Courbes elliptiques et cryptographie

Balthazar BAUER Pierre DONAT-BOUILLUD Victor DURAND

2011

Introduction

Pour un jeu de données quelconque, le nombre d'éléments susceptibles d'être cryptés est généralement fini : alphabet, nombres exprimables sur un octet... La cryptographie opère donc de façon privilégiée sur les groupes finis, et, après application d'un isomorphisme idoine, sur $(\mathbb{Z}/k\mathbb{Z} \setminus \{0\}, \times)$ où k est premier, qui constitue le groupe le plus naturel. Alors que les outils de cryptanalyse deviennent plus raffinés, que les ordinateurs ou les réseaux de machines connectées gagnent en puissance, généraliser ce groupe, proposer un grand nombre, voire une infinité de groupes finis où pourront être mis en place des algorithmes de cryptographie analogues à ceux utilisés dans le groupe $(\mathbb{Z}/k\mathbb{Z} \setminus \{0\}, \times)$, complexifie la cryptanalyse, pour une plus grande sécurité des données. L'explosion structurelle qui en résulte empêche l'utilisation d'algorithmes adaptés spécifiquement pour un groupe. Et ce sont les courbes elliptiques [1] sur des corps finis qui définissent chacune un groupe, dont l'exploitation comme moyen de cryptographie efficace fera l'objet de cet étude.

1 Fondements mathématiques

1.1 Aperçu géométrique : Courbes elliptiques réelles

Dans ce paragraphe, on définit les courbes elliptiques sur le corps $\mathbb{R}$ des nombres réels et on définira et illustrera géométriquement (ce qui est plus difficile avec un corps fini) l'addition et le doublement. On note $\mathcal{P}$ le plan affine $\mathbb{R}^2$. Soit de plus Q_∞ un objet n'appartenant pas à $\mathcal{P}$. On note alors $\mathbb{P}$ l'ensemble $\mathcal{P} \cup \{Q_\infty\}$ [2]. On note $Q_\infty = (\infty, \infty)$.

Définition 1.1 (Courbe elliptique réelle)**.** *On appelle courbe elliptique sur $\mathbb{R}$ la réunion de toute courbe plane d'équation $y^2 = x^3 + ax + b$ (avec a et b deux réels vérifiant $4a^3 + 27b^2 \neq 0$) et de $\{Q_\infty\}$.*

Soit $\mathcal{E}$ une courbe elliptique réelle.
$\exists (a,b) \in \mathbb{R}^2$, $\begin{cases} \mathcal{E} = \{Q_\infty\} \cup \{(x,y) \in \mathcal{P} | y^2 = x^3 + ax + b\} \\ 4a^3 + 27b^2 \neq 0 \end{cases}$

Proposition 1.1. *La courbe plane réelle $\mathcal{E} \setminus \{Q_\infty\}$ admet en tout point une tangente.*

Démonstration. Ce résultat découle directement du fait que les applications $x \longmapsto \sqrt{x^3 + ax + b}$ et $x \longmapsto -\sqrt{x^3 + ax + b}$ sont dérivables sur leur ensemble de définition privé des racines de $X^3 + aX + b$ et que leurs courbes représentatives admettent une tangente verticale en ces derniers points. □

Soient $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ deux points de $\mathcal{E}$.
Si $P_1 = Q_\infty$, on pose $A(P_1, P_2) = P_2$.
Si $P_2 = Q_\infty$, on pose $A(P_1, P_2) = P_1$.
Si $P_1 = P_2 \neq Q_\infty$ et $y_1 \neq 0$, la tangente à $\mathcal{E} \setminus \{Q_\infty\}$ coupe $\mathcal{E}$ en un deuxième point P'_3, et on note $A(P_1, P_2)$ le symétrique orthogonal de $P'3$ par rapport à l'axe des abscisses.
Si $P_1 = P_2 \neq Q_\infty$ et $y_1 = 0$, on pose $A(P_1, P_2) = Q_\infty$.
Si $x_1 = x_2 \neq \infty$ et $y_1 \neq y_2$, on pose $A(P_1, P_2) = Q_\infty$.
Si $x_1 \neq x_2$, la droite affine (P_1P_2) coupe $\mathcal{E}$ en un troisième point, et on note $A(P_1, P_2)$ le symétrique orthogonal de ce troisième point par rapport à l'axe des abscisses.

Proposition 1.2. *L'application A de $\mathcal{E}$ dans $\mathbb{P}$ ainsi définie est une loi de composition interne sur $\mathcal{E}$, que l'on note $+$ et $(\mathcal{E}, +)$ est un groupe abélien.*

Démonstration. La commutativité et l'existence d'un élément neutre (Q_∞) sont immédiates. L'associativité peut se démontrer en établissant une expression algébrique des coordonnées du point somme , mais nous épargnerons au lecteur ce calcul lourd et peu intéressant. L'établissement de ces expressions est proposé en annexe. L'existence d'un symétrique à tous les points de la courbe découle directement de la définition de A. □

1. Les courbes elliptiques n'ont qu'un lointain rapport avec l'ellipse. Les courbes elliptiques tirent leur nom des intégrales elliptiques, qui servent à calculer la longueur d'un arc d'ellipse.

2. Q_∞ est en réalité un point à l'infini mais nous n'avons dans cette étude ni la prétention ni l'espace d'entrer dans les méandres de la géométrie projective. Q_∞ restera donc pour nous un "objet n'appartenant pas à P" mais l'appellation "point à l'infini" pourra nous aider à comprendre certaines conventions que nous serons amenés à prendre plus bas.

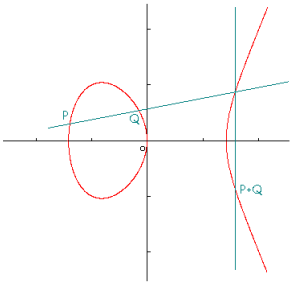

FIGURE 1 – Courbe elliptique et loi de composition

1.2 Structure de groupe

Ici, on présente les courbes elliptiques sur un corps quelconque et on définit l'addition des points. Soit donc $\mathbb{K}$ *un corps commutatif de caractéristique*[3] *différente de 2 et 3. Soit également* P_∞ *un objet n'appartenant pas à* $\mathbb{K}^2$*. On note* $P_\infty = (\infty, \infty)$*.*

Définition 1.2 (Courbe elliptique sur $\mathbb{K}$)**.** *On appelle courbe elliptique sur* $\mathbb{K}$ *la réunion de* $\{P_\infty\}$ *et de toute courbe plane sur* $\mathbb{K}$ *d'équation* $y^2 = x^3 + ax + b$*, avec a et b deux éléments de* $\mathbb{K}$ *vérifiant* $4a^3 + 27b^2 \neq 0$ *.*

Soit $\mathcal{E}$ *une courbe elliptique sur* $\mathbb{K}$ *:*

$$\exists (a,b) \in \mathbb{K}^2 \text{ , } \begin{cases} \mathcal{E} = \{P_\infty\} \cup \{(x,y) \in \mathbb{K}^2 | y^2 = x^3 + ax + b\} \\ 4a^3 + 27b^2 \neq 0 \end{cases}$$

Définition 1.3 (point régulier)**.** *On appelle point régulier de* $\mathcal{E}$ *tout point de* $\mathcal{E}$ *différent de* P_∞ *.*

Soient $P_1 = (x_1, y_1)$ *et* $P_2 = (x_2, y_2)$ *deux points de* $\mathcal{E}$ *(* $(x_1, y_1, x_2, y_2) \in \mathbb{K} \cup \{\infty\})^4$ *).*
- Si $P_1 = P_\infty$ *, on pose* $\mathcal{A}(P_1, P_2) = P_2$
- Si $P_2 = P_\infty$*, on pose* $\mathcal{A}(P_1, P_2) = P_1$
- Si $P_1 = P_2$ *et* $y_1 = 0$*, on pose* $\mathcal{A}(P_1, P_2) = P_\infty$
- Si $x_1 = x_2$ *et* $y_1 \neq y_2$ *, on pose* $\mathcal{A}(P_1, P_2) = P_\infty$
- Si $P_1 = P_2 \neq P_\infty$ *et* $y_1 \neq 0$ *, on pose*

$$\mathcal{A}(P_1, P_2) = (\lambda^2 - 2x_1, -\lambda^3 + 3\lambda x_1 - y_1)$$

avec $\lambda = \frac{3x_1^2 + a}{2y_1}$ *(bien défini car* $y_1 \neq 0$ *).*
- Si $x_1 \neq x_2$*, on pose*

$$\mathcal{A}(P_1, P_2) = (\lambda^2 - x_1 - x_2, -\lambda^3 + 2\lambda x_1 + \lambda x_2 - y_1)$$

avec $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ *(bien défini car* $x_1 \neq x_2$ *).*

Proposition 1.3. *L'application* $\mathcal{A} : \mathcal{E}^2 \longrightarrow \mathbb{K}^2 \cup \{P_\infty\}$ *ainsi définie est une loi de composition interne sur* $\mathcal{E}$*, que l'on note + et* $(\mathcal{E}, +)$ *est un groupe commutatif.*

Démonstration. On garde pour cette démonstration les notations plus haut.

Pour montrer que la somme de deux points est toujours dans $\mathcal{E}$, il suffit de regarder les deux derniers cas examinés plus haut, les autres étant triviaux. Il suffit pour cela de montrer dans chaque cas que les coordonnées (x_S, y_S) vérifient $y_S^2 = x_S^3 + ax_S + b$. Ce calcul sans difficulté particulière est laissé au bon soin du lecteur. La commutativité est triviale, l'existence d'un élément neutre (P_∞) aussi. Un point régulier (x,y) de $\mathcal{E}$ admet (x,-y) (qui appartient aussi à $\mathcal{E}$) comme symétrique, et le point à l'infini est son propre symétrique pour la loi $+$. Une démonstration de l'associativité de $+$ autre que de nombreux calculs fastidieux exigerait de mettre en place des outils que nous n'aurons pas ici l'occasion d'introduire. □

1.3 Ordre du groupe

Fonctions rationnelles et morphismes

Soit q un entier naturel non nul et soit $\mathbb{F}_q$ *un corps de cardinal q. Soit* $\mathbb{F}$ *la clôture algébrique*[4] *de* $\mathbb{F}_q$*. Soit* $\mathcal{E}$ *une courbe elliptique sur* $\mathbb{F}_q$*.*

3. Le plus petit entier non nul n tel que $n = 0$ dans $\mathbb{K}$
4. C'est un corps qui contient $\mathbb{F}_q$ tel que tout polynôme à coefficients dans ce corps de degré supérieur ou égal à 1 admette au moins une racine dans ce corps.

$\exists (a,b) \in \mathbb{F}_q^2$, $\begin{cases} \mathcal{E} = \{P_\infty\} \cup \{(x,y) \in \mathbb{F}_q^2 | y^2 = x^3 + ax + b\} \\ 4a^3 + 27b^2 \neq 0 \end{cases}$

On note $\mathcal{E}(\mathbb{F})$ la courbe elliptique sur $\mathbb{F}$ définie par l'équation $y^2 = x^3 + ax + b$.

Définition 1.4 (Polynômes et fonctions rationnelles sur $\mathcal{E}$)**.**

1. *L'anneau $\mathbb{F}[\mathcal{E}]$ des polynômes sur $\mathcal{E}$ est l'anneau quotient de $\mathbb{F}[x,y]$ par $y^2 - x^3 - ax - b$. C'est un anneau intègre.*
2. *Le corps $\mathbb{F}(\mathcal{E})$ des fonctions rationnelles sur $\mathcal{E}$ est le corps des fractions de cet anneau.*

Dans les deux cas, il s'agit de l'anneau des polynômes (ou du corps des fractions) dans lequel deux expressions qui diffèrent d'un multiple de $y^2 - x^3 - ax - b$, c'est à dire deux expressions qui prennent la même valeur en tout point de la courbe $\mathcal{E}(\mathbb{F})$ sont égales.

Proposition-définition 1.1 (Quelques propriétés des polynômes)**.** *Tout polynôme $p(x,y)$ de $\mathbb{F}[\mathcal{E}]$ peut s'écrire sous la forme $r(x) + ys(x)$ où r et s sont deux polynômes de $\mathbb{F}[x]$.*

*On appelle alors **conjugué** d'un polynôme $p(x,y) = r(x) + ys(x)$ le polynôme $\overline{p} = r(x) - ys(x)$, **norme** de p le polynôme $n(p) = p * \overline{p}$. C'est un polynôme de $\mathbb{F}[x]$, dont le degré sera appelé **degré** de p et noté $deg(p)$.*

Démonstration. Le premier résultat s'obtient en remplaçant y^2 par $x_3 + ax + b$ chaque fois que c'est possible dans l'expression de p.

$n(p) = (r(x) + ys(x))(r(x) - ys(x)) = r^2(x) - y^2 s^2(x) = r^2(x) - (x^3 + ax + b)s^2(x)$ est bien un polynôme en x seulement. □

Définition 1.5 (degré d'une fraction rationnelle)**.** *Soit f une fonction rationnelle sur $\mathcal{E}$. Il existe deux polynômes g et h de $\mathbb{F}[\mathcal{E}]$ tels que $f = \frac{g}{h}$. On appelle degré de f la quantité $deg(f) = deg(g) - deg(h)$. On admet que ce nombre ne dépend pas du choix de g et h.*

Convention 1.1.

1. $\forall B \in \mathbb{F}, B \times \infty = \infty$ *et* $B + \infty = \infty$
2. $\frac{1}{\infty} = 0$
3. $\infty \times \infty = \infty$

Proposition-définition 1.2 (pôle d'une fraction rationnelle, évaluation des fonctions rationnelles en un point)**.** *Soit f une fonction rationnelle sur $\mathcal{E}$. Soit $P = (x,y)$ un point régulier de $\mathcal{E}(\mathbb{F})$. On dit que P est régulier pour la fonction rationnelle f s'il existe un représentant $\frac{g}{h}$ de f ($(h,g) \in \mathbb{F}[\mathcal{E}]^2$) tel que $h(x,y) \neq 0$. Sinon, on dit que P est un pôle de f. On dit que P est un zéro de f s'il existe un représentant $\frac{g}{h}$ de f tel que $h(x,y) \neq 0$ et $g(x,y) = 0$. Si P est régulier pour f, la valeur de f en P est $f(P) = \frac{g(x,y)}{h(x,y)}$. Si P est un pôle, on note $f(P) = \infty$. On dit que P_∞ est un pôle pour f si $deg(f) > 0$, et que P_∞ est un zéro de f si $deg(f) < 0$. Si $deg(f) = 0$, la valeur de f en P_∞ est égale au rapport du coefficient du terme de plus haut degré de g et de celui de h. La valeur d'une fraction rationnelle en un de ses zéros est 0. La valeur d'une fraction rationnelle en un pôle est notée ∞.*

Définition 1.6 (point ordinaire, point spécial)**.** *Soit $P = (x,y)$ un point régulier de $\mathcal{E}(\mathbb{F})$. On dit que P est un point spécial si $y = 0$. Sinon, P est un point ordinaire.*

Définition 1.7. *Soit P un point de $\mathcal{E}$. Soit u une fraction de $\mathbb{F}(\mathcal{E})$. On dit que u est une uniformisante en P si $u(P) = 0$ et toute fraction f de $\mathbb{F}(\mathcal{E})$ peut s'écrire sous la forme $f = u^d * r$, avec d entier et r une fraction rationnelle dont P n'est ni un zéro ni un pôle. Pour une fonction rationnelle f donnée, on admet que la valeur de d ne dépend pas de l'uniformisante u choisie. On appelle alors **ordre de f en** P et on note $ord_P(f)$ cet entier relatif. On admet de plus que tout point de $\mathcal{E}(\mathbb{F})$ admet une uniformisante.*

Proposition 1.4. *Soit $f \in \mathbb{F}(\mathcal{E})$. Soit $P \in \mathcal{E}(\mathbb{F})$*

- *Si $ord_P(f) \geqslant 1$, P est un zéro de f*
- *Si $ord_P(f) \leqslant -1$, P est un pôle de f*
- *Si $ord_P(f) = 0$, P n'est ni un zéro ni un pôle pour f.*

Démonstration. Immédiat en considérant l'écriture $f = u^{ord_P(f)} r$. □

Définition 1.8 (isogénie)**.** *Soit ϕ une application de $\mathcal{E}$ dans $\mathcal{E}(\mathbb{F})$.*

On dit que ϕ est une isogénie de $\mathcal{E}$ si :

1. $\exists (r,s) \in \mathbb{F}(\mathcal{E})^2, \forall P \in \mathcal{E}, \phi(P) = (r(P), s(P))$
2. *ϕ est un endomorphisme du groupe $(\mathcal{E}(\mathbb{F}), +)$*

On note alors $\phi = (r,s)$.

Une condition pour que $\phi = (r,s)$ soit une application de $\mathcal{E}(\mathbb{F})$ dans $\mathcal{E}(\mathbb{F})$ est que, pour tout point P de $\mathcal{E}(\mathbb{F})$, $(r(P),s(P))$ appartienne à la courbe $\mathcal{E}(\mathbb{F})$, c'est à dire que $s^2 = r^3 + ar + b$.

On en déduit notamment que r et s ont alors les mêmes pôles : Soit P un pôle de s.
$s^2(P) = s(P) * s(P) = \infty = r(P)^3 + ar(P) + b$
$r(P) * (r(P)^2 + a) = \infty - b = \infty$
Donc $r(P) = \infty$ et P est un pôle de r. La preuve dans l'autre sens est du même goût.

Définition 1.9. *On appelle morphisme de la courbe elliptique $\mathcal{E}(\mathbb{F})$ une application de la forme $P \longmapsto (r(P),s(P))$, avec r et s des fonctions rationnelles de $\mathbb{F}(\mathcal{E})$ vérifiant la condition ci-dessus.*

Proposition 1.5. *Soit μ un morphisme de la courbe $\mathcal{E}(\mathbb{F})$. Alors μ est une isogénie si et seulement si μ est non constant et $\mu(P_\infty) = P_\infty$.*

Démonstration. Le sens direct est immédiat : un morphisme de groupes envoie l'élément neutre du groupe de départ sur l'élément neutre du groupe d'arrivée. On admet le sens réciproque. □

Lemme 1.1. *Soit u une fonction rationnelle de $\mathbb{F}(\mathcal{E})$ et P un point de $\mathcal{E}(\mathbb{F})$. Si u est une uniformisante en P, alors $ord_P(u) = 1$.*

Démonstration du lemme. Si u est une uniformisante en P, alors $u = u^1 * 1$ avec $1(P) \notin \{0,\infty\}$, et $ord_P(u) = 1$. □

Proposition-définition 1.3 (indice de ramification d'une isogénie, degré)**.** *Soit $\phi = (r,s)$ une isogénie de $\mathcal{E}$. Soit u une uniformisante en $\phi(P)$. La quantité $\eta_P(\phi) = ord_P(u \circ \phi)$ ne dépend pas de l'uniformisante u choisie.*

*On admet de plus qu'elle est égale pour tous les points P de $\mathcal{E}(\mathbb{F})$. On l'appelle **indice de ramification** de l'isogénie ϕ et on la note $\eta(\phi)$. On appelle **degré** de ϕ l'entier :*

$$deg(\phi) = Card(Ker(\phi)) * \eta(\phi)$$

Démonstration. Soient u_1 et u_2 deux uniformisantes en $\phi(P)$. Soit v une uniformisante en P. Il existe une fonction s dont P n'est ni pôle ni zéro telle que : $u_1 \circ \phi = v^{\eta(\phi)} * s$. D'après le lemme 1.1 : u_1 étant une uniformisante en $\phi(P)$, $ord_{\phi(P)}(u_1)$ = 1. Il existe une fonction r telle que $\phi(P)$ n'est ni un pôle ni un zéro de r et vérifiant $u_2 = u_1^1 * r = u_1 * r$.
v étant une uniformisante en P, en notant $d = ord_P(u_1 \circ \phi)$, il existe s une fonction vérifiant $s(P) \neq 0, s(P) \neq \infty$ et $u_1 \circ \phi = v^d s$.

On a alors $u_2 \circ \phi = r u_1 \circ \phi = (u_1 \circ \phi) * (r \circ \phi) = v^d s(r \circ \phi)$, avec $r \circ \phi$ n'ayant valeur ni 0 ni ∞ en P. On a bien $ord_P(u_1 \circ \phi) = ord_P(u_2 \circ \phi)$.

□

Quelques propriétés préliminaires

Soit n un entier naturel non nul. On suppose que q est une puissance de la caractéristique de $\mathbb{F}$.

Définition 1.10 (groupe de n-torsion)**.** *Soit P un point de $\mathcal{E}(\mathbb{F})$.*
On dit que P est un point de n-torsion de la courbe $\mathcal{E}(\mathbb{F})$ si $nP = P_\infty$. L'ensemble des points de n-torsion de $\mathcal{E}(\mathbb{F})$ est appelé groupe de n-torsion de $\mathcal{E}$. C'est évidemment un sous-groupe de $(\mathcal{E}(\mathbb{F},+)$ et on le note $\mathcal{E}[n]$.

Proposition 1.6 (Admise)**.** *Le groupe de n-torsion est isomorphe au groupe produit $(\mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z},+)$.*

On peut donc trouver deux points S et T de n-torsion tels que tout point P de $\mathcal{E}[n]$ peut s'écrire (de manière unique) $P = aS + bT$, avec $(a,b) \in (\mathbb{Z}/n\mathbb{Z})^2$. Dans la suite, S et T sont de tels points. On dira que (S,T) est une base de $\mathcal{E}[n]$.

Proposition 1.7. *Le groupe de n-torsion est stable par toute isogénie.*

Démonstration. Soit ϕ une isogénie de $\mathcal{E}(\mathbb{F})$. Soit P un point de n-torsion. $n\phi(P) = \phi(nP) = \phi(P_\infty) = P_\infty$, car ϕ est un morphisme de groupes. □

La restriction d'une isogénie ϕ au groupe de n-torsion peut donc être représentée par une matrice de type $(2,2)$ à coefficients dans $\mathbb{Z}/n\mathbb{Z}$. La matrice $\begin{pmatrix} a & c \\ b & d \end{pmatrix}$ représentera l'isogénie qui à S associe $aS + bT$ et à T associe $cS + dT$.

Proposition 1.8 (admise [5])**.** *On suppose que n n'est pas divisible par la caractéristique du corps $\mathbb{F}$. Soit ϕ une isogénie de $\mathcal{E}(\mathbb{F})$. Alors le degré de l'isogénie ϕ est congru modulo n au déterminant de sa matrice dans la base (S,T).*

5. La démonstration de ce résultat fait intervenir l'accouplement de Weil, que nous n'avons pas ici l'occasion d'introduire.

Définition 1.11 (isogénie de Frobenius)**.** *On appelle isogénie de Frobénius le morphisme (x^q, y^q), c'est à dire l'application :*

$$\phi_q : \begin{array}{l} \mathcal{E} \to \mathcal{E} \\ P = (x, y) \longmapsto (x^q, y^q) \end{array}$$

On rappelle que q est le cardinal du corps $\mathbb{F}_q$ sur lequel est définie la courbe $\mathcal{E}$. En vertu de la proposition 1.5, l'isogénie de Frobenius est une isogénie de $\mathcal{E}(\mathbb{F})$. C'est une isogénie de degré q.

Définition 1.12 (isogénie séparable)**.** *Soit ϕ une isogénie de $\mathcal{E}(\mathbb{F})$. On dit que ϕ est* séparable *si $\eta(\phi) = 1$. Sinon, ϕ est* inséparable*.*

Proposition 1.9. *Le morphisme $\Phi = \phi_q - id$ est une isogénie séparable de degré $Card(\mathcal{E})$.*

Démonstration. Φ est une isogénie comme somme de deux isogénies. Supposons que Φ soit inséparable. Comme ϕ_q est inséparable, on a alors : id est inséparable. Or id est séparable. Donc Φ est séparable. Donc $\eta(\Phi) = 1$. $\forall x \in \mathbb{F}, x \in \mathbb{F}_q \iff x^{q-1} = 1$ [6], donc $Ker(\Phi) = \mathcal{E}$($\mathcal{E}$ étant égal à l'ensemble des points de $\mathcal{E}(\mathbb{F})$ dont les coordonnées sont dans $\mathbb{F}_q \cup \{\infty\}$) . Donc le degré de Φ est égal à $1 \times Card(\mathcal{E})$. □

Définition 1.13. *On appelle trace de la courbe $\mathcal{E}$ l'entier $t = q + 1 - Card(\mathcal{E})$.*

Proposition 1.10. *Le polynôme $X^2 - tX + q$ est polynôme annulateur de $\phi_{q|\mathcal{E}[n]}$.*

Cette proposition est démontrée en annexe.

Théorème 1.1. *(de Hasse) La trace t de $\mathcal{E}$ vérifie :*

$$-2\sqrt{q} \geqslant t \leqslant 2\sqrt{q}$$

Démonstration. On note $A = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ la matrice de l'isogénie de Frobenius dans la base (S, T) du groupe de n-torsion. Calculons, pour deux entiers λ et μ le déterminant δ de la matrice $\lambda A - \mu I$. Les égalités qui suivent s'entendent dans l'anneau $\mathbb{Z}/n\mathbb{Z}$.

$$\delta = \begin{vmatrix} \lambda a - \mu & \lambda c \\ \lambda b & \lambda d - \mu \end{vmatrix} = \lambda^2(ad - bc) - \lambda\mu(a + d) + \mu^2 = \lambda^2 det(A) - \lambda\mu(a + d) + \mu^2$$

D'où, en vertu de la proposition 1.8 :
$deg(\lambda\phi_q - \mu id) \equiv q\lambda^2 - t\lambda\mu + \mu^2[n]$ (Le déterminant de A est congru modulo n au degré de l'isogénie de Frobenius, à savoir q [7]).

Pour un (λ, μ), les deux termes de cette relation ne dépendent pas de n. On peut donc prendre n assez grand pour transformer cet égalité modulo n en égalité dans $\mathbb{Z}$. Donc, pour $(\lambda, \mu) \in \mathbb{Z}^2$, on a :

$$deg(\lambda\phi_q - \mu id) = \mu^2(q\frac{\lambda^2}{\mu^2} - t\frac{\lambda}{\mu} + 1)$$

Le discriminant du trinôme $X^2 - tX + 1$ est $\Delta = t^2 - 4q$. L'égalité précédente implique que la valeur de ce trinôme en tout $r \in \mathbb{Q}$ est positive ou nulle. Donc le discriminant Δ est négatif ou nul. D'où le résultat. □

Théorème 1.2 (des restes chinois)**.** *Soit $k \in \mathbb{N}^*$.*

Soient $n_1, \ldots, n_k$ des entiers naturels deux à deux premiers entre eux et $a_1, \ldots, a_k$ des entiers. On note A le produit des a_i.

$$\exists! x \in \mathbb{Z}/A\mathbb{Z}, \begin{cases} x \equiv a_1[n_1] \\ \ldots \\ x \equiv a_k[n_k] \end{cases}$$

Démonstration. Ce résultat est démontré en annexe. □

6. Le sens direct est trivial en vertu du théorème de Lagrange, et on admet le sens réciproque.
7. Pour la démonstration de $a + d \equiv t[n]$, voir la démonstration de la proposition précédente en annexe.

Calcul du cardinal d'une courbe

Principe de l'algorithme *On veut utiliser le théorème chinois et l'encadrement de la trace t offert par le théorème de Hasse pour en déterminer la valeur exacte. Pour ce faire, on se donne un certain nombre d'entiers m (pas trop grands) deux à deux premiers entre eux et tels que leur produit soit plus grand que $4\sqrt{q}$. Pour chacun de ces m, on considère un point P de m-torsion. On a alors, en vertu de la proposition 1.10 :*

$$\phi_q^2(P) - (t \mod m)\phi_q(P) + (q \mod m)P) = P_\infty$$

En écrivant $P = (x, y)$, on a :

$$(x^{q^2}, y^{q^2}) = (t \mod m)(x^q, y^q) + (q \mod m)(x, y)$$

L'entier $t \mod m$ étant dans $\mathbb{Z}/m\mathbb{Z}$, et m n'étant pas trop grand, on trouve $u = t \mod m$ vérifiant l'égalité précédente par recherche exhaustive. En effectuant cette opération pour tous les m, et en utilisant le théorème chinois, on obtient t, d'où l'on déduit le cardinal de la courbe $\mathcal{E}$.

L'algorithme de Schoof remplit cette tâche :

Entrées : un entier q non nul et une courbe $\mathcal{E}$ sur $\mathbb{F}_q$
Résultat : la trace t de la courbe E
Données : $P = 1$ et $m = 2$, U un tableau suffisamment grand et rempli de 0, et $u = 0$ et M un tableau suffisamment grand rempli de 0
tant que $P \leqslant 4\sqrt{q}$ **faire**
  Soit $P = (x, y)$ un point de m-torsion. Soit $P_1 = (x_1, y_1) = (x^{q^2}, y^{q^2}) + (q \mod l)(x, y)$;
  Soit a =1 ;
  tant que $a(x^q, y^q) \neq P_1$ **faire**
    incrémenter a
  fin
  U.(u)<-a;
  M.(u)<-m;
  P<-P*m ;
  m<- le nombre premier suivant immédiatement m (nextprime(m)) ;
fin
Soit alors M' le tableau composé des u+1 premiers éléments du tableau M et U' celui composé des $u + 1$ premiers éléments de U ;
On a alors $t =$ restes_chinois U' M' ;

Algorithme 1 : Algorithme de Schoof

2 Chiffrement

2.1 Le problème du logarithme discret

Cas général

On se place dans un groupe fini $(E, \times)$. Soit p l'ordre de ce groupe. On note $\mathbb{1}_G$ l'élément neutre de ce groupe. On choisit un élément g de $(E, \times)$, et on appelle G le sous-groupe engendré par g. Soit n l'ordre de g.
On voudrait résoudre :

$$g^x = z \text{ où } z \in G \tag{1}$$

On considère :

$$\phi : \begin{cases} [\![0, n-1]\!] \longrightarrow G \\ x \longmapsto g^x \end{cases} \tag{2}$$

Proposition 2.1. *ϕ est bijective.*

Démonstration. Soit $(a, b) \in [\![0, n-1]\!]^2$ tel que $\phi(a) = \phi(b)$. On peut supposer que $a \leq b$.

$$g^a = g^b \text{ soit : } g^{b-a} = \mathbb{1}_G$$

$$0 \leq b - a \leq n - 1 \text{ donc } b - a = 0 \text{ , } b = a$$

Ainsi, ϕ est injective. Comme G est fini, ϕ est bijective. □

L'unicité d'une solution à l'équation (1) dans $[\![1, n-1]\!]$ est donc assurée.

Définition 2.1. *On appelle logarithme discret de z l'unique solution de (1). C'est la bijection réciproque de ϕ de (2). On le notera dorénavant : $\mathcal{L}_g$.*

Le problème du logarithme discret consiste, étant donner un élément $x \in G$, à trouver $\mathcal{L}_g(x)$.

L'élévation de g à une puissance $l \in \mathbb{N}^*$ peut se faire avec une complexité $O(\log l)$ en utilisant une méthode diviser pour régner. Le calcul du logarithme discret est plus lent. La méthode naïve de recherche exhaustive a une complexité exponentielle. On verra qu'il existe des méthodes relativement plus efficaces dans un groupe quelconque. Pour certains groupes spécifiques, la complexité peut même être encore améliorée, comme dans le groupe bien connu $(\mathbb{Z}/p\mathbb{Z} \setminus \{0\}, \times)$ où p est premier, pour atteindre une complexité sous-exponentielle.

Données *: Un générateur g de G*
Entrées *: Un nombre $x \in G$*
Résultat *: le logarithme discret de x*
rsultat ← g;
i ← 1;
tant que *résultat ≠ x et $i < |G|$* **faire**
    résultat ← g ∗ résultat;
    incrémenter i;
fin
si *résultat = x* **alors**
    retourner $i - 1$
sinon
    retourner *Logarithme non trouvé*
fin

***Algorithme 2** : L'algorithme naïf*

Proposition 2.2. *L'algorithme naïf de recherche du logarithme discret a une complexité exponentielle dans le pire des cas.*

Dans le cas où G est un sous-groupe de $\mathbb{Z}/p\mathbb{Z}$. g, générateur de G, comporte $\log_2(g)$ chiffres en binaire. La multiplication de deux nombres composés respectivement de n et de m chiffres a une complexité de $O(nm)$ avec l'algorithme usuel (appris à l'école primaire). On effectue $n-1$ multiplications par g dans le pire des cas, où n est l'ordre de G. Ainsi, l'algorithme naïf de calcul du logarithme discret a une complexité de $O((log_2(g))^n)$ Il s'agit bien d'une complexité exponentielle. On peut généraliser l'algorithme à un groupe à r générateurs $g_1, g_2, ..., g_r$, avec $r \geq 2$, en appliquant les étapes ci-dessus à chacun des générateurs $g_1, ..., g_r$.

On admet que la complexité dans le pire des cas reste dans le cas général de la forme $O(a^n)$ où $a \in]1, +\infty[$. □

On obtient ainsi une fonction à sens unique :

Définition 2.2. *On dit qu'un calcul est difficile si la complexité des algorithmes le calculant est au moins exponentielle. Dans le cas contraire, on dit que le calcul est facile. Une fonction f de $A \to B$, qu'on suppose bijective, est à sens unique si, connaissant $x \in A$, il est facile de calculer $f(x)$, mais difficile, connaissant $y \in B$, de calculer $f^{-1}(y)$.*

Un calcul difficile prendra en pratique un temps de quelques siècles, même sur des supercalculateurs, pour des entrées de taille importante. Pour la cryptographie, c'est la taille de la clé qui entre en compte. Dans le cas du logarithme discret, et en réalité de toutes les fonctions à sens uniques, la fonction n'est que supposée *être à sens unique : on n'a pas encore trouvé, et on espère ne pas trouver d'algorithmes à complexité polynomiale qui résolve le problème*[8].

Logarithme discret dans $\mathcal{E}(\mathbb{Z}/p\mathbb{Z})$

Le problème du logarithme discret peut être transposé aux courbes elliptiques, sur le groupe $\mathcal{E}(\mathbb{Z}/p\mathbb{Z})$ d'ordre q. On dispose d'un point Q de $\mathcal{E}(\mathbb{Z}/p\mathbb{Z})$. Soit $P \in \mathcal{E}(\mathbb{Z}/p\mathbb{Z})$. L'équation similaire à l'équation (1) s'écrit alors :

$$nP = Q \tag{3}$$

Ici, $\mathcal{L}_P(Q) = n$

On peut montrer que le groupe $\mathcal{E}(\mathbb{Z}/p\mathbb{Z})$ est un groupe cyclique, ou le produit de deux groupes cycliques : il admet un, ou deux générateurs.

8. Ces problèmes entrent souvent dans la catégorie des problèmes NP-complets, qui sont la clé et l'enjeu d'un des problèmes du millénaire, le problème $P = NP$.

2.2 L'échange de clés de Diffie-Hellman

Les fonctions à sens uniques, en permettant la communication d'une information transformée, en s'assurant que l'information de départ, ne peut être retrouvée, donnent une solution au problème du canal de transmission : en cryptographie classique, ou symétrique, une clé doit être partagée entre les deux parties pour pouvoir assurer chiffrement et cryptanalyse. Bob envoie la clé à Alice[9]*. Hubert, leur ennemi juré, s'est malheureusement placé sur le trajet de la clé et l'intercepte. Il peut alors sournoisement lire les messages qu'Alice et Bob s'envoient, et en envoyer d'autres en se faisant passer pour eux. Une première solution est de transmettre la clé en morceaux, de façon à ce qu'Hubert ne puisse s'en emparer.*

Dans le protocole d'échange de clés de Diffie-Hellman, Alice et Bob rendent publique une courbe elliptique $\mathcal{E}$ définie sur un corps fini Q (de la forme $\mathbb{Z}/n\mathbb{Z}$) et $P \in \mathcal{E}(Q)$. Alice choisit $x \in Q \setminus \{0\}$ et Bob, $y \in Q \setminus \{0\}$. Alice calcule alors :

$$X = xP \tag{4}$$

et Bob :

$$Y = yP \tag{5}$$

Chacun envoie à l'autre le résultat de 4, ou de 5, qui est une clé publique.

Alice considère :

$$K = xY$$

et Bob :

$$K' = yX$$

On obtient :

$$K = xY = x(yP) = y(xP) = yX = K' \tag{6}$$

K ou K' constituent une clé commune, secrète. On suppose qu'Hubert intercepte les clés publiques X et Y, et qu'il connaît P. Il ne peut découvrir K car il lui faudrait connaître x, ou bien y. Pour cela, il lui faut calculer le logarithme discret de X ou de Y, ce qui est difficile.

Ainsi, Alice et Bob ont échangé une clé privée de façon sécurisée. Ils peuvent ensuite utiliser un algorithme de cryptographie symétrique pour communiquer.

Cependant, cet algorithme utilisé tel quel est vulnérable à l'attaque de l'homme du milieu. Dans cette attaque, Hubert est en mesure d'intercepter les clés publiques, mais aussi de les modifier. Il peut donc choisir x' et y', envoyer $x'P$ et $y'P$. La clé K calculée par Alice et Bob est alors connue d'Hubert : c'est $x'y'P$. Pour parer cette attaque, il faut pouvoir identifier la provenance des clés, les signer, souvent à l'aide de certificats établis par une tierce autorité.

2.3 Chiffrement Elgamal

On reprend les notations de la partie précédente : on dispose toujours de $P \in \mathcal{E}(Q)$.

Le chiffrement Elgamal permet de crypter un message de façon asymétrique : Bob veut de nouveau envoyer un message $M \in \mathcal{E}(Q)$ à Alice de sorte que seule Alice puisse décoder le message. Alice a choisi $y \in Q$ et calcule $Y = yP$: y est sa clé secrète, Y sa clé publique, qu'elle dévoile à tous. Bob choisit $k \in Q$, qui sera sa clé secrète. Bob calcule :

$$\begin{aligned} A &= kP \\ G &= kY \\ B &= M + G \\ C &= (A, B) \end{aligned}$$

Il envoie le couple C de points à Alice. Elle réalise alors les opérations suivantes :

$$\begin{aligned} G' &= yA \\ M' &= B - G' \end{aligned}$$

Elle retrouve ainsi le message M car $M = M'$. En effet :

$$\begin{aligned} M' = B - G' &= B - yA \\ &= M + G - yA \\ &= M + kY - ykP \\ &= M + kyP - ykP \\ &= M \end{aligned}$$

9. Alice et Bob forment le couple récurrent de la cryptographie.

Là encore, la sécurité du système repose sur le problème du logarithme discret : trouver k revient à calculer le logarithme discret de A. Il faut donc représenter les informations à envoyer avec des points de la courbe elliptique. Cela peut se faire en encodant le message dans la première ou la seconde coordonnée du point. Si le message est trop long, on utilise plusieurs points.

3 Cryptanalyse

Dans cette partie lorsqu'on parlera du coût des algorithmes c'est en considérant que la loi de composition interne du groupe sur lequel on travail (en l'occurrence $(E,+)$) est une opération élémentaire. Si du point de vue d'un ordinateur c'est souvent faux (en particulier pour les courbes elliptiques) cela simplifie les comparaisons de coût pour différents algorithmes.

3.1 Pas de bébé, pas de géant

Principe théorique *Soit $c \in \mathbb{N}^*$ Le principe est le suivant, en connaissant a et b deux élément d'un groupe $(E,+)$ fini, avec a d'ordre c (on appellera le sous-groupe de E engendré par a, F), on cherche un entier n (défini modulo c car F est un groupe cyclique d'ordre c) tel que $n \times a = b$, on prend m un entier tel que $m \simeq \sqrt{c}$ et on sait qu'il existe un couple (i,j) d'entiers tel que $n = mi + j$ avec $0 \leqslant i \leqslant c/m$ et $0 \leqslant j \leqslant m-1$ (principe de la division euclidienne).*

On calcule tout les couples $(i, mi * a)$ possibles (les pas de géants) puis tous les couples $(j, b - j * a)$ (les pas de bébé). Puis on cherche un couple de chaque catégorie ayant un second terme identique.

Le résultat sera alors :

$$n = mi + j \tag{7}$$

Grand principe : *Si on néglige le coût de la recherche des deux couples (ce qui est raisonnable si on munit l'ensemble d'une relation d'ordre, qu'on range les éléments calculés à partir de cette relation et qu'on utilise des méthodes dichotomiques), une fois qu'ils ont tous été calculés, on se rend compte que l'on a un coût en m : car il y a m/c i possibles (donc en $\sqrt{c}$) et mj possibles (donc également linéaire en $\sqrt{c}$). On a donc un algorithme qui a un coût linéaire en $\sqrt{c}$ au lieu d'être linéaire en c.*

A noter qu'il est nécessaire de connaître l'ordre de c pour pouvoir utiliser cet algorithme. Pour cela on utilise l'algorithme de Schoof qui a un coût négligeable devant le calcul lors des "pas".

3.2 La méthode du kangourou

Principe théorique *On utilisera les mêmes notations que dans la partie précédent. L'idée est de parcourir aléatoirement le sous-groupe engendré par a, F (à l'image de sauts de kangourous) à l'aide de deux suites puis de calculer n à partir des indices où ces deux suites se sont rencontrées.*

Soit H un ensemble fini d'entiers dont le cardinal est du même ordre de grandeur que celui de F (l'ordre de H ne peut cependant être supérieur à celui F). Soit $(s,r) \in H^2$ tel que $(s-r) \wedge c = 1$ (avec $c = |F|$)[10] . Soit $f : F \to H$ une application surjective simple à calculer de tel sorte que les suites $U = (U_i)_{i\in\mathbb{N}}$ et $V = (V_i)_{i\in\mathbb{N}}$ définies par :

$$\begin{cases} U_0 = r * b \\ \forall i \in \mathbb{N}, U_{i+1} = f(U_i) * a + U_i \end{cases} \tag{8}$$

$$\begin{cases} V_0 = s * b \\ \forall i \in \mathbb{N}, V_{i+1} = f(V_i) * a + V_i \end{cases} \tag{9}$$

ait une évolution sur $\mathbb{N}$ qui a l'air aléatoire[11].

On calcule un à un les couples $(U_i; V_i)$ et on les met en mémoire puis on vérifie s'il n'existe pas un entier naturel j vérifiant :

$$j \leq i \text{ et } U_j = V_i \text{ et } U_i = V_j \tag{10}$$

On s'arrête dès qu'un j vérifiant ces propriétés a été trouvé. Quitte a renommer i et j, on peut supposer $U_i = V_j$. On en déduit :

$$r \times b + a \times (\sum_{t=0}^{i-1} f(U_t)) = s * b + a * (\sum_{t=0}^{j-1} f(V_t)) \tag{11}$$

On remplace par b par $n \times a$ et on en déduit, après avoir divisé des deux cotés par a :

$$r \times n + (\sum_{t=0}^{i-1} f(U_t)) = s \times n + (\sum_{t=0}^{j-1} f(V_t)) \tag{12}$$

10. On choisit H de tel façon à ce qu'un tel couple existe.
11. On admet l'existence d'une telle fonction.

$$(r-s)\times n = (\sum_{t=0}^{j-1} f(V_t)) - (\sum_{t=0}^{i-1} f(U_t)) \tag{13}$$

Or $(r-s)$ est inversible dans $\mathbb{Z}/c\mathbb{Z}$. On en déduit la valeur de n. De plus, on admet que l'on peut, étant donné le caractère aléatoire des suites U et V, dire que en moyenne il faut $\sqrt{c}$ calculs avant de trouver deux termes communs aux suites.

3.3 La méthode de Pohlig-Hellman

Cette fois, nous allons voir un algorithme qui sera non pas en $O(\sqrt{c})$ (c est toujours l'ordre du groupe dans lequel on travaille), mais en $O(\sqrt{p})$ où p est le plus grand facteur premier de de c. Nous ne verrons cette méthode que pour deux cas particuliers.

***Premier cas :** c* **est la puissance d'un nombre premier**

Appelons p ce nombre et β l'exposant ($p^\beta = c$). On cherchera alors à écrire n en base p. Rappelons de plus que n est défini modulo c (car F est un groupe cyclique d'ordre c).

$\exists(\alpha_0, ..., \alpha_{\beta-1}) \in (\llbracket 0, p-1 \rrbracket)^\beta, n = \sum_{k=0}^{\beta-1} \alpha_k * p^k$

On note pour tout $k \in \llbracket 0, p-1 \rrbracket$ T_k le sous-groupe de F engendré par $a_k = p^{\beta-k} \times a$. On remarque que ces sous-groupes sont d'ordre p^k. En effet : $p^k \times p^{\beta-k} \times a = c \times a = 0_F$ car F est d'ordre c.

Lemme 3.1. *Si on pose en recyclant les notations précédentes $b_k = p^{\beta-k} \times b$ alors [12] le logarithme m_k de b_k en base a_k est égal à $m_k = \sum_{i=0}^{k-1} \alpha_i * p^k$.*

Démonstration.

$$\begin{aligned}
b_k &= p^{\beta-k} * b = p^{\beta-k} * n * a = p^{\beta-k} * (\sum_{k=0}^{\beta-1} \alpha_k * p^k) * a \\
&= (\sum_{i=0}^{k-1} (\alpha_i * p^k) + (\sum_{i=k}^{\beta-1} alpha_i * p^k)) * p^{\beta-k} * a \\
&= \sum_{i=0}^{k-1} (\alpha_i * p^k) * a_k + \sum_{i=k}^{\beta-1} \alpha_i * p^k) * a_k \\
&= (\sum_{i=0}^{k-1} (\alpha_i * p^k) * a_k + (\sum_{i=0}^{\beta-k} \alpha_i * p^k) * p^k * a_k \\
&= (\sum_{i=0}^{k-1} (\alpha_i * p^k) * a_k + (\sum_{i=0}^{\beta-k} \alpha_i * p^k) * O_F \\
&= (\sum_{i=0}^{k-1} (\alpha_i * p^k) * a_k
\end{aligned}$$

□

La suite de la méthode est alors de calculer un à un les a_k par récurrence sur l. α_0 est le logarithme en base a_1 de $p^{\beta-1} * b$ (Simple à calculer s'il est assez petit). Soit $l \in \llbracket 0, \beta-2 \rrbracket$. Supposons que pour tout $k \in \llbracket 0, l \rrbracket$, on connaisse α_k. On déduit de lemme précédent que $m_{l+1} = m_l + \alpha_{l+1} * p^l$.

Lemme 3.2. *En réutilisant les mêmes notations, si on pose $z = b_{l+1} - m_l * a_{l+1}$ on en déduit que $z \in T_1$ et α_l est le logarithme de z en base a_1 .*

12. Rappel : *b* est l'élément de *F* dont on veut calculer le logarithme.

Démonstration.

$$\begin{aligned}
z &= b_{l+1} - m_{l+1} * a_{l+1} = p^{\beta-l-1} * b - m_l * p^{\beta-l-1} * a \\
z &= (n - m_l) * p^{\beta-1-l} * a = ((\sum_{i=0}^{\beta-1} \alpha_i * p^k) - (\sum_{i=0}^{l-1} \alpha_i * p^k)) * p^{\beta-1-l} * a \\
z &= (\sum_{i=(l)}^{\beta-1} \alpha_i * p^k) * p^{\beta-1-l} * a = \\
z &= (\sum_{i=(l)}^{\beta} \alpha_i * p^{k-l}) * p^{\beta-1} * a = (\sum_{i=(l)}^{\beta-1} \alpha_i * p^{k-l}) a_1 \\
z &= \alpha_l * p^{l-l} * a_1 + (\sum_{i=(l+1)}^{\beta-1} \alpha_i * p^{k-l}) a_1 = \alpha_l * a_1 + 0_F
\end{aligned}$$

□

On en déduit que $z \in T_1$ et qu'il a pour logarithme α_l dans la base a_1.

Ainsi, on en est réduit à calculer des logarithmes dans un groupe d'ordre p au lieu de les calculer dans un groupe d'ordre c, on aura donc un coût en p (éventuellement moins si on utilise un autre algorithme à ce moment là).

Regardons à présent un autre cas particulier.

***Second cas :* l'ordre de F est le produit de deux nombres premiers**

On suppose l'existence d'un couple $(p, q) \in \mathbb{P}^2$ tel que $c = p * q$. On appelle T_q le sous-groupe de F engendré par $p \times a$ (ce groupe est d'ordre q car $q \times (p \times a) = c \times a = 0_F$). On en déduit alors que $pb = n(pa)$ Si on trouve le logarithme n_q de pb en base pa on en déduit que $n_q \equiv n[q]$ et de manière analogue si on trouve le logarithme n_p de qb en base qa, on saura que $n_p \equiv n[p]$. Il ne restera alors plus qu'à résoudre le système (en sachant que n est inférieur à c) éventuellement en utilisant le théorème des restes chinois :

$$\begin{cases} n_q \equiv n[q] \\ n_p \equiv n[p] \end{cases}$$

Et on aura trouvé n en utilisant un algorithme qui aura un coût en $max(p, q)$ (éventuellement moins si on utilise un autre algorithme à ce moment là) [13].

Conclusion

A ce jour [14]*, une clé de 160 bits avec des courbes elliptiques assure le même niveau de sécurité qu'une clé de 1024 bits avec l'algorithme RSA. Par ailleurs, il est facile de fabriquer des puces spécialisées qui effectuent les calculs que nécessite la cryptographie par courbes elliptiques. On retrouve ainsi ces puces dans les cartes bancaires. La cryptographie par courbe elliptique, nouvelle application des mathématiques, de la géométrie à l'algèbre en passant par l'arithmétique, est une avancée majeure de la cryptographie. Encore jeune, elle est cependant entravée par une multitude de brevets qui empêchent son expansion et son étude. Elle n'a donc pas cette sûreté que de nombreux échecs des cryptologues pour casser le cryptosystème confère et révèle. Mais déjà, de nouveaux objets mathématiques sont récupérés pour créer de nouveaux algorithmes cryptographiques : les courbes hyperelliptiques.*

13. Cet algorithme permet de simplifier le problème dans certains cas particuliers et de le résoudre si on le couple avec un autre algorithme comme ceux ci-dessus.

14. 2011

1 Démonstrations supplémentaires

Formules d'addition *Soient $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ deux points de mathcal$E \setminus \{Q_\infty\}$ avec $x_1 \neq x_2$.*

La droite (P_1P_2) a pour équation $y = \lambda x + \mu$, avec $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ et $\mu = y_1 - \lambda x_1$ L'ensemble des intersections de cette droite est caractérisé par le système : $\begin{cases} y^2 = x^3 + ax + b \\ y = \lambda x + \mu \end{cases} \iff \begin{cases} \lambda^2 x^2 + \mu^2 + 2\lambda\mu x = x^3 + ax + b \\ y = \lambda x + \mu \end{cases}$

$\iff \begin{cases} x^3 - \lambda^2 x^2 + (a - 2\lambda\mu)x + (b - \mu^2) = 0 \\ y = \lambda x + \mu \end{cases}$

P_1 et P_2 sont déjà deux solutions de ce système, donc le polynôme $X^3 - \lambda^2 X^2 + (a - 2\lambda\mu)X + (b - \mu^2)$ admet, en plus de x_1 et x_2 une troisième racine x_3 qui vérifie :

$$x_1 + x_2 + x_3 = -(-\lambda^2) = \lambda^2$$

Donc les coordonnées (x_S, y_S) du point somme (symétrique de l'intersection qu'on vient de déterminer) sont :

$$\begin{cases} x_S = \lambda^2 - x_1 - x_2 \\ y_S = -\lambda x_S - \mu = -\lambda^3 + \lambda x_1 + \lambda x_2 - y_1 + \lambda x_1 = -\lambda^3 + 2\lambda x_1 + \lambda x_2 - y_1 \end{cases}$$

On retrouve les formules d'addition sur un corps quelconque. La preuve est du même goût dans le cas $P_1 = P_2$.

Théorème chinois *Pour $i \in \{1, \cdots, k\}$, les entiers n_i et $N_i = \frac{A}{n_i}$ sont premiers entre eux, car tous les n_j sont premiers entre eux. En vertu du théorème de Bezout, il existe deux entiers u_i et v_i tels que*

$$u_i n_i + v_i N_i = 1$$

On pose $\alpha_i = v_i N_i$. Alors :

Pour tout $j \neq i$, $\alpha_i \equiv 0[n_j]$et $\alpha_i \equiv 1[n_i]$. On note alors x' la somme de tous les $a_i\alpha_i$. $x' \equiv a_i[n_i]$, pour tout i de $\{1, \cdots, k\}$ Et le reste x de la division euclidienne de x' par A est bien, modulo A, la seule solution de ce système.

Polynôme annulateur de $\phi_{q|\mathcal{E}[n]}$ *On note $A = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ la matrice de ϕ_q dans la base (S,T). Le polynôme caractéristique de cette matrice, qui en est aussi un polynôme annulateur, est $X^2 - (a + d)X + det(A)$ En vertu la la proposition 1.8 , $det(A) \equiv deg(\phi_q) = q[n]$ De plus, d'après les propositions 1.8 et 1.9, on a (égalités entendues modulo n) :*

$$Card(Ker(\phi_q - id)) = det(A - I) = \begin{vmatrix} a - 1 & c \\ b & d - 1 \end{vmatrix} = ad - bc - a - d - 1$$

D'où $a + d \equiv t[n]$

Pour tout point P de n-torsion, on a $nP = P_\infty$ Donc $(a + d)\phi_q(P) = P_\infty + t\phi_q(P)$ et $det(A)P = qP$.

D'où la proposition.

2 Le programme de chiffrement et déchiffrement

```
type infini= Inf;;
(*on cree le point infini*)

type plan_proj= I of infini|Z of int*int;;
(*on cree l'ensemble des points *)

let modu a b= if a mod b>=0 then a mod b else b+(a mod b);;

type cEllip = {a:int;b:int ; p:int};;
(* on cree la courbe a valeur dans Z/pZ qui verifie y carre = xcube +ax +b  *)

let rec bezout a b =let r=a mod b and d=a / b in
        if r = 0 then (0, 1) else let (u, v) = bezout b r in (v, u - v * d);;


let inverse_modulo p a= match bezout p a with
          x,y when y>=0-> y mod p
        | x,y when y <0-> p+ (y mod p) ;;
(*ces deux fonctions donnent le couple (u,v) du theoreme de Bezout
```

```
d'un couple (a , b) et l'inverse de a modulo p*)

let addition_des_points courbe c d= match c,d with
        I Inf, d->d
        |c, I Inf-> c
        |Z(x,y),Z(z,t) when (modu x courbe.p )= (modu z courbe.p ) & ((y+t) mod courbe.p =0 )->I Inf
        |Z(x,y), Z(z,t) when (modu x courbe.p) <> (modu z courbe.p)->
                let g= (t-y)*(inverse_modulo courbe.p (z-x)) in
                        Z( modu (g*g-x-z) courbe.p , modu (2*g*x-g*g*g+g*z-y) courbe.p )
        |Z(x,y),Z(z,t) when (modu x courbe.p)= ( modu z courbe.p) & modu y courbe.p= modu t courbe.p ->
                let g=(3*x*x+ courbe.a )*(inverse_modulo courbe.p (2*y) ) in
                        Z( (modu (g*g-2*x) courbe.p) , modu (3*g*x - g*g*g -y) courbe.p);;
(*on additionne c et d dans une courbe courbe*)

let oppose c= match c with
  I Inf-> I Inf
  |Z(x,y)->Z(x,-y);;
  (*Donne l'oppose d'un point d'un courbe elliptique*)

let puissance n= let u=ref 1 in for i=1 to n do u:= 2* !u done; !u;;
(*donne 2 puissance n*)

let exposant n= let u= ref 0 in while puissance ( !u)<= n do incr u done; !u -1;;
(*calcule le logarithme en base 2 de n*)


let pluspetit u v c = match u, v with
   (f , I(Inf) ) , (g , I(Inf)) -> true
  | (f , I(Inf) ), (g , Z(x,y) ) -> true
  | ( f , Z(x,y) ) , (g , I(Inf) ) -> false
  | (f , Z(x,y)) , ( g , Z(z,t) ) when modu x c.p < modu z c.p-> true
  | (f , Z(x,y)) , ( g , Z(z,t) ) when modu x c.p > modu z c.p-> false
  | (f , Z(x,y)) , ( g , Z(z,t) ) -> modu y c.p <= modu t c.p;;
  (*f et g correspondent aux indices (qui n'interviennent pas dans cette
  relation d'ordre. *)

let tri_rapide t c =
        let n=Array.length t in
        let echange a b =
                let x = t.(a) in t.(a) <- t.(b) ; t.(b) <- x in
                let partition a b = let c1 = ref a and c2 = ref (b-1) and x = t.(b) in
                while (!c1) < b &  (pluspetit t.(!c1) x c ) do incr c1 done ;
                while (!c2) >= a & not ( pluspetit  t.(!c2)  x c ) do decr c2 done ;
        while (!c1) < (!c2) do
                echange (!c1) (!c2);
                incr c1 ; decr c2 ;
                while (!c1) < b & (pluspetit  t.(!c1)  x c ) do incr c1 done ;
                while (!c2) >= a & not ( pluspetit  t.(!c2)  x c ) do decr c2 done ;
        done ;
        if not ( pluspetit  t.(!c1)  x c ) then echange (!c1) b ;
        !c1 in
        let rec tri i j =
        if i < j then begin
                let x = (partition i j ) in
                tri i (x-1) ;
                tri (x+1) j
                end in
  tri 0 (n-1) ;;
(*on trie le tableau petit a petit de gauche a droite*)
let indice (x , y) =  x;;
(*renvoie l'indice d'un point ; utile pour l'algorithme
pas_de_bebe_pas_de_geant ci-dessous *)

let egal x y c = (pluspetit x y c) & (pluspetit y x c) ;;
(*permet de dire que deux points sont egaux  malgres leurs indices*)

let cherche x tab c = let i = ref 0 and j = ref ((Array.length tab) -1) in
        while (!j - !i)>0 do
```

```
88                 let k = (!i + !j)/2 in
89                 if (not(pluspetit x tab.(k) c )) then i:= k+1
90                 else j:= k
91         done;
92 if (egal tab.(!i) x c)
93         then !i
94         else (-1) ;;
95 (*On utilise des compteurs i et j pour faire une methode dichotomique*)
96
97 let doublement courbe b= addition_des_points courbe b b ;;
98 (*additionne deux memes points *)
99
100 let forme_non_adjacente tableau= let n= Array.length tableau in let resultat= Array.make (2*n+2) 0 in let j= ref
101   for i=0 to (n-1) do
102   if !indice=0 then
103   if tableau.(i)=0 then begin resultat.( !j)<- 0; incr j end
104   else incr indice
105   else if !indice=1 then
106   if tableau.(i)=1 then begin resultat.( !j)<- -1;
107                               incr j;
108                               resultat.( !j)<-0;
109                               incr j;
110                               indice:=2 end
111   else begin resultat.( !j)<- 1;
112              incr j;
113              resultat.( !j)<- 0;
114              incr j;
115              indice := 0 end
116   else if tableau.(i)=0 then indice:= !indice -1
117                         else begin resultat.( !j)<- 0;
118                              incr j end done;
119   if !indice=0 then () else begin resultat.( !j)<-1;incr j;resultat.( !j)<-0 end; resultat;;
120
121 let ecriture_binaire n=let resultat= Array.make ((exposant n) +1) 0  in  let k= ref n in
122   for i= exposant n downto 0 do if (puissance i)<= !k then begin resultat.(i)<-1;
123                                                                 k:= !k - (puissance i) end
124   done; resultat;;
125 (*donne l'ecriture en base deux d'un nombre*)
126
127 let binaire_non_adjacente n= forme_non_adjacente (ecriture_binaire n);;
128
129 let multip courbe n c= let r= ref (I Inf) and j= ref c and tab=binaire_non_adjacente n in
130 for i=0 to (Array.length tab)-1 do
131  if tab.(i)=1 then r:= addition_des_points courbe !r !j
132
133                                 else if tab.(i)= -1 then r:= addition_des_points courbe !r  (oppose !j);
134
135 j:= doublement courbe !j done;
136 !r;;
137 (* calcule de facon economique la multiplication d'un point par un
138  entier naturel *)
139
140 let pas_de_bebe_pas_de_geant e f n courbe =
141         let m = int_of_float (sqrt (float_of_int n) )
142         and tab = Array.make ((n/int_of_float (sqrt (float_of_int n) ) ) + 1) (0, I(Inf) )
143         and u = ref (I(Inf)) and q = ( multip courbe (int_of_float (sqrt (float_of_int n) )) e )
144         and p = ref 0 and k = ref 0 in
145         for i = 1 to (n/m) do
146                 u:= addition_des_points courbe q !u ; tab.(i) <- ( i , !u ) done ;
147                 u:= f ; p:= -1 ; tri_rapide tab courbe ;
148         while !p = -1 do
149                 p := (cherche (1 , !u) tab courbe) ;
150                 if !p <> (-1)   then p := m*(indice tab.(!p)) + !k
151                                       else begin incr k ;
152                                                  u:= addition_des_points courbe (oppose e) !u
153                                                  end
154         done ;
155   !p ;;
```

```
156   (*on cherche a calculer le logarithme en base e de f et n est l'ordre de e dans
157   la courbe courbe, tab correspond au tableau dans lequel on met les valeurs
158   calculees en faisant les pas de geants la deuxieme etape consiste a calculer un
159   a un les pas de bebes et a verifier si cela ne correspond pas avec une valeur
160   de tab.*)
161
162   let produit tableau i=
163         let u=ref 1 in
164         for j= 0 to ((Array.length tableau)-1) do
165                 if j=i then () else  u:= !u* tableau.(j)
166         done;
167         !u;;
168
169 let restes_chinois N A=
170         let s=ref 0 and k=Array.length A in
171           for i=0 to k-1 do
172                 s:= !s +A.(i)*(produit i N)*second( bezout N.(i) (produit i N) )
173           done; (!s mod (produit (k+1) N) ;;
174
175 let divise a b= float_of_int(b/a)= ((float_of int b) /. (float_of_int a))
176
177 let isprime p= let u= ref true in for i=0 to p-1 do if divise i p then u:=false; !u;;
178
179 let nextprime p= let u= ref (p+1) in while (isprime !u)= false do incr u done; !u;;
```

Table des matières

Bibliographie

[1] A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *Taher* ELGAMAL, *1985*

[2] Introduction élémentaire à la théorie des courbes elliptiques, *Marc* JOYE, *1995*

[3] Algorithmes de calcul de logarithmes discrets dans les corps finis, *Emmanuel* THOMÉ, *2003*

[4] Initiation à la cryptographie, *Gilles* DUBERTRET, *2002*

[5] `http://ser-info-02.ec-nantes.fr/users/info3/weblog/920cf/Algorithmes_discrets_utilises_en_cryptographie_asymetrique.html`, *2011*

[6] `http://fr.wikipedia.org`, *2011*

[7] Cryptographie théorique et pratique, *Douglas* STINSON

[8] Courbes elliptiques et cryptographie, *Hägler* MICHAEL

[9] Logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$, *Pierrick* GAUDRY

[10] Courbes elliptiques : une présentation élémentaire pour la cryptographie, *Philippe* GUILLOT

[11] `http://www.bibmath.net/dico/index.php3?action=affiche&quoi=./c/courbelliptique.html`, *2011*