisième point d'intersection entre de droite $(AB)$ avec la courbe $E$ est le point à l'infini $\mathcal{O}$. Ainsi $(A*B) = \mathcal{O}$, donc $A+B = \mathcal{O}*(A*B) = \mathcal{O}*\mathcal{O} = \mathcal{O}$.

❑

**Proposition 12 :** *Formules d'addition*

Soit une courbe elliptique $E = \{(x,y) \in \mathbb{K}^2 \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$.
Soient $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux points de E. On suppose que $P$ et $Q$ ne sont pas opposés ($P \neq -Q$).
On a $P + Q = R$ avec $R = (x_R, y_R)$ où :

$$\begin{cases} x_R = \lambda^2 - x_P - x_Q \\ y_R = \lambda(x_P - x_R) - y_P \end{cases} \quad \text{avec} \quad \lambda = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P} & \text{si} \quad P \neq Q \\ \dfrac{3x_P^2 + a}{2y_P} & \text{si} \quad P = Q \end{cases}$$

**Démonstration :**

Montrons par l'absurde que la droite $(PQ)$ n'est pas verticale.

Si la droite $(PQ)$ est verticale, $\mathcal{O} \in (PQ)$. Or $\mathcal{O} \in E$, donc $\mathcal{O}$ est le troisième point d'intersection de la droite $(PQ)$ avec $E$. On a donc $(P * Q) = \mathcal{O}$ et $P + Q = \mathcal{O} * (P * Q) = \mathcal{O} * \mathcal{O} = \mathcal{O}$. Or par hypothèse, $P \neq -Q$, donc c'est absurde.

$(PQ)$ admet donc une équation affine de la forme $y = \lambda x + \nu$.

1er cas : si $P \neq Q$

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P} \quad \text{et} \quad \nu = y_P - \lambda x_P$$

2e cas : si $P = Q$

$x_P = x_Q$ et $y_P = y_Q$

Comme $P \neq -Q$ par hypothèse, $y_P \neq -y_Q$ donc $y_P \neq 0$.

$(PQ)$ est la tangente au point $P$ de la courbe E. $(PQ)$ a pour équation :

$$\frac{\partial F}{\partial X}(P)x + \frac{\partial F}{\partial Y}(P)y + \frac{\partial F}{\partial Z}(P)z = 0$$

Comme $y_P \neq 0$, $\frac{\partial F}{\partial Y}(P) \neq 0$, ainsi $\mathcal{O}$ n'est pas inclus dans la droite. On peut donc passer en coordonnées non homogènes $[x, y, 1]$ pour se ramener dans le plan affine :

$$\frac{\partial F}{\partial X}(x_P, y_P, 1)x + \frac{\partial F}{\partial Y}(x_P, y_P, 1)y + \frac{\partial F}{\partial Z}(x_P, y_P, 1) = 0 - (3x_P^2 + a)x + (2y_P)y + -(y_P^2 - 2ax_P + 3b) \quad = 0$$

$$(2y_P)y = (3x_P^2 + a)x + (y_P^2 - 2ax_P + 3b)$$

$$y = \frac{(3x_P^2 + a)}{2y_P} + \frac{(y_P^2 - 2ax_P + 3b)}{2y_P}$$

On prend donc $\lambda = \frac{3x_P^2+a}{2y_P}$ et toujours $\nu = y_P - \lambda x_P$.

Ainsi, dans tous les cas, pour $M = (x_0, y_0) \in D \cap E$, $M$ vérifie

$$\begin{cases} y_0^2 = x_0^3 + ax_0 + b \\ y_0 = \lambda x_0 + \nu \end{cases}$$

Ainsi $x_0$ est racine du polynôme :

$$P = X^3 + -\lambda^2 X^2 + (a - 2\lambda\nu)X + (b - \nu 2)$$

On connait déjà deux racines de $P$, $x_P$ et $x_Q$, et comme $P$ est de degré 3, il existe une troisième racine $x_{R'} \in E$ telle que

$$\begin{aligned} P &= (X - x_P)(X - x_Q)(X - x_{R'}) \\ P &= X^3 - (x_P + x_Q + x_{R'})X^2 + (x_P x_Q + x_P x_R + x_Q x_R)X - x_P x_Q x_{R'} \end{aligned}$$

Ainsi, on obtient $\lambda^2 = -x_P - x_Q - x_R$, donc $x_R = \lambda^2 - x_P - x_Q$.

Ensuite,

$$\begin{aligned} y_{R'} &= \lambda x_{R'} + \nu \\ y_{R'} &= \lambda x_R + y_P - \lambda x_P \\ y_{R'} &= \lambda(x_R - x_P) + y_P \end{aligned}$$

Le point $R'$ correspond au troisième point d'intersection de la droite $D$ à la courbe $E$, ie $(P * Q)$. Le point $R = P + Q = \mathcal{O} * (P * Q)$ est donc l'opposé du point $R'$, on obtient finalement

$$\begin{cases} x_R = \lambda^2 - x_P - x_Q \\ y_R = \lambda(x_P - x_R) - y_P \end{cases}$$

❑

**Interprétation géométrique dans $\mathbb{R}$ :**

Soit $E$ une courbe elliptique et $(P, Q) \in E^2$.

Pour additionner $P$ et $Q$, on trace la droite passant par $P$ et $Q$ et on obtient un point d'intersection $R$ sur la courbe. La somme $P + Q$ est égale au symétrique de $R$ par rapport à l'axe des abscisses.

Si $P = Q$, alors on utilise la tangente à la courbe au point P.

On retrouve que si $P$ et $Q$ sont symétriques par rapport à l'axe des abscisses, $P + Q = O$ et donc que $Q$ et $P$ sont opposés.

### 1.4.4 Multiplication scalaire

**Définition 19 :** *Multiplication scalaire*

Soit $E$ une courbe elliptique.

L'itération de la loi d'addition permet de définir une loi de composition externe appelée **multiplication** d'un point $P \in E$ par un entier $n \in \mathbb{N}$ :

$$\times : \begin{cases} \mathbb{N} \times E \longrightarrow E \\ (n, P) \longmapsto \begin{cases} \mathcal{O} & \text{si } n = 0 \\ \underbrace{P + \cdots + P}_{n \text{ fois}} & \text{si } n \geq 1 \end{cases} \end{cases}$$

On notera indifféremment $n \times P$ ou plus simplement $nP$ pour $\times(n, P)$.

# 2 Courbes elliptiques sur les corps finis

## 2.1 Préliminaires

**Proposition 13**

Soit $p \in \mathbb{N}^*$.
L'anneau $\mathbb{Z}/p\mathbb{Z}$ est un corps si et seulement si $p$ est un nombre premier. On note ce corps $\mathbb{F}_p$.

Rappelons à présent quelques notions sur les groupes finis et cycliques.

**Définition 20 :** *Groupe fini*

Soit $(G, *)$ un groupe.

$(G, *)$ est un groupe *fini* si l'ensemble $G$ est fini. On définit alors *l'ordre du groupe* $(G, *)$ comme le cardinal de l'ensemble $G$.

**Définition 21 :** *Ordre d'un élément*

Soit $(G, *)$ un groupe. On note $e$ l'élément neutre.

Un élément $a \in G$ est d'*ordre fini* s'il existe $n \in \mathbb{N}^*$ tel que $a^n = e$.

Pour un élément $a \in G$ d'ordre fini, on définit l'*ordre de l'élément* $a$ comme le plus petit $n \in \mathbb{N}^*$ tel que $a^n = e$.

**Proposition 14**

Soit $(G, *)$ un groupe.
Si $(G, *)$ est un groupe fini, alors tous ses éléments sont d'ordre fini.

**Définition 22 :** *Sous-groupe engendré par un élément*

Soit $(G, *)$ un groupe et $a \in G$.

Le *sous-groupe engendré par l'élément* $a$ est le plus petit sous-groupe de $G$ contenant $a$. On le note $\langle a \rangle$.

On a $\langle a \rangle = \left\{ a^k,\ k \in \mathbb{Z} \right\}$.

**Proposition 15**

Soit $(G, *)$ un groupe et $a \in G$.
Si $a$ est d'ordre fini, alors le sous-groupe engendré par $a$ est fini et l'ordre de l'élément $a$ correspond à l'ordre de ce sous-groupe $\langle a \rangle$.

**Définition 23 :** *Groupe cyclique*

Un groupe $(G, *)$ est dit *cyclique* s'il est fini et engendré par un singleton $\{g\}$.

On note alors (abusivement) $G = \langle g \rangle$.

## 2.2 Applications aux courbes elliptiques

Nous nous intéressons maintenant aux courbes elliptiques sur un corps finis. Dans cette étude, nous nous restreignons aux corps premiers finis $\mathbb{F}_p$, avec $p$ un nombre premier.

Pour rappel, nous avions ainsi défini les courbes elliptiques dans un corps $\mathbb{K}$ de caractéristiques différentes de 2 ou 3 :

Une courbe elliptique est l'ensemble défini par :

$$E = \{(x, y) \in \mathbb{K}^2 \ / \ y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

avec $(a, b) \in \mathbb{K}^2$ tel que $\Delta = -16(4a^3 + 27b^2) \neq 0$, et $\mathcal{O}$ le *point à l'infini* de la courbe.

Dans le corps $\mathbb{F}_p$ avec $p > 3$ premier, cette définition devient alors :

**Définition 24 :** *Courbe elliptique dans* $\mathbb{F}_p$

Soit $p > 3$ un nombre premier.

Une courbe elliptique $E$ est l'ensemble défini par :

$$E(\mathbb{F}_p) = \{(x, y) \in (\mathbb{F}_p)^2 \ / \ y^2 \equiv x^3 + ax + b \mod p\} \cup \{\mathcal{O}\}$$

avec $(a, b) \in (\mathbb{F}_p)^2$ tel que $\Delta = -16(4a^3 + 27b^2) \not\equiv 0 \mod p$, et $\mathcal{O}$ le *point à l'infini* de la courbe.

**Remarque :**

$E(\mathbb{F}_p)$ correspond à l'ensemble des points de la courbe $E$. Cette notation permet d'insister sur le fait que nous travaillons dans un corps fini.

**Représntation graphique :**

Il est possible de représenter graphiquement une telle courbe, cependant la représentation n'est plus une courbe finie comme dans $\mathbb{R}$, mais seulement un nuage de points. Prenons l'exemple de la

courbe elliptique d'équation $y^2 = x^3 + 3x + 10$ dans le corps $\mathbb{F}_{97}$ et representons l'ensemble de ses points (figure 8).

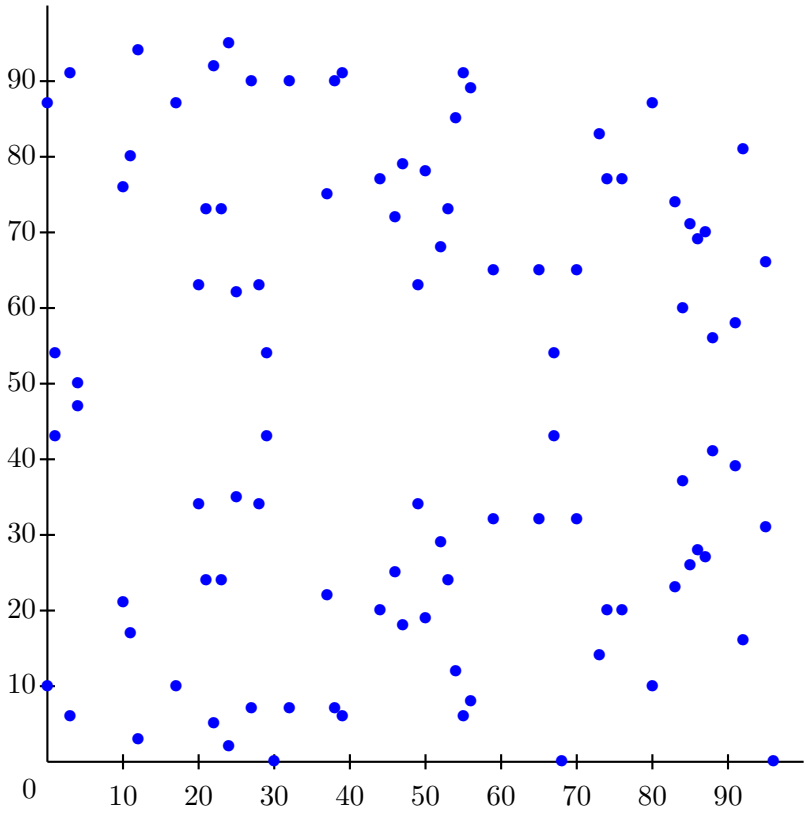



FIGURE 8 – Courbe $y^2 = x^3 + 3x + 10$ dans le corps $\mathbb{F}_{97}$

On remarque que la symétrie pas rapport à l'axe des abscisses n'est plus présente, en revanche, il y a une symétrie par rapport à la droite $y = \frac{p}{2}$.

Toutes les propriétés que nous avions données sont encore vérifiées, notamment, la loi d'addition permet toujours de construire un **groupe commutatif**, cependant les calculs se font maintenant modulo $p$.

L'interprétation graphique est plus compliquée à cause de la notion de droite. Dans $\mathbb{R}^2$, une droite est l'ensemble des points $(x, y) \in \mathbb{R}^2$ vérifiant une équation du type $ax + by + c = 0$. Dans $(\mathbb{F}_p)^2$, nous pouvons assimiler une droite à l'ensemble des points $(x, y) \in (\mathbb{F}_p)^2$ vérifiant une équation du type $ax + by + c \equiv 0 \mod p$.

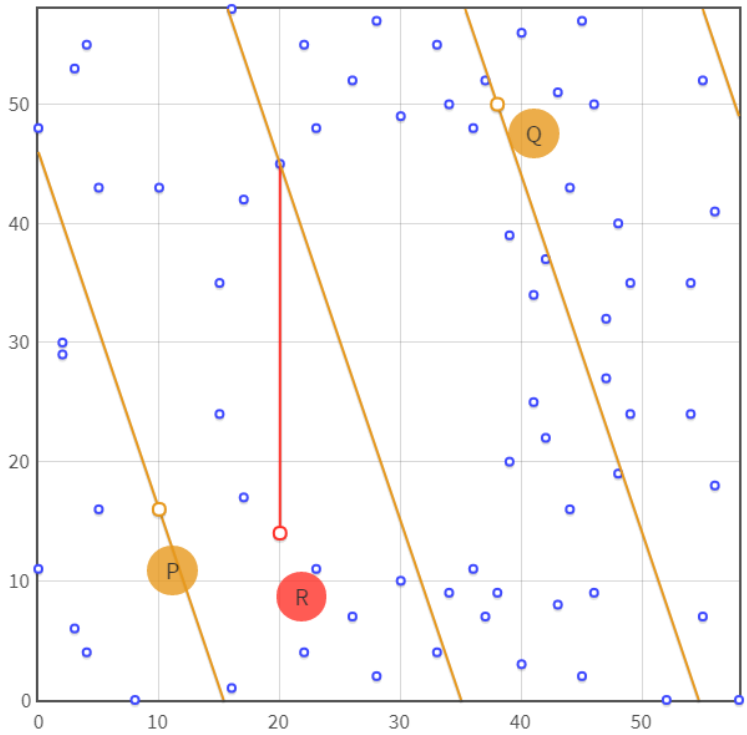



FIGURE 9 – Addition des points $P$ et $Q$ sur la courbe $y^2 = x^3 + 3x + 10$ dans le corps $\mathbb{F}_{59}$
Créé avec Elliptic Curve point addition $(\mathbb{F}_p)$ [2]

**Proposition 16**

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_p$.
Le groupe $(E(\mathbb{F}_p), +)$ est un groupe commutatif *fini*.

**Démonstration :**

$E(\mathbb{F}_p)$ est clairement un ensemble fini en tant que partie de $(\mathbb{F}_p)^2$ qui est fini.

Nous savons déjà que $(E(\mathbb{F}_p), +)$ est un groupe commutatif d'après la proposition 9.

❑

Nous pouvons maintenant nous intéresser au groupe $(E(\mathbb{F}_p), +)$. Comme ce groupe est fini, tout élément de $G$ va engendrer un sous-groupe qui sera **cyclique**. Reprenons l'exemple de la courbe d'équation $y^2 = x^3 + 3x + 10$ définie dans le corps $\mathbb{F}_{97}$. On considère alors le point de la courbe $H = (29, 43)$. Nous pouvons calculer les différents multiples de $H$ et on s'aperçoit que $10H = \mathcal{O}$ puis $11H = H$. L'ordre du point $H$ est donc 10 et le cardinal du sous-groupe $\langle H \rangle$ vaut 10.

2. `https://cdn.rawgit.com/andreacorbellini/ecc/920b29a/interactive/modk-add.html`

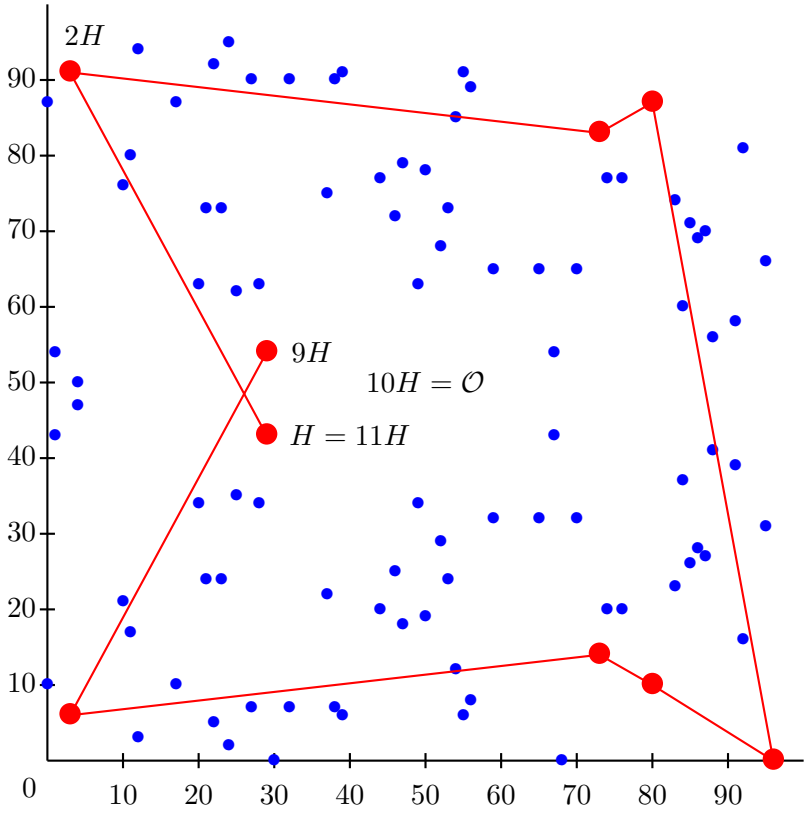



FIGURE 10 – Itération de l'addition sur le point $H$

En cryptographie, nous nous placerons dans le cadre de ces sous-groupes cycliques.

# 3 Implémentation sur Python

Nous proposons ici une implémentation des courbes elliptiques sur Python [3] permettant de manipuler les opérations présentées précédemment.

## 3.1 Première approche

On implémente tout d'abord les courbes elliptiques dans le corps des réels.

On crée une classe pour les courbes.

```
class Courbe(object):
    def __init__(self,a,b):
        """Initialisation"""
        self.a = a
        self.b = b

        self.discriminant = -16 * (4*a**3 + 27*b**2)

        if self.estsinguliere():
            raise Exception("La courbe est singulière")

```

3. Python 3.6.5

```
12      def possede_point(self, x, y):
13          """Renvoie True si le point est sur la courbe"""
14          return (y**2) == (x**3 + self.a * x + self.b)
15
16      def estsinguliere(self):
17          """Renvoie True si la courbe est non singulière"""
18          return self.discriminant == 0
19
20      def __eq__(self, autre):
21          """Egalité"""
22          return(self.a, self.b) == (autre.a, autre.b)
23
24      def __repr__(self):
25          """Représentation"""
26          return "y^2 = x^3 + {}x + {}".format(self.a, self.b)
```

Comme on ne travaille pas en coordonnées projectives, mais dans le plan, on ne peut pas définir le point à l'infini avec 2 coordonnées. On crée donc une classe spécifique.

```
1   class Infini(object):
2       def __init__(self,courbe):
3           """Initilisation"""
4           self.courbe = courbe
5
6       def __repr__(self):
7           return "Infini"
8
9       def __eq__(self, autre):
10          """Egalité"""
11          return isinstance(autre, Infini)
12
13      def __neg__(self):
14          """Point opposé"""
15          return self
16
17      def __add__(self, autre):
18          """Addition"""
19          return autre
```

On peut ensuite définir une classe pour les points. On choisit ici de définir un point par rapport à une courbe car on ne travaille en général que sur une seule courbe. L'addition est définie selon les formules démontrées précédemment. La multiplication, qui correspond à l'itération de l'addition, est implémenté avec l'algorithme d'exponentiation rapide [4].

```
1   class Point(object):
2       def __init__(self,courbe, x,y):
3           """Initialisation"""
4           self.courbe = courbe
5           self.x = x
6           self.y = y
7
8           if not self.courbe.possede_point(x,y):
9               raise ValueError("Le point {} n'est pas sur la courbe
                ↪  {}".format(self, self.courbe))
```

4. L'addition est bien associative.

```
10
11      def __repr__(self):
12          """Représentation"""
13          return "({},{})".format(self.x, self.y)
14
15      def __eq__(self, autre):
16          """Egalité"""
17          return (self.courbe, self.x, self.y) == (autre.courbe, autre.x,
            ↪  autre.y)
18
19      def __neg__(self):
20          """Point opposé"""
21          return Point(self.courbe, self.x, - self.y)
22
23      def __add__(self,autre):
24          """Addition"""
25          assert self.courbe == autre.courbe
26
27          if isinstance(autre, Infini): #Point infini
28              return self
29
30          xs, ys, xa, ya = self.x, self.y, autre.x, autre.y
31
32          if xs == xa:
33              if ys != ya :   #Points opposés:
34                  return Infini(self.courbe)
35
36              if ys == 0:     #ya = 0 aussi
37                  return Infini(self.courbe)
38              else:   #Méthode tangente
39                  Lambda = (3* xs**2 + self.courbe.a)/(2 * ys)
40
41          else :  #Cas où xs != xa
42              Lambda = (ya - ys) / (xa - xs)
43
44          x = Lambda**2 - xs - xa
45          y = Lambda*(xs -x ) - ys
46
47          return Point(self.courbe, x, y)
48
49      def __mul__(self,n):
50          """Multiplication par un entier"""
51          if not isinstance(n, int):
52              raise Exception("La multiplication requiert un entier")
53          else:
54              if n < 0:
55                  return -self * -n
56              if n == 0:
57                  return Infini(self.courbe)
58              else:
59                  result = Infini(self.courbe)
60                  q = n
61                  puiss = self
62                  while q != 0:
63                      if q % 2 == 1:   # si q impair
64                          result = result + puiss
65                      puiss = puiss + puiss
```

```
66                      q = q // 2
67                  return result
```

Vérifions l'implémentation sur quelques points :

```
>>> C = Courbe(-1,1)
>>> C
y^2 = x^3 + -1x  1
>>> P = Point(C,-1,-1)
>>> Q = Point(C,0,1)
>>> P+Q
(5.0,-11.0)
>>> O = Infini(C)
>>> P + O
(-1,-1)
>>> P+P
(3.0,5.0)
>>> P = Point(C,-1,-1)
>>> Q = Point(C,-1,1)
>>> P+Q
Infini
```

Cependant, les limites de la représentation des flottants provoquent assez vite des problèmes.

```
>>> C = Courbe(-1,1)
>>> P = Point(C,-1,-1)
>>> P*2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 65, in __mul__
  File "<stdin>", line 47, in __add__
  File "<stdin>", line 9, in __init__
ValueError: Le point (0.7600000000000007,0.8239999999999981) n'est pas
↪  sur la courbe y^2 = x^3 + -1x + 1
>>> P+P+P+P
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 47, in __add__
  File "<stdin>", line 9, in __init__
ValueError: Le point (0.76,0.824) n'est pas sur la courbe y^2 = x^3 + -1x
↪  + 1
```

Dans le calcul de `P*2`, on remarque que le problème provient du calcul de `P*4` qui a lieu dans l'exponentiation rapide.

Une première solution serait de remplacer, dans le test d'appartenance d'un point à une courbe, l'égalité par une comparaison avec une valeur absolue. Une autre solution serait d'utiliser les fractions qui sont déjà implémentées dans Python.

Finalement, en cryptographie les courbes elliptiques sont généralement utilisées sur des corps finis.

## 3.2 Implémentation sur corps fini

Nous reprenons la même implémentation que précédemment en effectuant la plupart des calculs modulo $p$.

Pour l'addition, nous avons besoin de déterminer l'inverse modulaire, nous utilisons pour cela l'algorithme d'Euclide étendu.

```
    """
    Inverse modulaire
    """
    if p == 0:
        raise Exception("p=0")

    c, d = a, p  # Copies

    u1, v1 = 1, 0
    u2, v2 = 0, 1
    r1, r2 = c, d

    while r2 != 0:
        q = r1 // r2
        u = u1
        v = v1
        u1, v1 = u2, v2
        u2, v2 = u - q * u2, v - q * v2
        r1, r2 = r2, r1 - q * r2

    assert r1 == 1
    assert (a * u1) % p == 1

```

# 4 Cryptographie basée sur les groupes

## 4.1 Théorie

### 4.1.1 Le problème du logarithme discret

On se place dans un groupe fini $(E, \times)$ d'ordre p.

Soit $g$ un élément de $E$, on considère $G$ le sous-groupe engendré par $g$. Le groupe $G$ est un groupe cyclique fini, et on note $n$ l'ordre de $g$.

**Proposition 17**

On considère l'application suivante :

$$\phi : \begin{cases} [\![0\,,\,n-1]\!] & \longrightarrow & G \\ x & \longmapsto & g^x \end{cases}$$

L'application $\phi$ est bijective.

**Démonstration :**

Montrons que $\phi$ est injective.

Soit $(a, b) \in [\![0\,,\, n-1]\!]^2$ tel que $\phi(a) = \phi(b)$. En échangeant $a$ et $b$ si nécessaire, on se ramène à $a \leq b$.

On a $g^a = g^b$, ainsi $g^{b-a} = \mathbb{1}$.

Comme $0 \leq b - a \leq n - 1$ et que $n$ est l'ordre de $g$, $b - a = 0$ ainsi $a = b$.

L'application $\phi$ est injective entre 2 ensembles finis de même cardinal $n$, donc $\phi$ est bijective.

❑

**Définition 25 :** *Problème du logarithme discret*

Soient $(G, \times)$ un groupe cyclique fini engendré par $g \in G$ et $n \in \mathbb{N}$ l'ordre de $g$.

Le *problème du logarithme discret* (DLP) pour $q \in G$ consiste à déterminer l'unique entier $k \in [\![0\,,\, n-1]\!]$ tel que $q = g^k$.

Le logarithme discret correspond à la bijection réciproque de l'application $\phi$. On le note $\log_g$.

**Remarque :**

Nous définissons ici le logarithme discret dans un groupe multiplicatif, mais nous pouvons le faire indifféremment dans un groupe additif ou multiplicatif.

### 4.1.2 Échange de clés de Diffie-Hellman

L'échange de clés de Diffie-Hellman fut l'un des premiers protocoles à clé publique. Il permet de partager une clé commune entre 2 parties de manière sécurisée. Cette clé pourra ensuite être utilisée dans le cadre d'un chiffrement.

**Protocole :**

- Alice et Bob choisissent publiquement un groupe fini cyclique $(G, *)$ et un générateur $g$ de ce groupe. On note $n$ l'ordre de $g$.
- Alice choisit sa clé privée $n_A \in \mathbb{N}^*$ et crée sa clé publique $P_A = g^{n_A}$ qu'elle envoie à Bob.
- Bob choisit sa clé privée $n_B \in \mathbb{N}^*$ et crée sa clé publique $P_B = g^{n_B}$ qu'il envoie à Alice.
- Alice calcule $(P_B)^{n_A}$ et Bob calcule $(P_A)^{n_B}$.
- Alice et Bob disposent maintenant d'une clé secrète $K = (g^{n_A})^{n_B} = (g^{n_B})^{n_A} = g^{n_A \times n_B}$.

La confidentialité de la clé secrète repose sur le problème du logarithme discret. En effet, si un attaquant connait $P_A$ et $P_B$, pour pouvoir calculer $K = g^{n_A \times n_B}$, il doit calculer $n_A$ et $n_B$ connaissant $g^{n_A}$ et $g^{n_B}$ [5].

5. En réalité, le fait de déterminer $g^{xy}$ connaissant $g^x$ et $g^y$ est appelé le *problème de Diffie-Hellman* (DHP). La résolution du problème du logarithme discret (DLP) implique la résolution du problème de Diffie-Hellman, mais pas l'inverse. Cependant, on suppose actuellement que le seul moyen de résoudre le DHP est de résoudre le DLP, on suppose donc que la sécurité de l'échange de clés de Diffie-Hellman repose sur celle du DLP.

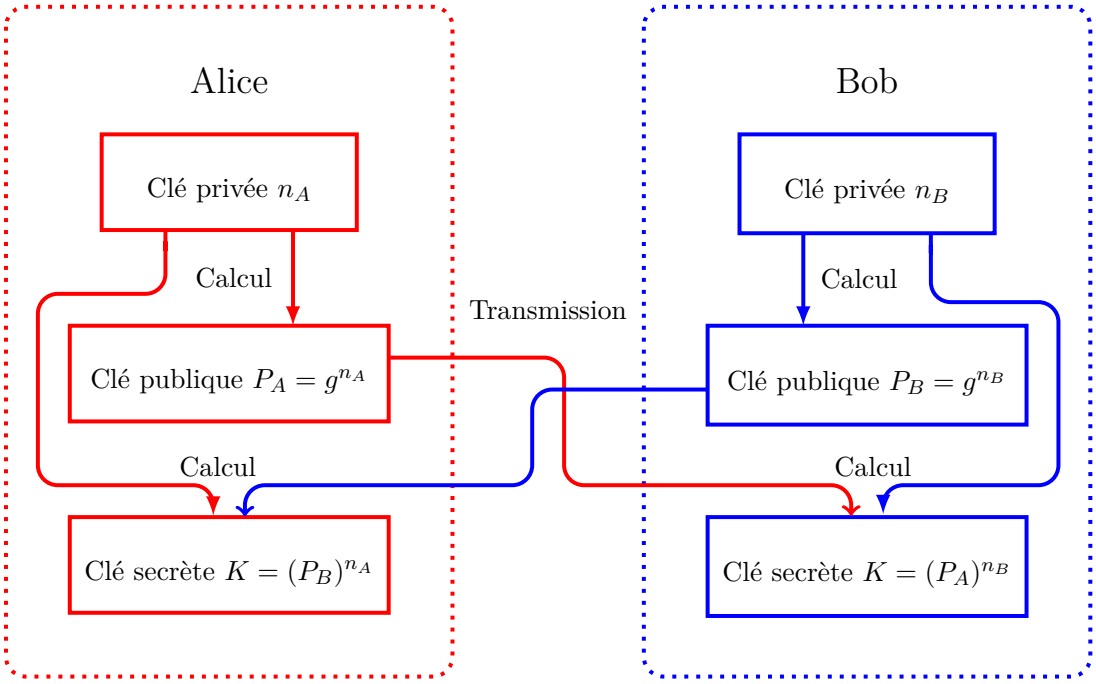



FIGURE 11 – Protocole de DIFFIE-HELLMAN

### 4.1.3 Cryptosystème de ElGamal

Le cryptosystème de ELGAMAL permet de transmettre un message crypté de manière asymétrique. On suppose qu'Alice souhaite envoyer un message secret à Bob.

**Protocole :**

- Alice et Bob choisissent publiquement un groupe fini cyclique $(G, *)$ et un générateur $g$ de ce groupe. On note $n$ l'ordre de $g$.
- Bob choisit sa clé privée $y \in \mathbb{N}^*$ et crée sa clé publique $Y = g^y$ qu'il envoie à Alice.
- Alice choisit un aléa $k \in \mathbb{N}^*$ et calcule $A = g^k$. A partir d'un message $m \in G$ et de la clé publique $Y$ de Bob, Alice calcule $P = Y^k$ puis $B = m * P = m * Y^k$, et envoie à Bob le couple $C = (A, B)$ qui correspond au message chiffré.
- Après avoir reçu le couple $C$, Bob calcule $P' = A^y$ puis $m' = B * P'^{-1}$. Bob dispose alors du message $m$, en effet, $P' = (g^k)^y = (g^y)^k = Y^k = P$, puis $m' = B * P^{-1} = m$.

La confidentialité repose là encore sur le problème du logarithme discret. En effet si un attaquant connait $Y$ qui est la clé publique de Bob et le couple $C = (A, B)$, pour calculer $m$, il doit connaitre $P = Y^k$, et donc $k$, en connaissant $A = g^k$ et $g$.

Il est aussi important qu'Alice change son aléa $k$ à chaque transmission d'un message. En effet, si pour deux messages $m_1$ et $m_2$, Alice utilise le même $k$, alors $A_1 = A_2$, et $B_1 * B_2^{-1} = (m_1 * Y^k) * (m_2 * Y^k)^{-1} = m_1 * m_2^{-1}$. Il suffit alors de connaitre le 1er message pour calculer le 2e.

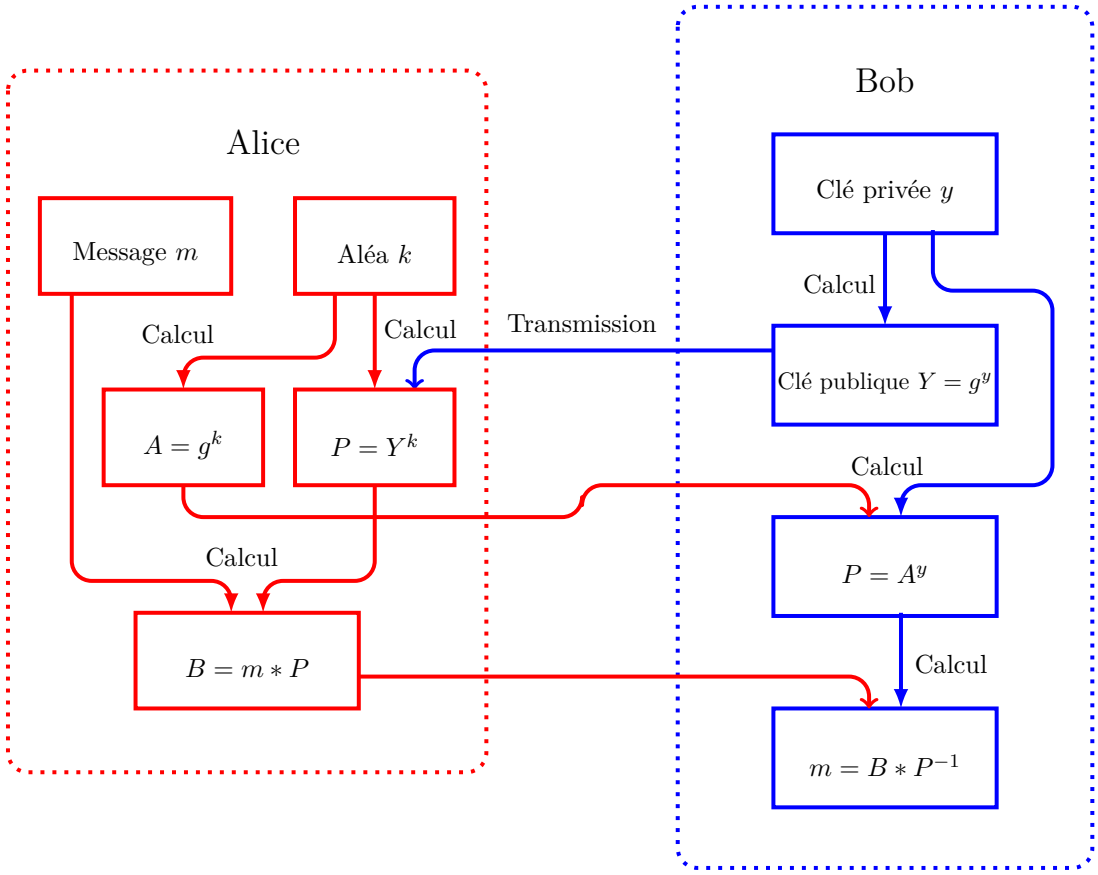



FIGURE 12 – Cryptosystème de ELGAMAL

### 4.1.4 Signature de ElGamal

La signature électronique est un mécanisme permettant de vérifier la conformité d'un message et d'authentifier l'identité de son auteur. Les signatures électroniques s'appuient en général sur la cryptographie asymétrique et utilisent des fonctions de hachage.

**Définition 26 :** *Fonction de hachage*

On considère un ensemble de données $D$.

Une *fonction de hachage* $H$ est une fonction qui à une donnée $x$ de taille quelconque, calcule une empreinte $H(x)$ d'une taille fixée. L'empreinte est aussi appelée *haché* (*hash* en anglais), *condensat* ou *signature*.

L'empreinte d'un élément est généralement beaucoup plus petite que l'élément lui-même.

**Définition 27 :** *Collision*

Pour une fonction de hachage $H$, une collision est un couple $(x, x') \in D^2$ tel que

$$x \neq x' \quad \text{et} \quad H(x) = H(x').$$

**Remarque :**

Un des intérêts d'une fonction de hachage est de pouvoir identifier une donnée à partir de son empreinte, les collisions sont donc considérées comme indésirables. Cependant, elles sont inévitables, en effet comme les empreintes ont une taille fixée, l'ensemble des empreintes est fini, alors que l'ensemble de départ d'une fonction de hachage est infini, il ne peut donc pas exister d'injection entre ces deux

ensembles.

Pour une utilisation cryptographique, une fonction de hachage $H$ doit satisfaire les propriétés suivantes :

- Pour une données $x \in D$, il est facile de calculer $H(x)$
- **Résistance au calcul de préimage :** Pour une empreinte $y$, il est difficile de trouver $x \in D$ tel que $H(x) = y$.
- **Résistance au calcul de seconde préimage :** Étant donné $x \in D$, il est difficile de trouver $x' \in D$ tel que $x \neq x'$ et $H(x) = H(x')$.
- **Résistance aux collisions :** Il est difficile de trouver $(x, x') \in D^2$ tel que $x \neq x'$ et $H(x) = H(x')$.

**Exemples :** Des fonctions de hachage couramment utilisées sont `MD5`, `SHA-1` et `SHA-2`.

La signature de ElGamal se décompose en trois procédures : la création des clés, la signature d'un message et la vérification. On suppose qu'Alice souhaite signer un message qu'elle envoie à Bob.

On considère une fonction de hachage $H$, un groupe fini cyclique $(G, *)$ et un générateur $g$ de ce groupe. On note $n$ l'ordre de $g$.

**Protocole :**

Création des clés :

- Alice choisit sa clé privée $n_A \in \mathbb{N}^*$ et crée sa clé publique $P_A = g^{n_A}$.

Signature du message :

- Alice choisit un entier $k \in [\![1\,,\, n-1]\!]$ tel que $k \wedge n = 1$, puis calcule $R = g^k$.
- A partir d'un message $m \in N$, Alice calcule $s = k^{-1}[h(m) - n_A h(R)] \mod n$.
- Alice envoie le triplet $(m, s, R)$ qui correspond au message signé.

Vérification de la signature :

- Bob calcule $H(m)$ et $H(R)$.
- Bob calcule $V_1 = (P_A)^{h(R)} * R^s$ et $V_2 = g^{h(m)}$.
- La signature est authentique si et seulement si $V_1 = V_2$.

**Remarque :**

On considère ici que la fonction de hachage $H$ calcule à partir d'un nombre entier ou d'un élément quelconque du groupe $G$ une empreinte qui est un nombre entier. Dans d'autre versions du protocole, on peut considérer que $H : \mathbb{N} \longrightarrow \mathbb{N}$ et il faut alors prendre une fonction $f : G \longrightarrow \mathbb{N}$. On calcule alors $f(R)$ au lieu de $H(R)$.

Démontrons le dernier point de la vérification de la signature :

$(\Rightarrow)$ Montrons que si la signature est valide, alors $V_1 = V_2$. Comme on suppose que la signature est valide, on a bien $s = k^{-1}[h(m) - n_A h(R)] \mod n$ et $R = g^k$. Ainsi,

$$\begin{aligned} V_1 &= (P_A)^{h(R)} * R^s \\ &= g^{n_A h(R)} * g^{k \times k^{-1}[h(m) - n_A h(R)]} \\ &= g^{n_A h(R)} * g^{-n_A h(R)} * g^{h(m)} \end{aligned}$$

$$= g^{h(m)}$$
$$= V_2$$

(⇐) En théorie, l'égalité $V_1 = V_2$ n'implique pas que la signature soit valide. Cependant, le problème du logarithme discret empêche de calculer $n_A$ à partir de $P_A$ et $k$ à partir de $R$. De plus, il est très difficile de trouver un autre message $m'$ tel que $H(m) = H(m')$ comme on considère que la fonction de hachage est résistante aux collisions.

## 4.2 Applications aux courbes elliptiques

### 4.2.1 Paramètres des courbes

En cryptographie, comme nous le verrons plus tard, certaines courbes elliptiques permettent d'obtenir une meilleure sécurité que d'autres. Le choix d'une courbe elliptique et d'un sous-groupe est donc important. La diffusion de ces paramètres est normalisée pour faciliter le partage et la réutilisation.

Nous parlerons abusivement de « courbe elliptique » pour désigner une courbe elliptique, le corps fini sur lequel elle est définie, et le sous-groupe cyclique utilisé.

Les paramètres d'une courbe correspondent à un sextuplet $(p, a, b, G, n, h)$ avec

- Le nombre premier $p$ qui définit le corps fini $\mathbb{F}_p$ ;
- Les coefficients $a$ et $b$ qui définissent la courbe elliptique d'équation $y^2 = x^3 + ax + b$ ;
- Le point $G$ générateur d'un sous-groupe cyclique ;
- L'ordre $n$ de ce sous-groupe ;
- Le cofacteur $h$ (rapport du cardinal de la courbe sur l'ordre $n$).

Pour des applications cryptographiques, il est nécessaire de trouver des courbes elliptiques avec un cardinal important, puis de déterminer un sous-groupe avec un ordre le plus grand possible (l'idéal étant de trouver un sous-groupe d'ordre égal au cardinal de la courbe, le cofacteur valant alors 1). Toutes ces contraintes demandent de très nombreux calculs réalisés par des agences qui publient des courbes utilisables en cryptographie. Deux exemples de courbes couramment utilisées sont la courbe `P-256` (figure 13) recommandée par la NSA [6] dans son standard Suite B, et la courbe `secp256k1` (figure 14) utilisée par la cryptomonnaie Bitcoin.

$p = $ `ffffffff 00000001 00000000 00000000 00000000 ffffffff ffffffff ffffffff`
$a = -3$
$b = $ `5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b`
$G_x = $ `6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296`
$G_y = $ `4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5`
$n = $ `ffffffff 00000000 ffffffff ffffffff bce6faad a7179e84 f3b9cac2 fc632551`
$h = $ `1`

FIGURE 13 – Courbe `P-256`

6. La *National Security Agency* est une agence de renseignement américaine spécialisée en cryptographie.

$$
\begin{aligned}
p &= \texttt{ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff fffffffe fffffc2f} \\
a &= \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000} \\
b &= \texttt{00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007} \\
G_x &= \texttt{79be667e f9dcbbac 55a06295 ce870b07 029bfcdb 2dce28d9 59f2815b 16f81798} \\
G_y &= \texttt{483ada77 26a3c465 5da4fbfc 0e1108a8 fd17b448 a6855419 9c47d08f fb10d4b8} \\
n &= \texttt{ffffffff ffffffff ffffffff fffffffe baaedce6 af48a03b bfd25e8c d0364141} \\
h &= \texttt{1}
\end{aligned}
$$

FIGURE 14 – Courbe `secp256k1`

### 4.2.2 Échange de clés ECDH (Elliptic Curve Diffie-Hellman)

Nous appliquons l'échange de clés de DIFFIE-HELLMAN sur les courbes elliptiques.

On considère une courbe $E$ définie sur un corps fini et un point $G$ de $E$. On se place alors dans le sous-groupe engendré par $g$ qui est un groupe fini cyclique (pour la loi d'addition).

**Protocole :**

- Alice et Bob choisissent publiquement une courbe elliptique $E$ définie sur un corps fini $\mathbb{F}_p$ et un point $G \in E$.
- Alice choisit sa clé privée $n_A \in \mathbb{N}^*$ et crée sa clé publique $P_A = n_A \times G$ qu'elle envoie à Bob.
- Bob choisit sa clé privée $n_B \in \mathbb{N}^*$ et crée sa clé publique $P_B = n_B \times G$ qu'il envoie à Alice.
- Alice calcule $n_A \times P_B$ et Bob calcule $n_B \times P_A$.
- Alice et Bob disposent maintenant d'une clé secrète $K = n_B \times (n_A \times G) = n_A \times (n_B \times G) = (n_A \times_{\mathbb{N}} n_B) \times G$.

Implémentons à présent cet algorithme.

Alice et Bob doivent tout d'abord choisir une clé privée. Nous allons les générer aléatoirement grâce à la fonction `os.urandom()` qui renvoie une suite "pseudo" aléatoire d'octets. Cette fonction est préférable au module `Random` en cryptographie car elle utilise une meilleure source aléatoire.

```
    """Renvoie une clé aléatoire de c bits sous forme hexadecimal"""
    a = os.urandom(c // 8)
    return (int.from_bytes(a, byteorder='big'))

```

Nous transcrivons ensuite directement l'algorithme.

```
def simulation(p, a, b, g, c):
    E = Courbe(a, b, p)
    G = Point(E, g[0], g[1])

    # Alice
    nA = générateur(c)      # Alice génère sa clé privée
    PA = nA * G             # Alice calcule sa clé publique

```

```
    # Bob
    nB = générateur(c)      # Bob génère sa clé privée
    PB = nB * G             # Bob calcule sa clé publique

    # Alice
    KA = nA * PB            # Alice calcule la clé secrète à partir de PB

    # Bob
    KB = nB * PA            # Bob calcule la clé secrète à partir de PA

    print("Clé privée d'Alice:", hex(nA))
    print("Clé publique d'Alice:", (hex(PA.x), hex(PA.y)))
    print("Clé privée de Bob:", hex(nB))
    print("Clé publique de Bob:", (hex(PB.x), hex(PB.y)))
    print("Clé secrète d'Alice:", (hex(KA.x), hex(KA.y)))
    print("Clé secrète de Bob:", (hex(KB.x), hex(KB.y)))
```

Voici un test effectué en utilisant la courbe `P-256` et une longueur de clé de 256 bits :

```
>>> simulation(p,a,b,G,256)
Clé privée d'Alice:
 ↪ 0x2ce59eaf0bba88cd914888c92f7ce34551afc369b43a271d3a02760a087d04df
Clé publique d'Alice:
 ↪ ('0xc6d267e45e0ebed603edeea59490f074e9a7fad8bb04c6b28cacfa688b0075d6',
 ↪ '0x2351adf01021be7bd2149644dc4940602047e69560a5c076a9d985b71a077b6d')
Clé privée de Bob:
 ↪ 0x5ca2d5f5631fae4e451d1566ddb02f88774ede99b780f1e3328325b7e72b43c6
Clé publique de Bob:
 ↪ ('0x2b4fc6b7bf8f71154ed72930b8162a29b29303a75b837ed7416b824ea1798c7a',
 ↪ '0xfa398abfd0b604562615d23d0c70506f781dccc6875575c376e84653ecea35ed')
Clé secrète d'Alice:
 ↪ ('0xade65630a03814a1439fabe32b86e844ac380abe1f916a533eb8f95a938a8970',
 ↪ '0x6806fb3fc15dfdd0395457731103d85f8c5a4fc36000e6c6109be41e14198dad')
Clé secrète de Bob:
 ↪ ('0xade65630a03814a1439fabe32b86e844ac380abe1f916a533eb8f95a938a8970',
 ↪ '0x6806fb3fc15dfdd0395457731103d85f8c5a4fc36000e6c6109be41e14198dad')
```

Alice et Bob possèdent bien la même clé secrète !

### 4.2.3 Chiffrement de ElGamal

Appliquons le chiffrement de ElGamal sur les courbes elliptiques en supposant qu'Alice souhaite envoyer un message à Bob.

Comme pour l'échange de clés ECDH, on considère une courbe $E$ définie sur un corps fini et un point $G$ de $E$. On se place alors dans le sous-groupe engendré par $G$ qui est un groupe fini cyclique (pour la loi d'addition). Le message $m$ à envoyer sera un point quelconque de la courbe $E$.

**Protocole :**

- Alice et Bob choisissent publiquement une courbe elliptique $E$ définie sur un corps fini $\mathbb{F}_p$ et un point $G \in E$.
- Bob choisit sa clé privée $y \in \mathbb{N}^*$ et crée sa clé publique $Y = y \times G$ qu'il envoie à Alice.
- Alice choisit un aléa $k \in \mathbb{N}^*$ et calcule $A = k \times G$. A partir d'un message $m \in E$ et de la clé publique $Y$ de Bob, Alice calcule $P = k \times Y$ puis $B = m + P = m + kY$, et envoie le couple

$C = (A, B)$ qui correspond au message chiffré à Bob.

- Après avoir reçu le couple $C$, Bob calcule $P' = y \times A$ puis $m' = B + (-P')$. Bob dispose alors du message $m$, en effet, $P' = y \times (k \times G) = k \times (y \times G) = k \times Y = P$, puis $m' = B + (-P) = m$.

Dans ce protocole, nous avons besoin de calculer l'opposé d'un point. D'après la proposition 10 appliquée dans un corps fini $\mathbb{F}_p$, l'opposé d'un point $P = (x, y)$ d'une courbe elliptique est le point $-P = (x, -y \mod p)$.

Implémentons à présent cet algorithme.

Pour générer aléatoirement la clé secrète $y$ de Bob et l'aléa $k$, nous réutilisons la fonction `générateur(c)` définie précédemment.

```
    """Renvoie une clé aléatoire de c bits sous forme hexadecimal"""
    a = os.urandom(c // 8)
    return (int.from_bytes(a, byteorder='big'))

```

Nous transcrivons ensuite directement l'algorithme.

```
def simulation(p, a, b, g, c, m):
    E = Courbe(a, b, p)
    G = Point(E, g[0], g[1])
    m = 3 * G

    # Bob
    y = 100                 # Bob génère sa clé privée
    Y = y * G               # Bob calcule sa clé publique

    # Alice
    k = 50                     # Alice génère un aléa k
    A = k * G               # Alice calcule A
    B = m + k * Y           # ALice calcule B à partir de Y

    # Bob
    Pb = y * A              # Bob calcule P' à partir de A
    mb = B + (-Pb)          # Bob calcule m' à partir de B

    print("Message envoyé par Alice:", (hex(m.x), hex(m.y)))
    print("Message déchiffré par Bob:", (hex(mb.x), hex(mb.y)))
```

Voici un test effectué en utilisant la courbe `P-256`, une longueur de clé de 256 bits, et un point de la courbe comme message :

```
>>> simulation(p,a,b,g,256,m)
Message envoyé par Alice:
 ↪  ('0x7cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978',
 ↪  '0x7775510db8ed040293d9ac69f7430dbba7dade63ce982299e04b79d227873d1')
Message déchiffré par Bob:
 ↪  ('0x7cf27b188d034f7e8a52380304b51ac3c08969e277f21b35a60b48fc47669978',
 ↪  '0x7775510db8ed040293d9ac69f7430dbba7dade63ce982299e04b79d227873d1')
```

Bob retrouve bien le message qui lui avait envoyé Alice !

### 4.2.4 Signature ECDSA (Elliptic Curve Digital Signature Algorithm)

La signature ECDSA est un algorithme de signature qui s'inspire de la signature de ElGamal. Elle est notamment utilisée par le réseau Bitcoin pour signer et authentifier toutes ses transactions.

On considère une courbe elliptique $E$ définie sur un corps fini $\mathbb{F}_p$ et un point $G$ de $E$. On se place alors dans le sous-groupe engendré par $G$ et on note $n$ l'ordre de $G$. Il est nécessaire que $n$ soit **premier** [7]. On considère aussi une fonction de hachage $H$ et $m$ le message à signer.

**Protocole :**

Création des clés :

- Alice choisit sa clé privée $n_A \in [\![1\,,\, n-1]\!]$ et calcule sa clé publique $P_A = n_A \times G$.

Signature du message :

- Alice choisit aléatoirement un entier $k \in [\![1\,,\, n-1]\!]$, puis calcule $k \times G = (x, y)$.
- Alice calcule $r = x \mod n$. Si $r = 0$, Alice recommence en choisissant un autre entier $k$.
- Alice calcule $s = k^{-1}(H(m) + rn_A) \mod n$. Si $s = 0$, Alice recommence la signature.
- La signature du message $m$ est alors le couple $(r, s)$.

Vérification de la signature :

- Bob vérifie que $P_A \neq \mathcal{O}$ et que $P_A$ appartient bien à la courbe $E$.
- Bob vérifie que $n \times P_A = \mathcal{O}$.
- Bob vérifie que $(r, s) \in [\![1\,,\, n-1]\!]^2$.
- Bob calcule $u_1 = H(m)s^{-1} \mod n$ et $u_2 = rs^{-1} \mod n$.
- Bob calcule le point $C = (x_C, y_C) = u_1 \times G + u_2 \times P_A$.
- La signature est valide si et seulement si $r = x_C \mod n$.

**Remarque :**

Pour que la signature fonctionne, il est nécessaire que la longueur binaire de $H(m)$ soit inférieur à la longueur binaire de $n$. Comme le recommande le standard FIPS 186-4 du NIST [8], on tronque $H(m)$ en ne conservant que les bits à gauche nécessaires.

Nous pouvons montrer facilement que si la signature est valide, alors $r = x_C \mod n$, en montrant que le point $C$ calculé par Bob correspond au point $k \times G$.

$$\begin{aligned} C &= u_1 \times G + u_2 \times P_A \\ &= u_1 \times G + u_2 \times (n_A \times G) \\ &= (u_1 + u_2 n_A) \times G \\ &= (H(m)s^{-1} + rn_A s^{-1}) \times G \\ &= (H(m) + rn_A)s^{-1} \times G \\ &= (H(m) + rn_A)k^{-1}(H(m) + rn_A)^{-1} \times G \\ &= k \times G \end{aligned}$$

Cet algorithme repose aussi sur le problème du logarithme discret, en effet, si on arrive à calculer $n_A$ à partir de $P_A = n_A \times G$, alors on peut facilement créer une signature valide pour un autre message.

7. Nous utilisons l'inverse modulaire dans l'algorithme.

8. Le *National Institute of Standards and Technology* est une agence américaine du département du Commerce qui publie des normes sur des technologies.

De plus, il est très important de choisir un entier $k$ différent à chaque signature, en effet, dans le cas contraire, il est possible d'extraire la clé privée utilisée !

Implémentons à présent cet algorithme sur la courbe `secp256k1` ($n$ est bien premier).

Nous déclarons en variables globales les paramètres de la courbe.

Pour générer aléatoirement la clé secrète $n_A$ d'Alice, nous réutilisons la fonction `générateur(c)` et créons une fonction `créationclés(c)` qui renvoie un couple de clé secrète et clé publique.

```

def générateur(c):
    """Renvoie une clé aléatoire de c bits sous forme hexadecimal"""
    a = os.urandom(c // 8)
    return (int.from_bytes(a, byteorder='big'))


def créationclés(c):
    nA = générateur(c)    # Alice génère sa clé privée
```

Nous créons ensuite une fonction `hash(message)` qui va renvoyer l'empreinte d'une chaine de caractère. Nous choisissons la fonction de hachage SHA-3 intégrée dans le module `hashlib`. Nous effectuons si nécessaire une troncature grâce à la commande `>>`, comme expliqué dans la remarque précédente.

```
    return(nA, PA)


def hash(message):
    a = hashlib.sha3_512(bytes(message, "utf-8")).digest()
    e = int.from_bytes(a, byteorder='big')
    if e.bit_length() >= n.bit_length():
        z = e >> (e.bit_length() - n.bit_length())
```

Nous transcrivons ensuite directement l'algorithme en créant une fonction de signature et une fonction de vérification.

```
    else:
        return(e)


def signature(message):
    r = 0
    s = 0
    while r == 0 or s == 0:
        k = random.randint(1, n - 1)
        x = (k * G).x
        r = x % n
        s = ((hash(message) + r * nA) * invmod(k, n)) % n
    return(r, s)


def vérification(PA, message, r, s):
    if r < 1 or r > (n - 1) or s < 1 or s > (n - 1):
        return("Signature invalide")

    u1 = (hash(message) * invmod(s, n)) % n
    u2 = (r * invmod(s, n)) % n
```

Nous pouvons vérifiez l'implémentation :

```
>>> nA,PA = créationclés(256)
>>> r,s = signature("Bonjour")
>>> vérification(PA,"Bonjour",r,s)
'Signature valide'
>>> vérification(PA,"Bonjour!",r,s)
'Signature invalide'
```

Comme prévu, si le message est modifié, la signature qui correspondait au message original n'est plus valide avec le message modifié ?

### 4.2.5 Remarques sur l'implémentation

Nous avons fait le choix dans l'implémentation d'utiliser les classes qui imposent une certaine complexité en rajoutant une couche supplémentaire, mais permettent une meilleure lisibilité dans le code. De plus, à chaque création d'un point, nous vérifions par un calcul que le point appartient bien à la courbe pour se prémunir d'erreurs, cependant, ceci n'est pas nécessaire en théorie. Dans l'implémentation du chiffrement de ElGamal, le message envoyé correspond à un point de la courbe, si on souhaite transmettre un message d'un autre type, du texte par exemple, il est nécessaire de trouver un codage sur les points de la courbe. Cependant, en général le chiffrement asymétrique n'est pas utilisé pour l'échange de données, à cause de sa plus grande lenteur face au chiffrage symétrique. Ainsi, le chiffrement asymétrique est seulement utilisé à l'initialisation d'une communication pour échanger des clés symétriques qui permettront ensuite de chiffrer le reste de l'échange.

## 4.3 Le problème du choix d'une courbe elliptique

Pour une utilisation cryptographique, le choix d'une courbe elliptique est primordial, en effet, choisir une certaine courbe elliptique peut altérer la sécurité d'un système en facilitant certaines attaques.

On considère une courbe elliptique $E$ définie dans un corps premier fini $\mathbb{F}_p$ et $G$ un point de $E$ qui définira un sous-groupe.

- L'ordre du point $G$ ne doit pas être un multiplie trivial d'un nombre premier. Dans le cas contraire, il est possible de résoudre le problème du logarithme discret à partir de l'algorithme de Pohlig-Hellman qui divise le problème en plusieurs sous-problèmes en utilisant le théorème des restes chinois.
- Le nombre de points de $E$ (ie le cardinal de $E(\mathbb{F}_p)$ doit être différent de $p$. Dans le cas contraire, la courbe $E$ est une courbe à anomalie. Il est alors possible de résoudre plus facilement le problème du logarithme discret.
- Dans certaines courbes particulières appelées courbes supersingulières, il est possible de transférer le problème du logarithme discret d'un un groupe où il sera plus facile à résoudre.

Nous remarquons que la détermination du nombre de points d'une courbe et de l'ordre du point de base est important. L'algorithme de Schoof est un algorithme déterministe qui permet de compter les points d'une courbe ellipiques avec un complexité de $O(\log^8 p)$. Il se base notamment sur le théorème de Hasse énonçant que pour une courbe elliptique $E$ définie sur un corps $\mathbb{F}_p$, on a $|\, p + 1 - \text{card}E(\mathbb{F}_p)\,| \leq 2\sqrt{p}$.

# Bibliographie


[1] National Institute of Standards and Technology (NIST). *FIPS PUB 186-4, Digital Signature Standard (DSS)*. 2013. URL : https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[2] Balthazar Bauer, Pierre Donat-Bouillud et Victor Durand. *Courbes elliptiques et cryptographie*. ENS Rennes, 2011. URL : http://perso.eleves.ens-rennes.fr/~pdonatbo/documents/maths/cryptoEllip.pdf.

[3] Andrea Corbellini. *Elliptic Curve Cryptography : ECDH and ECDSA*. 2015. URL : http://andrea.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/.

[4] Andrea Corbellini. *Elliptic Curve Cryptography : finite fields and discrete logarithms*. 2015. URL : http://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms.

[5] Taher ElGamal. « A public key cryptosystem and a signature scheme based on discrete logarithms ». In : *IEEE transactions on information theory* 31.4 (1985), p. 469-472. URL : https://ieeexplore.ieee.org/document/1057074/.

[6] Philippe Guillot. *Courbes elliptiques. Une présentation élémentaire pour la cryptographie*. Hermès - Lavoisier, 2010. ISBN : 9782746223929.

[7] Michael Hägler. *Courbes elliptiques et cryptographie*. 2006. URL : http://math.univ-bpclermont.fr/~rebolledo/page-fichiers/projetMichael.pdf.

[8] Marc Joye. « Introduction élémentaire à la théorie des courbes elliptiques ». Thèse de doct. Université catholique de Louvain, 1995. URL : http://joye.site88.net/theses/Joye_DEA.pdf.

[9] Alain Kraus. *Cours de cryptographie MM067 - 2012/13*. Université Pierre et Marie Curie, 2012-2013. URL : http://www.math.univ-paris13.fr/~boyer/enseignement/crypto/Chap7.pdf.

[10] Emmanuel Peyre. *Corps finis et courbes elliptiques*. Université de Grenoble, 2014. URL : http://www-fourier.ujf-grenoble.fr/~peyre/enseignement/poly-SCCI-2014.pdf.