Stage Liesse - Factorisation et cryptographie par les courbes elliptiques

14 mai 2019

Les exercices peuvent se résoudre avec SageMath, que l'on lance depuis un terminal en indiquant `sage -n jupyter`. SageMath est un logiciel de calcul mathématique gratuit et ouvert. Sa particularité est de s'appuyer sur des logiciels préexistants et d'utiliser Python comme langage de programmation. SageMath est à la fois une distribution de logiciels, une interface vers ceux-ci et une bibliothèque de méthodes mathématiques.

Exemple 1. L'algorithme ρ de Pollard peut être implanté comme suit.

```
def rho_pollard(n):
    x0 = Zmod(n).random_element()
    f = lambda z : z^2+1
    x=f(x0); y=f(x)
    while gcd(x-y,n)==1:
        x=f(x)
        y=f(f(y))
    return gcd(x-y,n)
```

Le langage est du Python, mais l'objet `Zmod(n)`, qui code $\mathbb{Z}/n\mathbb{Z}$, la méthode `random_element()` ou la fonction `gcd` viennent de SageMath.

Les algorithmes de factorisation présentés ici reposent sur le théorème suivant.

Théorème 2 (Lagrange)**.** *L'ordre d'un élément d'un groupe est un diviseur de l'ordre du groupe.*

En particulier, on retrouve le petit théorème de Fermat en se plaçant dans le groupe $\mathbb{F}_p^\times$:

$$\forall a \in \mathbb{F}_p^\times, \qquad a^{p-1} \equiv 1 \mod p.$$

Mise en route : l'algorithme $p-1$ de Pollard

L'algorithme $p-1$ est un algorithme inventé par John Pollard en 1974. Il repose sur l'idée suivante. Soit n un entier à factoriser et p l'un de ses facteurs premiers. Supposons que $p-1$ divise un certain entier m. Alors, par le petit théorème de Fermat, pour tout entier a, $a^m \equiv 1 \mod p$. Donc le $\text{pgcd}(a^m - 1, n)$ n'est pas trivial et peut fournir un diviseur de n.

Il nous rester à décider d'une valeur de m. Nous choisissons le ppcm de l'ensemble des entiers $\{1, 2, 3, \ldots, b\}$ pour une certaine borne b.

Definition 3. On dit qu'un entier x est *b-friable* lorsque tous ses diviseurs premiers sont inférieurs à b.
On dit qu'un entier x est *b-ultrafriable* si toutes les puissances d'un nombre premier qui le divisent sont inférieures à b.

Ce choix de m permet d'attraper les facteurs premiers p de n tels que $p-1$ est b-ultrafriable.

On obtient l'algorithme suivant :

Algorithme 1 : Méthode $p-1$ de Pollard

Entrées : Entier n, borne de friabilité b, liste P des entiers premiers $\leq b$

Sorties : Diviseur propre de n ou « échec : pas de facteur p avec $p-1$ qui soit b-ultrafriable »

```
a ← Elément aléatoire de ⟦2, n − 1⟧.
si g = a ∧ n > 1 alors
    retourner g
pour p ∈ P faire
    Calculer le plus grand entier α_p tel que p^{α_p} ≤ b
    pour j ∈ ⟦1, α_p⟧ faire
        a ← a^p mod n
        si 1 < g = (a − 1) ∧ n < n alors
            retourner g
retourner Échec
```

Bilan : Soit n un entier que nous cherchons à factoriser. On suppose pour simplifier que n se décompose en un produit $n = pq$ où p et q sont deux entiers premiers entre eux. Alors, par le lemme chinois,

$$(\mathbb{Z}/n\mathbb{Z})^\times = (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times.$$

La méthode $p-1$ de Pollard révèle le facteur p lorsque l'on a obtenu dans l'algorithme les éléments $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ et m tels que

$$a^m = 1 \mod p \qquad a^m \neq 1 \mod q. \tag{1}$$

Exemple 4 (Cas pathologique où la méthode ne fonctionne pas)**.** On fixe $p = 83$, $q = 107$ et $r = 149$. On pose $n = pqr = 1323269$. On choisit $b = 35$ comme borne de friabilité. On a

$$\begin{aligned} m &= \operatorname{ppcm}\{1,2,3,\ldots,35\} \\ &= 2^6 \cdot 3^4 \cdot 5^3 \cdot 7^2 \cdot 11^2 \cdot 13^2 \cdot 17^2 \cdot 19^2 \cdot 23^2 \cdot 29^2 \cdot 31^2 \\ &= 2896164734485379574016800 \end{aligned}$$

Mais

$$\begin{aligned}(\mathbb{Z}/n\mathbb{Z})^\times &= (\mathbb{Z}/p\mathbb{Z})^\times \oplus (\mathbb{Z}/q\mathbb{Z})^\times \oplus (\mathbb{Z}/r\mathbb{Z})^\times \\ &= (\mathbb{Z}/82\mathbb{Z}) \oplus (\mathbb{Z}/106\mathbb{Z}) \oplus (\mathbb{Z}/148\mathbb{Z})\end{aligned}$$

Notons que $p-1 = 2\cdot 41$, $q-1 = 2\cdot 53$ et $r-1 = 2^2\cdot 37$. Aussi, lorsqu'on calcule a^m, on obtient en général un élément d'ordre 41 modulo p, d'ordre 53 modulo q et d'ordre 37 modulo r. Il est donc très peu probable que le pgcd $(a^m - 1 \wedge n)$ soit non trivial. En vérité cette probabilité est $\frac{1}{41}\frac{52}{53}\frac{36}{37} + \frac{40}{41}\frac{1}{53}\frac{36}{37} + \frac{40}{41}\frac{52}{53}\frac{1}{37} + \frac{1}{41}\frac{1}{53}\frac{36}{37} + \frac{1}{41}\frac{52}{53}\frac{1}{37} + \frac{40}{41}\frac{1}{53}\frac{1}{37}$ ou environ 6.86%. Sauf cas exceptionnel, la méthode $p-1$ de Pollard échoue dans cet exemple.

Exercice 5. 1. Programmer l'algorithme $p-1$ de Pollard.

2. Vérifier sur plusieurs exemples qu'il fonctionne quand n possède des facteurs ultrafriables.
3. Vérifier sur un exemple qu'il fonctionne pas sinon.

Les courbes elliptiques

Nous nous plaçons sur un corps fini $\mathbb{F}_p$ avec p premier et > 3. Une courbe elliptique $\mathcal{E}$ est la réunion disjointe d'un point noté $0_\mathcal{E}$ (et dit point à l'infini) et l'ensemble des points affines (x, y) vérifiant

$$\mathcal{E}_{a,b} : \quad y^2 = x^3 + ax + b$$

pour des coefficients a et b fixés tels que $4a^3 + 27b^2 \neq 0 \in \mathbb{F}_p$. Nous noterons $\mathcal{E}_{a,b}(\mathbb{F}_p)$ cette courbe.

On peut définir une loi d'addition sur $\mathcal{E}_{a,b}(\mathbb{F}_p)$ par les règles suivantes :

— le neutre est le point à l'infini $0_\mathcal{E}$,

— l'opposé d'un point affine $P(x_P, y_P)$ est son symétrique $(x_P, -y_P)$ par rapport à l'axe (Ox),

— trois points alignés sont de somme nulle.

Ainsi, pour calculer la somme $P + Q$, on recherche la troisième intersection de la droite (PQ), ou de la tangente à $\mathcal{E}$ en P, avec $\mathcal{E}$ et on renvoie sa symétrie par rapport à l'axe (Ox) (voir figure 1).

Théorème 6. *Le groupe des points $\mathcal{E}_{a,b}(\mathbb{F}_p)$ est toujours de la forme*

$$\mathbb{Z}/r\mathbb{Z} \oplus \mathbb{Z}/s\mathbb{Z} \quad (\textit{avec } r|s).$$

De plus, on a la borne suivante, dite de Hasse,

$$p + 1 - 2\sqrt{p} \leq |\mathcal{E}_{a,b}(\mathbb{F}_p)| \leq p + 1 + 2\sqrt{p}$$

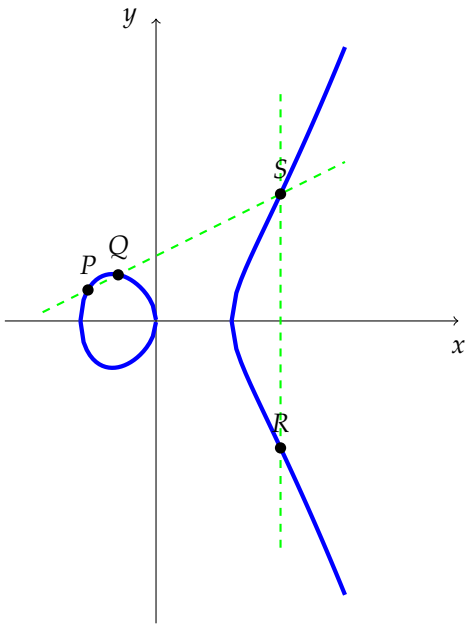

FIGURE 1: Somme de deux points : $P + Q = R$, $S = -R$.

Exemple 7. Dans un avis publié au JORF le 16 octobre 2011, l'ANSSI [1] recommande d'utiliser la courbe elliptique FRP256v1. Il s'agit de la courbe $\mathcal{E}_{a,b}(\mathbb{F}_p)$ où

1. Agence nationale de sécurité des systèmes informatiques

$$
\begin{aligned}
p &= \texttt{0xF1FD178C0B3AD58F10126DE8CE42435B3961ADBCABC8CA6DE8FCF353D86E9C03}\\
a &= \texttt{0xF1FD178C0B3AD58F10126DE8CE42435B3961ADBCABC8CA6DE8FCF353D86E9C00}\\
b &= \texttt{0xEE353FCA5428A9300D4ABA754A44C00FDFEC0C9AE4B1A1803075ED967B7BB73F}.
\end{aligned}
$$

Cette courbe a pour structure $\mathbb{Z}/n\mathbb{Z}$ avec

$$n = \texttt{0xF1FD178C0B3AD58F10126DE8CE42435B53DC67E140D2BF941FFDD459C6D655E1}.$$

qui est d'ailleurs premier. L'ANSSI préconise l'utilisation du point $G = (x_G, y_G)$ avec

$$
\begin{aligned}
x_G &= \texttt{0xB6B3D4C356C139EB31183D4749D423958C27D2DCAF98B70164C97A2DD98F5CFF}\\
y_G &= \texttt{0x6142E0F7C8B204911F9271F0F3ECEF8C2701C307E8E4C9E183115A1554062CFB}
\end{aligned}
$$

comme générateur du groupe abélien des points de la courbe.

Exemple 8. La courbe P_{256}, standardisée par le NIST [2], est souvent utilisée en cryptographie. Elle a pour équation

$$y^2 = x^3 - 3x + b \in \mathbb{F}_p[x, y]$$

où

2. Détails dans le document FIPS 186-4 du NIST (*cf.* page 91 de `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`).

$$
\begin{aligned}
p &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1\\
&= \texttt{0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff}\\
b &= \texttt{0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b}.
\end{aligned}
$$

Cette courbe a pour structure $\mathbb{Z}/n\mathbb{Z}$ avec

$n =$ `0xffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551`.

qui est d'ailleurs premier. Le NIST préconise l'utilisation du point $G = (x_G, y_G) \in P_{256d}(\mathbb{F}_p)$ avec

x_G = `0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296`

y_G = `0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5`

comme générateur.

Théorème 9. *Pour tout entier $m \in \left[p+1-2\sqrt{p}, p+1+2\sqrt{p}\right]$, il existe une courbe elliptique telle que $|\mathcal{E}_{a,b}(\mathbb{F}_p)| = m$.*
De plus, il existe $c > 0$ tel que si $\mathcal{S}$ est un sous-ensemble de $\left[p+1-2\sqrt{p}, p+1+2\sqrt{p}\right]$ de plus de 3 éléments, alors le nombre $N(\mathcal{S})$ de courbes elliptiques dont le cardinal est dans $\mathcal{S}$ satisfait

$$N(\mathcal{S}) > \frac{c|\mathcal{S}|p^{3/2}}{\log p}.$$

Application 10 (Diffie-Hellman)**.** Le protocole de Diffie-Hellman (1976) permet à deux parties, Alice et Bob, de convenir d'un secret commun sur un canal public.

Paramètres Alice et Bob s'accordent sur un groupe abélien et un générateur g.
Clés Secrètement, Alice et Bob choisissent chacun un entier a et b (respectivement).
Échange Alice communique $g_a = a \cdot g$ à Bob, Bob communique $g_b = b \cdot g$ à Alice.
Secret commun Le secret commun est $(ab) \cdot g = b \cdot (g_a) = a \cdot (g_b)$.

Longtemps, ce protocole a été utilisé dans le groupe multiplicatif d'un corps fini $\mathbb{F}_p^\times$. Il est aujourd'hui utilisé plutôt dans le groupe des points d'une courbe elliptique.

Exercice 11. Dans cet exercice, on utilisera la courbe $\mathcal{E}_{5,20}(\mathbb{F}_{97})$ munie du point $G = (86, 63)$. Ce point est d'ordre $n = 113$, ce qui est aussi le cardinal de la courbe.

1. Tirer au hasard un entier a entre 1 et $n-1$. Calculer le point $a \cdot G = (x_A, y_A)$
2. Échanger l'abscisse de x_A avec son voisin. On notera x_B l'abscisse reçue.
3. Trouver un point de la courbe $B = (x_B, y_B)$ et calculer le point $C = a \cdot B$.

4. Vérifier avec son voisin que l'abscisse coïncide.

Application 12 (Elliptic curve digital signature algorithm)**.** Le protocole ECDSA permet à deux parties d'authentifier un message que la première, disons Alice, envoie à la seconde, disons à Bob.

Paramètres Alice et Bob s'accordent sur une courbe elliptique $\mathcal{E}$ et un point G d'ordre premier n.

Clés Secrètement, Alice fixe un entier s (clé privée) et publie le point $Q = sG$.

Signature Pour signer un message m, Alice

— choisit un entier aléatoire k entre 1 et $n-1$,

— calcule le point $K = (x, \_) = k \cdot G$

— calcule $y = k^{-1}(m + sx) \mod n$

Alice envoie (x, y) comme signature du message transmis m.

Vérification Bob calcule le point $my^{-1}G + xy^{-1}Q$ (l'inverse de y est modulo n). L'abscisse doit être égale à x.

Une signature d'Alice est toujours vérifiée car

$$my^{-1}G + xy^{-1}Q = (my^{-1} + xy^{-1}s)G = (m + xs)y^{-1}G = kG.$$

En pratique, on utilise le haché du message m et non m directement.

Exercice 13. Dans cet exercice on utilisera la courbe $\mathcal{E}_{-3,10}(\mathbb{F}_{41})$ munie du point $G = (38, 22)$ d'ordre 41 également. Alice envoie à Bob le message `"Hello world!"`, de haché $m =$ `hash("Hello world !")` $= -874365580694288015$. Elle a publié comme clé plublique le point $(39, 34)$. Elle signe son message par le couple

$$\begin{aligned} x &= 34 \\ y &= 4 \end{aligned}$$

Son message est-il authentique ?

Application 14. Le cryptosystème de Menezes & Vanstone est un protocole (aujourd'hui déconseillé) de cryptographie asymétrique qui permet à Alice de recevoir un message de Bob

Génération de clés Alice fixe une courbe elliptique $\mathcal{E}(\mathbb{F}_p)$, des points G, H et un entier m tel que $mG = H$.

Clé privée L'entier m est sa clé secrète.

Clé publique Elle publie la courbe et les points G et H.

Chiffrement Bob peut envoyer des messages $(x_1, x_2) \in \mathbb{F}_p^2$. Pour cela, il choisit un entier aléatoire k. Il calcule $Y = kG$. Il calcule $(c_1, c_2) = kH$. Il calcule enfin $s_1 = c_1 x_1 \mod p$ et $s_2 = c_2 x_2 \mod p$. Il envoie à Alice le triplet $(Y, s_1, s_1) \in \mathcal{E} \times \mathbb{F}_p \times \mathbb{F}_p$.

Déchiffrement Alice calcule $mY = mkG = kH = (c_1, c_2)$. À partir de c_1 et de c_2, elle retrouve le message avec $m_1 = c_1^{-1} s_1 \mod p$ et $m_2 = c_2^{-1} s_2 \mod p$.

Exercice 15. On donne la cubique $\mathcal{E}$ défine sur $\mathbb{F}_{13}$ par

$$y^2 = x^3 + 4x + 7.$$

1. Vérifier qu'il s'agit d'une courbe elliptique et que $G = (4, 3)$ appartient à la courbe.
2. Alice et Bob utilisent le cryptosystème de Menezes et Vanstone pour communiquer, avec le point G et la clé secrète $m = 9$ (notation de l'exemple 14).
   (a) Calculer le point H de la clé publique d'Alice.
   (b) Bob veut envoyer le message $(2, 7)$ à Alice. Calculer son chiffré.
   (c) Alice reçoit le message (Y, s_1, s_2) de la part de Bob avec $Y = (1, 5)$, $s_1 = 7$ et $s_2 = 2$. Retrouver le clair.

L' algorithme ECM

L'algorithme ECM (Elliptic Curve Method) a été inventé en 1985 par Lenstra. Il revisite la méthode $p - 1$ de Pollard.

Soit $\mathcal{E}_{a,b} : y^2 = x^2 + ax + b$ une cubique sur $\mathbb{Z}/n\mathbb{Z}$ (avec $(4a^3 + 27b^2) \wedge n = 1$). Grâce au lemme chinois, on a une application naturelle

$$\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z}) \to \mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z}) \times \mathcal{E}_{a,b}(\mathbb{Z}/q\mathbb{Z})$$

entre les points affines de ces courbes (que l'on étend au point à l'infini par $0_{\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})} \mapsto (0_{\mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z})}, 0_{\mathcal{E}_{a,b}(\mathbb{Z}/q\mathbb{Z})})$). La methode ECM consiste à trouver un point $A \in \mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})$ et un entier m tel que

$$\begin{cases} m \cdot A = 0_{\mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z})} & \text{dans } \mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z}) \\ m \cdot A \neq 0_{\mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z})} & \text{dans } \mathcal{E}_{a,b}(\mathbb{Z}/q\mathbb{Z}) \end{cases}. \tag{2}$$

(comparer avec l'équation (1))

Dans cette situation, le calcul de $m \cdot A$ dans $\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})$ échoue. En effet, tous les calculs restent valable modulo p et montrent que $m \cdot A$ est le point $0_{\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})}$. Mais tous les calculs restent valable modulo q aussi et montrent que $m \cdot A$ ne peut pas être le point $0_{\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})}$. Calculer $m \cdot A$ dans $\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})$ revient à faire une suite d'opérations arithmétiques dans $\mathbb{Z}/n\mathbb{Z}$: elles peuvent toutes êtres réalisées sauf lorsqu'on chercher à diviser par un élément x de $\mathbb{Z}/n\mathbb{Z}$ non inversible. Mais dans ce cas, le pgcd de x et de n est non trivial et fournit un facteur de n.

En faisant le choix $m = \text{ppcm}\{1, 2, 3, \ldots, b\}$, on obtient l'algorithme suivant :

Algorithme 2 : Factorisation par courbes elliptiques de Lenstra (ECM)

Entrées : Entier n premier avec 6 et qui n'est pas une puissance d'un nombre premier. Borne B.
Sorties : Diviseur non-trivial de n ou « échec »

Tirer au hasard $a, x_0, y_0 \in [\![0, n-1]\!]$
$b \leftarrow y_0^2 - x_0^3 - ax_0$
$g \leftarrow \text{pgcd}(4a^3 + 27b^2, n)$
si $1 < g < n$ **alors**
    retourner g
sinon si $g = n$ **alors**
    retourner *Echec*
$\mathcal{E} \leftarrow \mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})$
$A \leftarrow (x_0, y_0)$
pour p *premier* $\leq B$ **faire**
    Trouver e maximal tel que $p^e \leq B$
    essayer
        $A \leftarrow p^e \cdot A$ dans $\mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z})$
    excepté *Erreur d'inversion*
        retourner *Facteur de n obtenu*
retourner *Echec*

Remarque 16. La complexité de ECM est liée à la taille du plus petit diviseur de n et non pas à n lui même (voir dernière section). Aussi, la méthode est particulièrement efficace pour éliminer les facteurs de n de taille moyenne. Elle peut être utilisée en complément à d'autres méthodes pour retirer de n les facteurs de petite et moyenne taille, le cas le plus défavorable étant le cas $n = pq$ avec p et q premiers de même taille.

Exemple 17. Cherchons à factoriser $n = 3397$ par la méthode ECM.

On tire au hasard la cubique d'équation

$$y^2 = x^3 + 4x + 25$$

et le point $P = (3, -8)$.

— On commence par calculer $2P$ selon les formules habituelles. On calcule la pente de la tangente en P qui vaut

$$\lambda = \frac{3x_P^2 + a}{2y_P} = \frac{31}{-16} = 635 \mod n.$$

On en tire le point $2P$ de coordonnées :

$$x_{2P} = \lambda^2 - 2x_P = 2373 \qquad y_{2P} = -y_P + \lambda(x_P - x_{2P}) = 3326.$$

— On calcule ensuite $3P = 2P + P$. La pente de la droite $(P, 2P)$ devrait être

$$\lambda = \frac{y_{2P} - y_P}{x_{2P} - x_P} = \frac{3326 - (-8)}{2373 - 3} = \frac{3334}{2370}.$$

Mais 2370 n'est pas inversible modulo n. En effet, le pgcd $2370 \wedge 3397$ vaut 79. Le calcul de λ échoue et révèle, ce faisant, un facteur de n, à savoir 79.

Exercice 18. On considère les paramètres

$$a = b = 4, \quad p = 47, \quad q = 59 \quad \text{et} \quad P = (1, 3).$$

On pose $n = pq$.

1. Quel est l'ordre ω_p et ω_q de P dans $\mathcal{E}_{a,b}(\mathbb{F}_p)$ et dans $\mathcal{E}_{a,b}(\mathbb{F}_q)$?
2. Soit ϕ_p la projection modulo p, à savoir $\phi_p : \mathcal{E}_{a,b}(\mathbb{Z}/n\mathbb{Z}) \to \mathcal{E}_{a,b}(\mathbb{F}_p)$. Quels sont les antécédants du point $(0 : 1 : 0)$ par ϕ_p ?
3. Que se passe-t-il si on cherche à calculer $\omega_p P$ et $\omega_q P$ dans $\mathcal{E}(\mathbb{Z}/n\mathbb{Z})$?

Afin d'implémenter l'algorithme ECM avec Sage, nous voulons redéfinir l'algorithme de division afin que le calcul de x/y dans $\mathbb{Z}/n\mathbb{Z}$ quand y n'est pas inversible lève une erreur qui indique le facteur de n que l'on a découvert. On définit

```
class FoundFactor(Exception):
    def __init__(self, value):
        self.value = value
    def __str__(self):
        return repr(self.value)
```

Exercice 19. Ecrire une fonction `division` qui prend en entrée deux éléments x et y de $\mathbb{Z}/n\mathbb{Z}$ et renvoie le quotient x/y lorsque celui-ci existe, une erreur *ZeroDivisionError* lorsque y est nul dans $\mathbb{Z}/n\mathbb{Z}$ et une erreur *FoundFactor* lorsque y est non nul mais n'est pas inversible dans $\mathbb{Z}/n\mathbb{Z}$.

Exercice 20. On désigne par (x_M, y_M) les coordonnées affines du point M.

1. Montrer que l'addition $R = P + Q$ de deux points distincts P et Q de la courbe $\mathcal{E}_{a,b}$ conduit aux formules

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}, \quad x_R = \lambda^2 - x_P - x_Q, \quad y_R = -y_P + \lambda(x_P - x_R)$$

   où λ est la pente de la droite (PQ).

2. Montrer que la duplication $R = 2P$ d'un point P de $\mathcal{E}_{a,b}$ conduit en général aux formules

$$\lambda = \frac{3x_P^2 + a}{2y_P}, \quad x_R = \lambda^2 - 2x_P, \quad y_R = -y_P + \lambda(x_P - x_R)$$

   où λ est la pente de la tangente à $\mathcal{E}$ en P. Que se passe-t-il si $y_P = 0$?

3. Comment faire pour calculer l'ensemble des points de 2-torsion de $\mathcal{E}_{a,b}$? Quelle structure du sous-groupe $\mathcal{E}[2]$ de 2-torsion peut-on obtenir ?

4. Écrire une fonction `addition` qui prend en entrée deux points P et Q d'une courbe elliptique et renvoie leur somme $P + Q$.

5. Comparer avec la méthode native de SAGE. On pourra prendre par exemple la courbe $\mathcal{E}_{a,1}(\mathbb{F}_q)$ et différents multiples du point P_0 de coordonnées $(0, 1)$.

Exercice 21. 1. Ecrire une fonction `multiplication` qui prend en entrée un point P d'une courbe elliptique et un scalaire λ et renvoie le produit $\lambda \cdot P$.
   On pourra utiliser une méthode d'exponentiation rapide dont le principe est le suivant :

$$\forall \lambda \in \mathbb{N}, \quad \lambda \cdot P = \begin{cases} 0_{\mathcal{E}_{a,b}} & \text{si} \quad \lambda = 0 \\ P & \text{si} \quad \lambda = 1 \\ \dfrac{\lambda}{2} \cdot (P + P) & \text{si} \quad \lambda \text{ est pair} \\ P + \dfrac{\lambda - 1}{2} \cdot (P + P) & \text{si} \quad \lambda \text{ est impair} \end{cases}$$

2. Comparer avec la méthode native de SAGE.

Exercice 22. 1. Écrire une fonction `ECM` qui implante l'algorithme ECM de Lenstra.

2. Tester votre fonction sur différents exemples.

Exercice 23. On pose $n = 42857766101$. Quelle est la plus petite valeur de la borne de friabilité B telle que l'algorithme factorise n avec les paramètres $a = b = 4$ et $P = (1, 3)$? [Indication : factoriser les exposants des groupes $\mathcal{E}(\mathbb{F}_p)$ et $\mathcal{E}(\mathbb{F}_q)$ où p et q sont les deux diviseurs premiers de n.]

Remarque sur la complexité d'ECM

Notons $\psi(x,b)$ le nombre d'entiers b-friables inférieur à x. La probabilité qu'un entier $n \leq x$ soit b-friable est donc $\frac{1}{x}\psi(x,b)$. Posons

$$u = \frac{\log x}{\log b}.$$

Le théorème de Canfield-Erdös-Pomerance donne l'estimation suivante

$$\frac{1}{x}\psi(x,b) = u^{-u+o(u)}$$

quand $u, x \to \infty$, à condition que $u < (1-\epsilon)\log x/\log\log x$ pour un certain $\epsilon > 0$.

L'analyse de l'algorithme ECM et l'optimisation de la borne de friabilité est compliquée et se base sur des heuristiques en l'absence de meilleurs résultats mathématiques. Supposons pour simplifier que n est le produit deux nombres premiers $n = pq$. Il nous faut d'une part estimer le coût de l'algorithme par courbe elliptique et d'autre part estimer la probabilité de succès pour une courbe.

Le coût de l'algorithme est proportionnel au nombre de passage dans la boucle `for`. Notons $\pi(B)$ le nombre d'entiers premiers inférieurs à B. Une estimation classique donne $\pi(B) \sim B/\log B$. On peut majorer par $\log B$ la complexité de chaque étape de la boucle. Nous majorons le coût de l'algorithme par courbe elliptique par $O(B)$.

L'algorithme réussit quand on trouve un point A dont l'ordre soit B-friable dans $\mathcal{E}_{a,b}(\mathbb{Z}/p\mathbb{Z})$ et ne le soit pas dans $\mathcal{E}_{a,b}(\mathbb{Z}/q\mathbb{Z})$. En vérité, la deuxième condition est très improbable lorsque la première est satisfaite et peut être oubliée pour notre estimation. Notons $\mathcal{S}$ l'ensemble des nombres friables compris entre $p+1-2\sqrt{2}$ et $p+1+2\sqrt{2}$. Il y a p^2 choix pour les valeurs de a et de b, d'où une probabilité

$$\Omega\left(\frac{|\mathcal{S}| \cdot \frac{p^{3/2}}{\log p}}{p^2}\right) = \Omega\left(\frac{|\mathcal{S}|}{\sqrt{p}} \cdot \frac{1}{\log p}\right)$$

Nous recourons à l'heuristique suivante : la probabilié qu'un entier compris dans l'intervalle de Hasse soit B-friable comparable à celle qu'un entier $\leq p$ soit B-friable, soit essentiellement $u^{-u+o(u)}$ avec $u = \frac{\log p}{\log B}$. En ignorant les termes en $\log p$, qui sont négligeable devant $u^{o(u)}$, on obtient une probabilité de l'ordre de u^{-u}.

Puisque $B = p^{1/u}$, le coût de l'algorithme ECM itéré sur plusieurs courbes se simplifie ainsi en

$$p^{1/u}u^u.$$

dont le minimum (à des facteurs asymptotiquement négligeables près) est atteint pour $u = \sqrt{2\log p/\log\log p}$.

La borne de friabilité optimale est donc

$$B = \exp\left(\left(1/\sqrt{2} + o(1)\right)\sqrt{\log p \log\log p}\right) = L_p(1/2, 1/\sqrt{2}).$$

En pratique, on ne connait par la taille du facteur p que l'on cherche, mais on peut la deviner a priori pour fixer B et doubler cette taille périodiquement pour parvenir à une factorisation. Il est remarquable que le coût de l'algorithme ne dépende que de la taille (estimée) du facteur premier que l'on cherche et non pas de celle de l'entier que l'on veut factoriser.

Corrigés de certains exercices

Solution 5. 1. Une implantation de la méthode $p-1$ de Pollard :

```
def Pollard(n,b):
    Zn=Zmod(n)
    a=Zn.random_element()
    g = gcd(ZZ(a),ZZ(n))
    if g>1:
        return g
    for p in primes(b):
        c=1
        while(c<b):
            c*=p
            a=mod(a^p ,n)
            g=gcd(ZZ(a-1),ZZ(n))
            if g>1:
                return g
    return false
```

2.

3. On peut fabriquer un entier composé réfractaire comme suit :

```
b=30
L = [p for p in primes(b+1,10*b) if max((p-1).prime_divisors()) >= b]
p=L[0]
q=L[1]
n=p*q
print factor(p-1), factor(q-1)
Pollard(n,b)
```

(Les entiers $p-1$ et $q-1$ sont bien b-ultrafriables !)

Solution 11. On peut faire :

```
p=97;Fp=GF(p)
alpha=Fp(5);beta=Fp(20)
E=EllipticCurve(Fp,[alpha,beta])
G=E(86,63)
n=G.order()
```

```
a=ZZ(Zmod(n).random_element())
b=ZZ(Zmod(n).random_element())
A=a*G
B=b*G
xA=A[0]; xB=(b*G)[0]
yB = sqrt(xB^3+alpha*xB+beta)
B_alice = E(xB,yB)
(a*B_alice)[0]
```

Solution 13. On peut coder l'ensemble par les instructions suivantes.

```
p = 41
a=-3
b = 10
E =  EllipticCurve(GF(p),[a,b])
xG = 38
yG = 22
G = E(xG,yG)
n = 41
```

```
def ECDSA_keygen():
    s= ZZ(Zmod(n).random_element())
    return (s,s*G)

def ECDSA_sign(mm,sk):
    m = hash(mm)
    k = ZZ(Zmod(n).random_element())
    print "k=",k
    print "kG=", k*G
    (x,_,_) = k*G
    y = mod(k,n)^(-1) * mod(m+sk*x,n)
    return (ZZ(x),ZZ(y))

def ECDSA_check(mm,sign,pk):
    m = hash(mm)
    x = sign[0]
    y = sign[1]
    yy = ZZ(mod(y,n)^(-1))
    return (m*yy*G+x*yy*pk)[0]-x ==0

(secret_key, public_key) = ECDSA_keygen()
message = "Hello world!"
signature = ECDSA_sign(message,secret_key)
check = ECDSA_check(message,signature,public_key)
print check
```

Solution 15. 1. Le discriminant est non nul.

2.(a) $H = (11, 11)$.

(b) $k = 5$, donc $Y = kG = (11, 2)$. $kH = (1, 5)$. Donc $s_1 = x_1 = 2$ et $s_2 = 5x_2 = 9$.

(c) Le clair est $(5, 8)$.

Solution 18. 1. On a $\omega_p = 3$ et $\omega_q = 4$.

```
p=47
E = EllipticCurve(Zmod(p),[4,4])
A = E(1,3)
A.order()
```

2. Il n'y a qu'un seul point : $(0 : 1 : 0)$

3. On doit avoir $\phi_p(\omega_p P) = (0 : 1 : 0)$ et $\phi_q(\omega_p P) \neq (0 : 1 : 0)$, ce qui implique (par la question précdente) que $\omega_p P = (0 : 1 : 0)$ et que $\omega_p P \neq (0 : 1 : 0)$. Contradiction. Calculer $\omega_p P$ déclenche une erreur.

```
p=47
q=59
n=p*q
E = EllipticCurve(Zmod(n),[4,4])
A = E(1,3)
3*A
4*A
```

Solution 19. On peut faire

```
def division(x,y):
    n = y.modulus()
    if ZZ(n).divides(ZZ(y)):
        raise ZeroDivisionError
    elif gcd(ZZ(y),ZZ(n))>1:
        raise FoundFactor(gcd(ZZ(y),ZZ(n)))
    else:
        return mod(x,n)/mod(y,n)
```

Solution 20. 1. L'équation de la droite (PQ) est clairement

$$y - y_P = \lambda(x - x_P)$$

Pour trouver S, on doit donc résoudre le système

$$\begin{cases} y - y_P = \lambda(x - x_P) \\ y^2 = x^3 + ax + b \end{cases}$$

qui conduit à l'équation

$$x^3 - \lambda^2 x^2 + [\cdots] = 0.$$

Mais

$$x^3 - \lambda^2 x^2 + [\cdots] = (x - x_P)(x - x_Q)(x - x_S).$$

Donc par identification

$$x_S = \lambda^2 - x_P - x_Q$$

La conclusion suit immédiatement.

2. On se souvient que la tangente en x_P a pour équation

$$\frac{\partial f}{\partial x}(P) \cdot (x - x_P) + \frac{\partial f}{\partial y}(P) \cdot (y - y_P) = 0$$

ce qui fournit bien la pente $\lambda = \frac{3x_P^2 + a}{2y_P}$. Le reste du calcul est identique la question précédente. Si $y_P = 0$, on obtient une tangente verticale qui intersecte la courbe en $0_\mathcal{E}$. Donc $2P = 0_\mathcal{E}$.

3. Le sous-groupe $\mathcal{E}[2]$ de 2-torsion correspond aux points P tels que $2P = 0_\mathcal{E}$. On retrouve donc le point $0_\mathcal{E}$ et l'ensemble des points $(\sigma, 0)$ tels que $\sigma^3 + a\sigma + b = 0$. Il y en a 0, 1 ou 3, ce qui correspond aux groupes trivial, $\mathbb{Z}/2\mathbb{Z}$ ou $(\mathbb{Z}/2\mathbb{Z})^2$.

4. Algorithme d'addition

```
def addition(P,Q):
    if P[2]==0:
        return Q
    elif Q[2]==0:
        return P
    elif P[0]==Q[0] and P[1]==-Q[1]:
        return P.curve()(0)
    else:
        if P==Q:
            lamb = division( (3*P[0]^2+P.curve().a4()) , (2*P[1]))
        else:
            lamb = division(Q[1]-P[1], Q[0]-P[0])
        x = lamb^2-P[0]-Q[0]
        y = -P[1]+lamb*(P[0]-x)
        return P.curve()(x,y)
```

Solution 21. 1. Multiplication

```
def multiplication(lamb,P):
    E = P.curve()
    if lamb == 0:
        return E(0)
    elif lamb == 1:
        return P
    elif lamb%2 == 0:
        return multiplication(lamb/2,addition(P,P))
    else:
        Q = multiplication(ZZ((lamb-1)/2),addition(P,P))
        return addition(P,Q)
```

2.

Solution 22. 1. Algorithme ECM

```
def ECM(n,B):
    Zn = Zmod(n)
    a = Zn.random_element()
    x0 = Zn.random_element()
    y0 = Zn.random_element()
    b = y0^2-x0^3-a*x0
    g = gcd(ZZ(4*a^3+27*b^2),ZZ(n))
    if g==n:
        return False
    elif g>1:
        return g
    E = EllipticCurve(Zn, [a,b])
    A = E(x0,y0)
    for p in primes(B):
        c=1
        while (c<B):
            c*=p
            try:
                A = multiplication(p,A)
            except FoundFactor as FF:
                print 'Facteur trouvÃ©l' :'
                return FF.value
    return False
```

2.

Solution 23. On peut faire

```
n = 42857766101
print factor(n)
p=63029
q=679969
Ep = EllipticCurve(GF(p),[4,4])
Eq = EllipticCurve(GF(q),[4,4])
Ep.abelian_group(), Eq.abelian_group()
Ep.cardinality().factor() , Eq.cardinality().factor()
```

On remarque alors que les deux groupes sont des groupes cycliques $\mathcal{E}(\mathbb{F}_p) = \mathbb{Z}/62896\mathbb{Z}$ et $\mathcal{E}(\mathbb{F}_q) = \mathbb{Z}/678501\mathbb{Z}$ et que, de plus,

$$62896 = 2^4 \cdot 3931 \qquad \text{et} \qquad 678501 = 3^2 \cdot 75389.$$

Comme le plus petit des plus grands facteurs de ces ordres est 3931, il faut donc aller génériquement jusqu'à la borne $B = 3931$ (incluse).

Il se trouve que ici P est d'ordre $2^3 \cdot 3931$ (faire `Ep(Pn).order().factor()` pour s'en apercevoir). On peut confirmer la borne trouvée en faisant

```
n = 42857766101
En = EllipticCurve(Integers(n),[4,4])
Pn = En(1,3)

Q= Pn
for r in primes(3932):
    for i in range(log(n,r)+1):
        Q= r*Q
```

Remarque : le hasard aurait pu nous donner $P' = (21091706543, 38501272299)$ au lieu de P et qui est d'ordre 8 dans dans $\mathcal{E}(\mathbb{F}_p)$. Dans ce cas, calculer $8P'$ dans $\mathcal{E}(\mathbb{Z}/n\mathbb{Z})$ provoque déjà une erreur.