Courbes elliptiques sur un anneau et applications cryptographiques

Marie Virat

UNIVERSITE DE NICE-SOPHIA ANTIPOLIS - UFR Sciences
Ecole Doctorale de Sciences Fondamentales Appliquées

THESE

pour obtenir le titre de
Docteur en Sciences
de l'UNIVERSITE de Nice-Sophia Antipolis

Discipline : MATHEMATIQUES

présentée et soutenue par
Marie VIRAT

Courbes elliptiques sur un anneau et applications cryptographiques

Thèse dirigée par André HIRSCHOWITZ et co-dirigée par Bruno MARTIN
soutenue le 17 avril 2009

Jury :

M. L. Evain	Maître de Conférences - HDR	Rapporteur
M. M. Girault	Expert indépendant - HDR	
M. A. Hirschowitz	Professeur	Directeur
M. D. Kohel	Professeur	
M. F. Leprévost	Professeur	Rapporteur
M. B. Martin	Professeur	Co-directeur
M. P. Véron	Maître de Conférences	

A ma mère,

à Heïkel,

à Bruno,

à Marc Birebent,

à mon grand-père,

Remerciements

Voici paraît-il les pages les plus agréables et les plus délicates à écrire, celles que le lecteur découvre en premier et que l'auteur écrit en dernier.

Pour une thèse débutée il y a bientôt sept ans de ça, je pourrais remercier la terre entière et ce serait bien mon genre. Donnons-nous donc un peu de contenance.

Eh bien, pour commencer, si vous lisez ces quelques lignes je vous en remercie, et je vous invite à lire les suivantes ! Et si vous avez des remarques, des corrections sur ce document, si vous avez besoin d'éclaircissements ou si vous souhaitez me donner votre éclairage, n'hésitez pas à me contacter à cette adresse Marie.Virat@gmail.com.

Ensuite, la vie de tous les jours. En sept ans, il peut s'en passer bien des choses et lorsqu'on choisit de faire deux enfants pendant cette période, la gestion du quotidien devient capitale pour l'épanouissement professionnel et personnel.
A ce sujet, je me dois de commencer par remercier Heïkel, le père de mes enfants, qui a osé relever ce pari, avec toute la confiance que cela nécessite malgré tous les doutes que cela comporte. Sans cette audace, ma (nos) vie(s) n'aurai(en)t pas pris tout son (leur) sens. Tu restes la clef de voûte de mon équilibre.
Et puis dans la gestion du quotidien, le moindre geste de soutien devient un cadeau du ciel, un petit miracle de l'instant. A commencer par mon frère Pierre, Valérie, Colette, Aïcha, Eddy, Lætitia, Barbara, Charlotte, le personnel de "La ritournelle", Sabah, Pierre, merci de votre présence protectrice et chaleureuse. Sans oublier tous ceux et celles qui me proposent leur aide régulièrement, sans vous tous les choses seraient bien plus difficiles.

Et puis, il y a la vie professionnelle. Et là aussi, même si les marques de soutien ne s'expriment pas de la même façon d'un individu à l'autre, elles n'en restent pas moins cruciales.

Il y a ceux qui vous font avancer par leur rigueur, leur constance, leur bienveillance, leurs remarques, leurs questions, leur pédagogie, leurs compétences, leur curiosité, leurs regards transversaux, leur honnnêteté, leur expérience : à commencer par mes deux directeurs André Hirschowitz et Bruno Martin ; puis ceux rencontrés en chemin Fabien Herbaut, Fabien Laguillaumie, Damien Vergnaud, Pascal Véron, Damien Stehlé, Marc Girault, Marco Maggesi, Christopher Wolf, René Schoof, Louis Granboulan, Christian Pauly, Guillaume Chèze, Patrick Solé, Mireille Fouquet, Pierrick Gaudry ; et pour finir ceux qui participent aux derniers mètres de ce doctorat et qui n'en restent pas moins capitaux les membres du jury bien sûr, David Kohel, Franck Leprévost et Laurent Evain.

Il y aussi ceux qui ne publient pas et qui nous soutiennent tous. Dans le monde du spectacle,

on les appelle les gens de l'ombre et on les remercie à la fin du show. En vérité, leur pouvoir est immense ; leur comportement et leurs compétences peuvent rendre les choses tellement plus simples. Merci à Rosalba, Jeanine, Bernard, Jean-Marc, Isabelle, Jean-Louis, Marie-Claude, Jean-Paul, Claudine, Fernande, Marie-France, Valérie, Julien, votre bienveillance souffle de l'air frais.
Les chefs d'orchestre eux aussi ont su me soutenir : Michel Miniconi, Frédéric Poupaud, Charles Walter, Philippe Maisonobe et Michel Merle. Merci d'avoir laissé vos portes ouvertes.

Enfin, il y a ceux qui sont dans les bureaux d'à côté, ou de franchement plus loin. Mais à l'ère de la mondialisation, tous les bureaux sont à côté.
La machine à chercher est à la fois gigantesque et microcosmique, et ils sont très nombreux dans les bureaux d'à côté. Au fil des jours, certains d'entre eux vont devenir importants à vos yeux. Le critère de sélection reste alors l'échange humain. Dans cette catégorie, la vie m'a bien servie : les "locaux", Fabien, Maëlle, Guillaume, Delphine, JP, Pierre, Patrick, Xavier, Nicolas, Christine, Julianna, Olivier, Frédérique, Marco, Francesca, Magali, Erwann, Joachim, Régine, Katia, Alain, Daniel, Stéphane, Michel, sans oublier l'équipe du café ; et les "plus loin" Fabien, Damien, Damien, Inam, René, Carla, Pascal, Mireille, Pierrick, David.

Pour finir, il y a ceux qui ne liront probablement jamais ces lignes et qui ont ensoleillé ma vie. Certains ne savent pas lire, ou pas encore lire ; d'autres ne sont plus là pour lire ces lignes ; et une grande majorité d'entre eux n'ont aucune raison de poser leurs yeux sur un tel document, à part pour me faire plaisir l'espace d'une minute.
J'ai commencé à faire la liste de ces personnes, mais elle est bien trop longue. S'ils lisent tout de même ces lignes, ils se reconnaîtront : ils m'ont accueillie chez eux ; nous avons eu beaucoup de difficultés à nous voir parfois car nos vies sont trop pleines ; nous avons partagé du temps, des expériences, du plaisir, des repas et des apéros, de la déception et des larmes parfois, des discussions, des embrassades, des sourires ; et puis des tours de manivelle, des balles et des foulards, des massues et des flammes, des baudriers, des bleus, des pas et des notes. Nous avons refait le monde et pas que sur le papier. Merci à vous de mettre des couleurs partout...

J'avais dit un peu de contenance, alors trêve de remerciements, passons aux choses sérieuses.

Table des matières

Table des figures

Liste des tableaux

Introduction

Une courbe elliptique est un objet mathématique étudié particulièrement en géométrie algébrique. On peut en donner plusieurs définitions selon la personne à laquelle on s'adresse. Certaines paraîtront trop complexes pour les uns, d'autres trop restreintes pour les autres.

La cryptographie est un domaine étudié depuis que l'homme a le besoin de communiquer des informations secrètes à certains de ses congénères (et là tous les coups sont permis pour obtenir ou sécuriser ces informations, cf [Sin99]). Des individus de divers horizons se sont intéressés à cette discipline : linguistes, cruciverbistes, chimistes, inventeurs, mathématiciens, curieux, informaticiens, ingénieurs, physiciens. De nos jours, la quasi-totalité des informations circule par voie électronique.

Cryptographie et courbes elliptiques se sont rencontrées sans le savoir, il y a maintenant un peu plus de trente ans, avec l'apparition du protocole d'échange de clés Diffie Hellman [DH76] et du cryptosystème ElGamal [Elg85]. Ces protocoles cryptographiques utilisent la structure de groupe de l'ensemble des éléments inversibles d'un corps fini $\mathbb{F}_p^*$.
En appliquant, ces procédés aux groupes définis par des courbes elliptiques (cf. [Mil86] et [Kob87]), une nouvelle spécialité voit le jour à la fin des années 80 : ECC, Elliptic Curve Cryptography.

Comme son nom l'indique, cette discipline caractérise les procédés cryptographiques utilisant les courbes elliptiques. Depuis ce temps, ce domaine est exploré par de nombreux chercheurs issus de la recherche mathématique, informatique, cryptographique et industrielle.
Aujourd'hui, le colloque ECC est dédié à ce type de recherche et avec la standardisation de plusieurs protocoles comme ECDSA (Elliptic Curve Digital Signature Algorithm) ou ECIES (Elliptic Curve Integrated Encryption System) l'utilité des courbes elliptiques en cryptographie n'est plus à démontrer. On en trouve notamment dans les dernières implémentations de la librairie openssl.

Le sujet d'étude de ma thèse concerne l'utilisation dans le domaine de la cryptographie de courbes elliptiques définies sur un anneau ($\mathbb{F}_p[\varepsilon]$ où $\varepsilon^2 = 0$).

Le cas des courbes elliptiques définies sur un anneau a été étudié sous différents aspects.
En géométrie algébrique, l'étude de ces courbes dans le cas d'un anneau local est exposée dans le livre de Silverman [Sil94].
En théorie des nombres, l'étude de ces courbes sur un anneau $\mathbb{Z}_{p.q}$ pour p et q nombres premiers distincts dans l'article [Len86] de Lenstra a permis de factoriser de grands entiers en utilisant les courbes elliptiques.
En cryptographie, la thèse de Sebastiá Martin [Mar98] étudie les courbes sur un anneau $\mathbb{Z}_N$

avec $N = p \times q$ deux premiers distincts. Il construit alors un cryptosystème de type ElGamal utilisant ces courbes. Il étudie également le nombre de points des courbes définies sur les anneaux de type $\mathbb{Z}_p^n$.

De notre coté, nous avons étudié les propriétés des courbes elliptiques définies sur un anneau de type $\mathbb{F}_p[\varepsilon]$ d'un point de vue cryptographique.

Cette étude nous a permis dans un premier temps de construire une attaque du problème du logarithme discret sur des courbes elliptiques définies sur un corps fini et de trace nulle. Dans un second temps, nous avons pu construire un cryptosystème basé sur ces courbes puis nous en avons étudié certaines propriétés de sécurité.
L'attaque est exposée dans la section 4.1, la construction de nouveaux cryptosystèmes est développée dans la section 4.2 et leur sécurité est étudiée dans la section 4.3. Le chapitre 4 est donc consacré aux applications cryptographiques de ces courbes.

Les trois premiers chapitres exposent les idées et les propriétés qui permettent d'aboutir à ces résultats.

Tout d'abord, le chapitre 1 contient les prérequis nécessaires à la compréhension de ma thèse dans les différents domaines concernés.
En effet, dans ce rapport, objets mathématiques et concepts informatiques se rencontrent et se croisent inlassablement. Avec l'étude de sécurité de système de chiffrement, les espaces probabilisés et les ensembles de distributions entrent eux aussi dans la danse.
Le chapitre 1 expose donc ces différentes notions en plusieurs parties. Chacune d'entre elles revient délibérément sur les notions de base. Le lecteur peut ainsi les retrouver directement dans ce chapitre.
J'y ai également apporté ma contribution en développant dans la partie 1.3.2 le concept d'ensemble effectif. Il permet d'associer une taille à un ensemble (ou à une famille d'ensembles) et de mesurer ensuite le temps de calcul et l'efficacité d'un algorithme.

Le chapitre 1 se déroule de la façon suivante.

La première section 1.1 traite de géométrie algébrique. Le niveau reste élémentaire. Y sont introduits l'anneau $\mathbb{F}_q[\varepsilon]$ et la notion de courbe elliptique la plus répandue en cryptographie.
La deuxième section 1.2 traite de probabilités. Même si les espaces probabilisés restent finis et donc, somme toute, assez simples à cerner, cette partie revient en détail sur leur construction et permet de comprendre les notations de probabilité largement usitées dans les preuves de sécurité.
La troisième section 1.3 est plus conséquente ; elle rappelle différentes notions d'algorithmique. Pour cela, je commence par décrire la modélisation d'un ordinateur (ou plus précisément d'un programme) par une machine de Turing (1.3.1), puis je donne différentes définitions relatives aux objets sous-jacents à la notion d'algorithme (1.3.2) ; en particulier j'y développe la notion d'ensemble effectif.
Ensuite, dans la partie 1.3.3 sont introduites les notions de temps de calcul et de polynomialité. Le cas des algorithmes déterministes étant traité, nous généralisons toutes ces notions au cas plus large des algorithmes probabilistes (1.3.4).
Enfin, dans les deux dernières sections de cette partie 1.3, nous abordons la notion de problèmes (au sens de la théorie de la complexité) en introduisant dans chacune d'entre elles les exemples

qui nous concernent. Plus précisément, les problèmes calculatoires sont traités dans la section 1.3.5, et les problèmes de logarithme discret et de problème CDH y sont introduits. Puis les problèmes décisionnels sont traités dans la section 1.3.6 où le problème DDH est introduit.
Les deux dernières parties du chapitre 1 rappellent certaines notions de base de cryptographie. La section 1.4 traite de système de chiffrement à clé publique et de différents types d'attaques envisageables. Enfin la section 1.5 traite de preuves de sécurité.

Une fois ces différentes notions introduites, pour traiter de l'utilisation cryptographique des courbes elliptiques définies sur l'anneau $\mathbb{F}_p[\varepsilon]$ où $\varepsilon^2 = 0$, la première étape est alors de comprendre les objets mathématiques manipulés. Ainsi, le chapitre 2 traite de géométrie algébrique et plus particulièrement de courbes elliptiques sur un anneau local. L'intérêt final de ce chapitre est de connaître explicitement les calculs à effectuer pour obtenir la somme de deux éléments de $E(\mathbb{F}_q[\varepsilon])$; il s'agit de ma contribution principale dans cette partie.

Le chapitre 2 est composé de trois parties.

La première 2.1 regroupe des résultats généraux concernant les courbes elliptiques définies sur un anneau local. Elle permet surtout de nous assurer que ces ensembles sont munis d'une structure de groupe. Elle donne également des formules de calcul explicites.
La deuxième partie 2.2 traite plus spécifiquement de l'anneau local $\mathbb{F}_q[\varepsilon]$ et étudie la structure des courbes elliptiques définies sur cet anneau. Ces résultats diffèrent selon la cardinalité de la courbe et tous les cas y sont traités.
Enfin, la troisième et dernière partie 2.3 est consacrée à la loi d'addition. Il s'agit d'obtenir des calculs simplifiés des formules plus générales rappelées dans la section 2.1. Celles-ci diffèrent selon la situation des éléments à additionner, il s'agit donc de savoir explicitement que faire et quand.

Les objets mathématiques cernés, nous devons alors les faire manipuler à un ordinateur. Le chapitre 3 traite alors des considérations effectives de ces nouveaux objets.

J'ai articulé le chapitre 3 de la façon suivante.

La première partie 3.1 traite du cas général : qu'attend-on d'un groupe pour pouvoir l'utiliser dans un programme informatique ? D. Vergnaud traite de ces questions dans sa thèse [Ver06, p.93] en introduisant la notion de représentation algorithmique d'un groupe. Pour ma part, j'introduis une notion voisine de groupe effectif. Elle aussi est munie d'une notion de taille associée (tout comme l'est celle d'ensemble effectif dans la section 1.3.3).
La deuxième partie 3.2 s'attarde alors sur les courbes elliptiques définies sur $\mathbb{F}_q[\varepsilon]$ afin de s'assurer qu'elles sont bien intéressantes d'un point de vue algorithmique ; qu'il n'y ait aucun problème de codage des données ou de temps de calcul de la somme.
Enfin, la troisième partie 3.3 de ce chapitre s'intéresse à certains problèmes liés à cette structure de groupe. En particulier, dans la section 3.3.1, je montre que les problèmes calculatoires du logarithme discret et de Diffie Hellman sont équivalents à leurs homologues sur des courbes elliptiques "classiques". Puis, dans la section 3.3.2, je traite le cas du problème décisionnel de Diffie Hellman, en montrant que celui-ci est facile à résoudre.

Le dernier chapitre 4, enfin, traite de cryptographie et illustre l'utilisation que nous pouvons faire des courbes elliptiques définies sur l'anneau $\mathbb{F}_q[\varepsilon]$.

La première partie 4.1 expose une attaque du problème du logarithme discret sur des courbes elliptiques définies sur un corps et vérifiant la propriéte suivante : leur cardinal est égal au cardinal du corps. Depuis une dizaine d'années, plusieurs attaques existent déjà sur ce type de courbes (cf. [AS98], [Sem98] et [Sma99]). Il n'en reste pas moins que celle que j'expose ici est particulièrement efficace (les opérations effectuées sont deux exponentiations et une division). Après avoir utilisé les courbes elliptiques définies sur $\mathbb{F}_q[\varepsilon]$ à des fins cryptanalystes, les deux dernières parties s'attachent à leurs aspects cryptographiques. Je les utilise pour concevoir un cryptosystème de type El Gamal. Ce dernier, comparé aux cryptosystèmes de type El Gamal définis sur des courbes elliptiques "classiques", fournit l'avantage de ne présenter aucun problème de codage. Le prix à payer est alors de diviser la bande passante par deux. Les solutions envisagées pour régler ce type de problème pour des courbes elliptiques "classiques" la divisant au moins par deux, notre système en devient plus performant de ce point de vue (cf. [Vir05]).
J'expose ce cryptosystème dans la partie 4.2 et enfin j'étudie certaines de ses propriétés de sécurité dans la partie 4.3. J'obtiens alors, dans la section 4.3.1, que ce système de chiffrement est OW CPA au même titre qu'une version El Gamal définie sur des courbes elliptiques "classiques" ; c'est à dire sous l'hypothèse que le problème CDH soit difficile pour ces courbes. Enfin dans la section 4.3.2, je présente un attaquant IND-CPA de ce cryptosystème, puis je précise son avantage. Cette faiblesse n'est pas surprenante dans la mesure où le cryptosystème est similaire au système El Gamal. Mais, cette étude a nécessité une preuve appropriée dans la mesure où notre cryptosystème n'est pas, à proprement parler, de type El Gamal même s'il s'en approche beaucoup.

Chapitre 1

Notions de bases

1.1 Algèbre

Dans cette section, nous rappelons brièvement la notion de courbe elliptique sur un corps (en particulier la loi d'addition), puis nous introduisons l'anneau $\mathbb{F}_q[\varepsilon]$ (et ses propriétés), enfin nous introduisons la notion de courbe elliptique sur un anneau.

1.1.1 Courbe elliptique sur un corps

Dans cette section, K désigne un corps de caractéristique différente de 2 et 3.

Définition 1.1.1 *Si K est un corps de caractéristique différente de 2 et 3, une courbe elliptique sur K est déterminée par une équation du type $Y^2Z = X^3 + aXZ^2 + bZ^3$ avec a et b dans K tels que $4a^3 + 27b^2 \neq 0$.*
On note alors $E_{a,b}$ cette courbe.
Les points de cette courbe sont les éléments $[X : Y : Z]$ de l'espace projectif $\mathbb{P}^2(K)$ vérifiant l'équation :

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

L'ensemble des points de $E_{a,b}$ est noté $E_{a,b}(K)$.

En s'intéressant aux points $[X : Y : Z]$ de $E_{a,b}(K)$ tels que Z soit nul, on ne trouve qu'un seul point vérifiant cette condition. Les points restants s'écrivent alors tous de façon unique sous la forme $[x : y : 1]$ et on obtient la proposition 1.1.1.

Proposition 1.1.1 *L'ensemble des points sur K d'une courbe elliptique $E_{a,b}$ vérifie :*

$$E_{a,b}(K) = \left\{[x : y : 1] : y^2 = x^3 + ax + b\right\} \cup \left\{[0 : 1 : 0]\right\}.$$

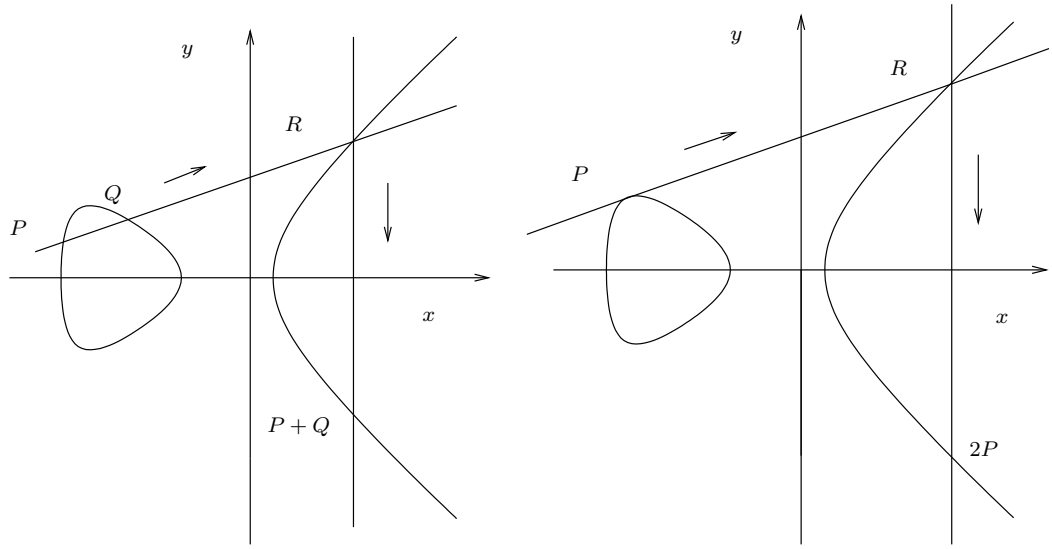
L'équation $y^2 = x^3 + ax + b$ est appelée équation de Weierstrass de la courbe *et le point $[0 : 1 : 0]$ appelé* point à l'infini, *il est noté O.*

Proposition 1.1.2 *L'ensemble $E_{a,b}(K)$ est muni d'une loi de groupe (notée additivement), dont l'élément neutre est O.*

Construction géométrique réelle. Lorsque K est le corps des réels, on peut tracer la courbe d'équation $y^2 = x^3 + ax + b$ dans un plan réel.
Les points de la courbe elliptique $E_{a,b}(\mathbb{R})$ sont alors représentés par les points de cette courbe hormis le point O que l'on peut imaginer "à l'infini", "en haut" ou "en bas". La somme de deux points distincts P et Q différents de O s'obtient alors géométriquement de la façon suivante :
- on considère la droite (PQ) ;
- si cette droite est parallèle à l'axe des ordonnées, la somme de P et Q est O ;
- sinon cette droite coupe la courbe elliptique en un unique troisième point R de coordonnées (x_R, y_R) et la somme de P et Q est le point $(x_R, -y_R) = -R$.

Pour obtenir la somme des points P et Q quand $P = Q$ (autrement dit pour obtenir le point $2P$), on procède de la même façon, en considérant la tangente à la courbe passant par le point P.

Cette construction s'appelle méthode de la sécante-tangente. La figure 1.1 l'illustre dans $\mathbb{R}$.



Si $P \neq Q$,
on trace la sécante (PQ).

Si $P = Q$,
on trace la tangente en P.

FIG. 1.1 – Méthode de la sécante-tangente

Cette construction permet d'obtenir des formules d'addition valables pour un corps de caractéristique différente de 2 et 3.
Les formules permettant de calculer la somme de deux points $P = [x_P : y_P : 1]$ et $Q = [x_Q : y_Q : 1]$ sont données dans le tableau 1.1.
On peut remarquer également que :
- l'opposé du point P est $-P = [x_P : -y_P : 1]$;
- si $x_P = x_Q$, alors $P = Q$ ou $P = -Q$;
- si $y_P = 0$, alors $P = -P$.

Ces propriétés permettent de s'assurer qu'il n'y a pas de problème de définition lors des calculs de la somme de deux points ; en effet :

Si $P = -Q$	i.e. si $x_P = x_Q$ et $y_P = -y_Q$	alors on a : $P + Q = O = [0 : 1 : 0]$
Si $P \neq -Q$	alors on a : $P + Q = [x_{P+Q} : y_{P+Q} : 1]$ avec $x_{P+Q} = \lambda^2 - x_P - x_Q$ et $y_{P+Q} = -y_P + \lambda\,(x_P - x_{P+Q})$	
si, de plus, $P = Q$	i.e. si $x_P = x_Q$ et $y_P = y_Q$	alors on a : $\lambda = \dfrac{3x_P^2 + a}{2y_P}$.
si, de plus, $P \neq Q$	i.e. si $x_P \neq x_Q$	alors on a : $\lambda = \dfrac{y_Q - y_P}{x_Q - x_P}$

TAB. 1.1 – Formules d'addition de deux points d'une courbe elliptique définie sur un corps de caractéristique différente de 2 et 3.

– si $P \neq -Q$ et $P \neq Q$, alors $x_Q - x_P \neq 0$;
– si $P \neq -Q$ et $P = Q$, alors $y_P \neq 0$.
On retrouve tous ces résultats dans le livre de Silverman [Sil85], ils sont également exposés avec un minimum de prérequis dans [Joy95].

Par la suite, nous étudions les courbes elliptiques définies sur un anneau. Dans ce cadre, les formules d'addition de le tableau 1.1 ne peuvent pas être utilisées : à la différence d'un corps, un anneau peut contenir des éléments non inversibles non nuls.
Cependant, un corps étant aussi un anneau, les formules valables pour un anneau doivent l'être a fortiori pour un corps.

Nous verrons comment étendre les formules du tableau 1.1 dans le cadre d'un anneau dans le chapitre 2.
En particulier, dans la partie 2.2, nous appliquerons ces résultats à l'anneau $\mathbb{F}_p[\varepsilon]$ introduit dans la section suivante.

1.1.2 L'anneau $\mathbb{F}_q[\varepsilon]$

Soit p un nombre premier, q une puissance de p. On considère le corps fini de cardinal q, noté $\mathbb{F}_q$. Et on note $\mathbb{F}_q[\varepsilon]$ l'anneau $\dfrac{\mathbb{F}_q[X]}{X^2}$. Autrement dit, on a :

$$\mathbb{F}_q[\varepsilon] \quad = \quad \{a + b\varepsilon : a, b \in \mathbb{F}_q, \varepsilon^2 = 0\}$$

Nous étudions, dans un premier temps, les éléments non inversibles de cet anneau puis nous nous intéressons à sa structure algébrique.

Lemme 1.1.1 *Les éléments non inversibles de $\mathbb{F}_q[\varepsilon]$ sont les $k\varepsilon$ avec k dans $\mathbb{F}_q$, et pour a et b dans $\mathbb{F}_q$ avec $a \neq 0$, on a :*

$$(a + b\varepsilon)^{-1} = a^{-1} - ba^{-2}\varepsilon.$$

Preuve. Pour a et b dans $\mathbb{F}_q$ avec $a \neq 0$, on a :

$$(a + b\varepsilon)(a^{-1} - ba^{-2}\varepsilon) = 1.$$

De plus, un élément de $\mathbb{F}_q[\varepsilon]$ écrit sous la forme $a + b\varepsilon$ avec a et b dans $\mathbb{F}_q$ est inversible si et seulement si a est non nul. □

Lemme 1.1.2 *L'anneau $\mathbb{F}_q[\varepsilon]$ admet $(\varepsilon) = \varepsilon\mathbb{F}_q$ pour idéal maximal.*

Preuve. Pour a, b et k dans $\mathbb{F}_q$, on a :

$$k\varepsilon(a + b\varepsilon) = ak\varepsilon$$

Donc, on a : $(\varepsilon)\mathbb{F}_q[\varepsilon] \subseteq (\varepsilon)$; et (ε) est un idéal.

De plus, l'anneau quotient est un corps car pour a et b dans $\mathbb{F}_q$, $a + b\varepsilon$ peut être représenté par a et :

$$\frac{\mathbb{F}_q[\varepsilon]}{(\varepsilon)} = \mathbb{F}_q.$$

Donc (ε) est un idéal maximal. □

L'anneau $\mathbb{F}_q[\varepsilon]$ a donc la propriété d'être un anneau local, comme l'indique le lemme 1.1.3.

Lemme 1.1.3 *L'anneau $\mathbb{F}_q[\varepsilon]$ est un anneau local de corps résiduel $\mathbb{F}_q$.*

Preuve. D'après les lemmes 1.1.1 et 1.1.2, l'ensemble des éléments non inversibles de $\mathbb{F}_q[\varepsilon]$ est un idéal maximal de corps résiduel $\mathbb{F}_q$. Donc l'anneau $\mathbb{F}_q[\varepsilon]$ est un anneau local. □

Mais cet anneau est aussi muni d'une structure d'espace vectoriel et d'algèbre, comme le précisent le lemme 1.1.4 et le corollaire 1.1.1.

Lemme 1.1.4 *L'anneau $\mathbb{F}_q[\varepsilon]$ est un $\mathbb{F}_q$-espace vectoriel de dimension 2 de base $(1, \varepsilon)$; soit :*

$$\mathbb{F}_q[\varepsilon] = \mathbb{F}_q + \mathbb{F}_q\varepsilon.$$

Preuve. Pour a, a', b, b' et k dans $\mathbb{F}_q$, on a :

$$a + b\varepsilon + a' + b'\varepsilon = (a + a') + (b + b')\varepsilon$$
$$k(a + b\varepsilon) = ka + kb\varepsilon. \quad \square$$

Corollaire 1.1.1 *L'ensemble $\mathbb{F}_q[\varepsilon]$ est une $\mathbb{F}_q$-algèbre locale.*

Preuve. D'après les lemmes 1.1.3 et 1.1.4. □

Nous utiliserons cette structure algébrique pour obtenir les formules d'addition établies dans la section 2.3.

1.2 Probabilités

Notations. Dans cette section, les espaces probabilisés seront finis. Pour un ensemble fondamental fini F, la tribu considérée sera toujours l'ensemble $\mathcal{P}(F)$ des parties de F. Ainsi, quand nous écrirons "l'espace probabilisé fini (F, P_F)", il s'agira de l'espace probabilisé $(F, \mathcal{P}(F), P_F)$ où P_F est la loi de probabilité définie sur l'ensemble F.
Toutes les mesures considérées ici sont des mesures de probabilité.

Pour étudier la sécurité d'un cryptosystème, nous allons considérer des expériences aléatoires décomposées en expériences élémentaires. La notation que nous allons introduire à l'issue de cette section permet de visualiser rapidement le déroulement de l'expérience aléatoire considérée.
Illustrons cette partie par l'exemple d'expérience aléatoire 1.2.1 :

Exemple 1.2.1 *On lance un dé de six.*
Si le résultat est impair, on lance un dé de dix ; sinon on lance un dé de quatre. Suite à cette expérience aléatoire, on s'intéresse à la probabilité que la somme des deux lancers soit égale à 6.
On peut considérer que cette expérience est composée de deux (ou trois) expériences élémentaires :

1. *le jet d'un dé de six. On note le résultat de cette expérience d_1 ;*
2. *le jet d'un second dé (de dix ou de quatre). On note le résultat de cette expérience d_2 ;*
3. *on peut ajouter, si on le souhaite, cette dernière expérience : le calcul de la somme de d_1 et d_2. On note le résultat somme. Cette variable est alors entièrement déterminée par les résultats des expériences précédentes et on a : somme $= d_1 + d_2$.*

On s'intéresse à la probabilité que la variable somme soit égale à 6.

Cet exemple illustre les trois types d'expériences élémentaires que nous allons considérer par la suite :

1. l'exécution d'un algorithme probabiliste, que l'on peut illustrer par la deuxième expérience de l'exemple 1.2.1 ou en cryptographie par le choix d'un couple de clés ;
2. l'exécution d'un algorithme déterministe, que l'on peut illustrer par la dernière expérience de l'exemple 1.2.1 ou en cryptographie par le déchiffrement d'un cryptogramme ;
3. le tir uniforme dans un ensemble fini, que l'on peut illustrer par la première expérience de l'exemple 1.2.1 ou en cryptographie par le choix d'un clair dans l'ensemble des clairs que l'on peut chiffrer à l'aide d'une clé donnée.

La définition 1.2.1 donne notre version d'une expérience élémentaire que nous avons restreint aux cas d'un espace fini :

Définition 1.2.1 *Soit F un ensemble fini. Une* expérience élémentaire indexée par F *est une famille d'espaces probabilisés finis indexée par une variable appartenant à F. On la note par exemple $E = (E(x), P_{E,x})_{x \in F}$ où pour tout x dans F, $E(x)$ est un ensemble fini muni d'une loi de probabilité notée $P_{E,x}$.*

Dans l'exemple 1.2.1, on peut décrire la première expérience de cette façon.

Exemple 1.2.2 *Il s'agit d'une famille contenant un seul espace probabilisé :*

$$E_1 = (E_1(x), P_{E_1,x})_{x\in F} \text{ où } F = \{\star\},\ E_1(\star) = [\![1,6]\!] \text{ et } P_{E_1,\star} \text{ est définie par :}$$
$$P_{E_1,\star}(1) = P_{E_1,\star}(2) = P_{E_1,\star}(3) = P_{E_1,\star}(4) = P_{E_1,\star}(5) = P_{E_1,\star}(6) = \frac{1}{6}.$$

Nous décrirons ultérieurement les autres expériences élémentaires de l'exemple 1.2.1.

La proposition 1.2.1 précise comment la réalisation d'une expérience élémentaire à partir de données issues d'un espace probabilisé, fournit un autre espace probabilisé modélisant le résultat de l'expérience élémentaire considérée.

Proposition 1.2.1 *Soit (F, P_F) un espace probabilisé fini et $E = (E(x), P_{E,x})_{x\in F}$ une expérience élémentaire indexée par F.*
Le symbole $\bigsqcup$ représente ici l'union disjointe.
Notons $E_F = \bigsqcup_{x\in F} E(x) = \{(x, y) : x \in F, y \in E(x)\}$; la fonction définie sur E_F par :

$$\forall (x, y) \in E_F, P_{E_F}(x, y) = P_F(x)P_{E,x}(y)$$

définit une mesure sur $\mathcal{P}(E_F)$.
*L'espace probabilisé fini (E_F, P_{E_F}) est appelé l'*espace probabilisé par F et E.

Illustrons la proposition 1.2.1 par la réalisation de la deuxième expérience élémentaire de l'exemple 1.2.1.

Exemple 1.2.3 *On pose ici pour $(F, P_F) = \left(E_1(\star), P_{E_1(\star)}\right)$ exposé dans l'exemple 1.2.2. La deuxième expérience de l'exemple 1.2.1 peut être alors décrite ainsi :*

$$E_2 = (E_2(x), P_{E_2,x})_{x\in F} \text{ où } F = E_1(\star) = [\![1,6]\!] ;$$

$$\text{pour } d_1 \text{ dans } \{1,3,5\},\ E_2(d_1) = [\![1,4]\!] \text{ et } P_{E_2,d_1} \text{ est définie par :}$$
$$P_{E_2,d_1}(1) = ... = P_{E_2,d_1}(4) = \frac{1}{4} ;$$

$$\text{pour } d_1 \text{ dans } \{2,4,6\},\ E_2(d_1) = [\![1,10]\!] \text{ et } P_{E_2,d_1} \text{ est définie par :}$$
$$P_{E_2,d_1}(1) = ... = P_{E_2,d_1}(10) = \frac{1}{10}$$

L'espace probabilisé par F et E_2 est alors $(E_{2_F}, P_{E_{2_F}})$ où :

$$E_{2_F} = (\{1,3,5\} \times [\![1,4]\!]) \bigcup (\{2,4,6\} \times [\![1,10]\!]) \text{ et } P_{E_{2_F}} \text{ est définie par :}$$

$$\text{pour } d_2 \text{ dans } [\![1,4]\!],\ P_{E_{2_F}}(1,d_2) = P_{E_{2_F}}(3,d_2) = P_{E_{2_F}}(5,d_2) = \frac{1}{6} \times \frac{1}{4} = \frac{1}{24} ;$$
$$\text{pour } d_2 \text{ dans } [\![1,10]\!],\ P_{E_{2_F}}(2,d_2) = P_{E_{2_F}}(4,d_2) = P_{E_{2_F}}(6,d_2) = \frac{1}{6} \times \frac{1}{10} = \frac{1}{60}.$$

Les expériences élémentaires, que nous considérons par la suite, fournissent une famille d'espaces probabilisés finis paramétrée par le résultat des expériences précédentes :

1. dans le cas de l'exécution d'un algorithme probabiliste, cette famille est entièrement déterminée par celui-ci. Nous détaillons ce procédé dans la section 1.3.4 et plus précisément dans la proposition 1.3.1 ;
2. dans le cas de l'exécution d'un algorithme déterministe, chaque espace probabilisé est réduit à un élément (la sortie de l'algorithme considéré, dont l'entrée peut dépendre du résultat des expériences précédentes) ;

3. dans le cas d'un tir uniforme, chaque espace probabilisé est muni d'une mesure uniforme.

Les expériences aléatoires que nous rencontrons sont décrites par une suite finie d'expériences élémentaires de ce type. Nous modélisons alors, dans la définition 1.2.2, une expérience aléatoire par une suite finie d'expériences élémentaires indexées par les résultats des expériences précédentes.

Définition 1.2.2 *Notons F_0 un espace probabilisé $(\star, P_\star)$ où $\star$ est un ensemble fondamental contenant exactement un élément.*
Une expérience aléatoire *E est la donnée d'une suite finie d'expériences élémentaires $(E_i)_{i=1..n}$ telles que :*
– *E_1 est indexée par F_0 ;*
– *pour tout i entre 2 et n, E_i est indexée par F_{i-1} l'espace probabilisé par F_{i-2} et E_{i-1}.*
*On pose alors $F = F_n = E_{nF_{n-1}}$. C'est l'*ensemble des sorties de l'expérience aléatoire *E.*

Une expérience aléatoire composée de n expériences élémentaires détermine alors un $(n+1)$-uplet d'espaces probabilisés par le procédé suivant :
– (F_0, P_{F_0}) ;
– pour i de 1 à n, (F_i, P_{F_i}) avec $P_{F_i} = P_{E_{iF_{i-1}}}$.
L'espace fondamental de chacun d'eux est composé des résultats de toutes les expériences élémentaires précédentes. En particulier, le dernier de ces espaces probabilisés modélise tous les résultats successifs obtenus lors de la réalisation de l'expérience aléatoire ainsi que la probabilité d'obtenir un tel résultat.

Décomposition d'une sortie de E. Par récurrence sur i de 1 à n, on a :

$$\text{pour tout } i \text{ de } 1 \text{ à } n,$$
$$F_i = \{(x_0, x_1, x_2, ..., x_i) : x_0 \in F_0, x_1 \in E_1(x_0), ..., x_i \in E_i(x_0, x_1, ..., x_{i-1})\}.$$

Ainsi, l'ensemble des sorties de l'expérience aléatoire $(E_i)_{i=1..n}$ est de la forme :

$$\{(x_0, x_1, x_2, ..., x_n) : x_0 \in F_0, x_1 \in E_1(x_0), ..., x_n \in E_n(x_0, x_1, ..., x_{n-1})\}.$$

Remarquons que cet ensemble n'est pas nécessairement un produit direct.

Notation. Notons x_0 l'unique élément de F_0, la sortie de l'expérience E est notée :

$$\left\{(x_0, x_1, ..., x_n) : x_1 \xleftarrow{P_{F_1}} F_1, x_2 \xleftarrow{P_{E_2,x_1}} E_2(x_1), ..., x_n \xleftarrow{P_{E_n,(x_1,..,x_{n-1})}} E_n(x_1, .., x_{n-1})\right\}.$$

Le formalisme de cette section permet de définir en 1.2.3 la probabilité d'un évènement lié aux résultats d'une suite d'expériences élémentaires.

Définition 1.2.3 *Soit E une expérience aléatoire de sorties F. Soit f une fonction de F dans un ensemble S. Pour un élément a de S, la probabilité $P_f(a)$ est appelée* probabilité que $f = a$ suite à E.

On la note :

$$P\left(f(x)=a : x \xleftarrow{P_E} E\right) =$$
$$P\left(f(x_0,x_1,...,x_n)=a : x_1 \xleftarrow{P_{F_1}} F_1, x_2 \xleftarrow{P_{E_2,x_1}} E_2(x_1),...,x_n \xleftarrow{P_{E_n,(x_1,..,x_{n-1})}} E_n(x_1,..,x_{n-1})\right).$$

De plus, pour une valeur fixée i_0,

1. si pour tout x dans F_{i_0-1}, on a $\#E_{i_0}(x)=1$, alors on note la probabilité que $f=a$ suite à E :

$$P\left(f(x_0,x_1,...,x_n)=a : x_1 \xleftarrow{...} F_1,..., \boxed{x_{i_0}=E_{i_0}(x_1,..,x_{i_0-1})},...,x_n \xleftarrow{...} E_n(x_1,..,x_{n-1})\right);$$

2. si pour tout x dans F_{i_0-1}, la mesure $P_{E_{i_0},x}$ est une loi uniforme sur $E_{i_0}(x)$, alors on note la probabilité que $f=a$ suite à E :

$$P\left(f(x_0,x_1,...,x_n)=a : x_1 \xleftarrow{...} F_1,..., \boxed{x_{i_0} \xleftarrow{u} E_{i_0}(x_1,..,x_{i_0-1})},...,x_n \xleftarrow{...} E_n(x_1,..,x_{n-1})\right).$$

En pratique, ces notations supplémentaires sont utilisées pour modéliser respectivement l'exécution d'un algorithme déterministe et un tir uniforme.

Elles permettent d'écrire l'exemple 1.2.1 de la façon suivante :

Exemple 1.2.4 *La probabilité que la somme des dés soit égale à* 6 *suite à l'expérience aléatoire décrite dans l'exemple 1.2.1 est notée :*

$$P\left(somme=6 : d_1 \xleftarrow{u} [\![1,6]\!], d_2 \xleftarrow{P_{E_2,d_1}} E_2(d_1), somme=d_1+d_2\right)=\frac{7}{60}.$$

Les notations $E_2(d_1)$ et P_{E_2,d_1} ont été introduites dans l'exemple 1.2.3.

Aussi, quand le contexte le permet, on omet de préciser la mesure dont est pourvu l'espace considéré. Par exemple, nous verrons dans la proposition 1.3.1 que l'exécution d'un algorithme probabiliste $\mathcal{A}$ sur une entrée x définit un espace probabilisé noté $(\mathcal{A}(x),P_{\mathcal{A},x})$. Cette famille d'espaces probabilisés est déterminée uniquement par $\mathcal{A}$ et x et on ne précisera pas que l'on munit l'ensemble fondamental $\mathcal{A}(x)$ de la mesure $P_{\mathcal{A},x}$.
L'expérience élémentaire associée sera alors notée, $y \xleftarrow{} \mathcal{A}(x)$ au lieu de $y \xleftarrow{P_{\mathcal{A},x}} \mathcal{A}(x)$.

Nous utiliserons ces notations dans la section 4.3.

1.3 Algorithmique

Détaillons maintenant les objets liés à la notion d'algorithme.
Pour cela, nous allons commencer par décrire le fonctionnement d'une machine de Turing (section 1.3.1), puis nous insistons sur l'importance du choix de codage des données (section 1.3.2). Nous introduisons les notions relatives aux algorithmes, en particulier le temps d'exécution (section 1.3.3).
Nous détaillons ensuite les notions plus spécifiques relatives aux algorithmes probabilistes (section 1.3.4). Enfin, dans la dernière section 1.3.5, nous introduisons la notion de problème liée à celle d'algorithme et nous illustrons ces relations par des exemples.

1.3.1 Modélisation d'un algorithme : machine de Turing

Dans cette section, nous allons décrire le fonctionnement d'une machine de Turing qui, à ce jour, modélise le plus soigneusement la notion d'algorithme. Cette description permet de clarifier certains objets et d'en comprendre l'existence et l'intérêt.
Pour formaliser le déroulement d'un algorithme, on utilise la notion de machine de Turing (à un ou plusieurs rubans).
Un *ruban* est une suite de caractères (appelé *symboles*) et d'un caractère supplémentaire appelé *le "Vide"*. Cette suite (infinie) est composée d'un nombre fini de symboles, les termes restants sont "Vide".
On représente un ruban par une succession de cases pouvant être vide ou contenir un symbole.
Une machine de Turing dispose d'une (ou de plusieurs) tête(s) de lecture et d'une tête d'écriture pouvant respectivement lire le contenu d'une case sur chaque ruban de lecture et écrire un symbole (et éventuellement le "Vide") contenu sur une case d'un ruban d'écriture.

Cette machine est capable d'agir sur ces rubans : elle peut les déplacer d'une case vers la droite ou vers la gauche et écrire sur le ruban d'écriture avant de le déplacer d'une case pour être en mesure d'inscrire un prochain caractère (dans la description usuelle, ce sont les têtes de lecture et écriture qui se déplacent sur les rubans).
Initialement, cette machine se trouve dans un état particulier (appelé à ce titre *état initial*) ; par la suite elle peut changer d'état (parmi une liste finie d'états déterminés).
Elle dispose alors d'un "mode emploi" (appelé *fonction de transition*) qui lui indique quelle action faire selon l'état courant et ce qu'elle lit sur le (ou les) ruban(s) de lecture.
La machine de Turing modélise alors l'ordinateur, pouvant lire et écrire des symboles éventuellement "Vide" et la fonction de transition correspond à la notion usuelle que nous avons d'un algorithme, comme une suite d'instructions à suivre selon différents cas de figure.
Une *entrée* soumise à un algorithme est alors modélisée par une suite finie de symboles, c'est à dire par un ruban dont les cases contiennent le "Vide" sauf sur un nombre fini de cases. On aura autant de rubans que d'entrées.
L'exécution d'un algorithme sur une entrée se modélise alors par une succession d'étapes (appelées *configurations*) décrivant le calcul de la machine de Turing sur son entrée.
Plus précisément, la première case non "Vide" du (ou des) ruban(s) représentant l'entrée est placée en face de la (ou des) tête(s) de lecture, un ruban vide est placé sur la tête d'écriture et la machine se trouve dans l'état initial.
Cette description (état, ruban(s) de lecture, ruban d'écriture, positionnement des têtes de lecture et écriture) définit la configuration initiale.
Une suite de configurations est alors définie selon le processus suivant :
en fonction :
- de l'état de la machine ;
- du (ou des) symbole(s) lu(s) (éventuellement "Vide") par la (ou les) têtes de lecture ;

la fonction de transition détermine la configuration suivante :
- en donnant le nouvel état de la machine (qui peut être le même que précédemment) ;
- en écrivant (ou pas) un symbole éventuellement "Vide" sur le ruban d'écriture ;
- en déplaçant (ou pas) la (ou les) ruban(s) de lecture d'une case (vers la gauche ou vers la droite).

Enfin, en plus de l'ensemble fini d'états dans lesquels la machine peut se trouver, il existe un état distingué (appelé *état final*) à la suite duquel la machine arrête son calcul.
Lorsque la machine de Turing est dans cet état, la suite de configurations est finie (l'action de

la machine est terminée) et on appelle *sortie* le ruban d'écriture.

Le cas de figure où la machine de Turing ne s'arrête pas ne nous intéresse pas. Donc, par la suite, nous restreindrons l'ensemble des entrées d'un algorithme aux seules entrées telles que la suite de configurations associée soit finie.
Pour conclure, nous pouvons donner une définition formelle d'une machine de Turing.

Définition 1.3.1 *Soit n un entier naturel, une* machine de Turing déterministe à n rubans *est la donnée d'un quadruplet (E, S, δ, e_0) où :*
- *E est un ensemble fini, dont les éléments sont appelés* états *;*
- *e_0 est un état de E, appelé* état initial *de la machine ;*
- *S est un ensemble fini dont les éléments sont appelés* symboles *;*
- *δ est une fonction de $E \times (S \cup \{Vide\})^n$ dans $(E \cup \{Stop\}) \times (S \cup \{Vide\})^n \times \{+1, -1, 0\}^n$. Elle est appelée* fonction de transition *de la machine.*

Stop est appelé état d'arrêt *de la machine de Turing.*

Nous allons voir dans les sections 1.3.2, 1.3.3 et 1.3.4 les objets liés à la notion de machine de Turing.

1.3.2 Le choix du codage : ensemble effectif

La notion de machine de Turing décrite dans la section 1.3.1 (comme celle de tout algorithme) pose un premier problème : celui du codage des éléments utilisés, à commencer par les objets décrits par le (ou les) ruban(s) d'entrées.
En effet, notons I l'ensemble des objets mathématiques que la machine doit pouvoir accepter en entrée. Il peut s'agir, par exemple, d'entiers, de rationnels, de booléens, de matrices, de points d'une courbe elliptique. La machine doit pouvoir manipuler ces objets, effectuer les opérations usuelles de calcul (addition, multiplication, division, calculs modulaires ou non, tests booléens...) ainsi que les instructions algorithmiques (boucles for, if, while, return...). Pour tout cela, il faut pouvoir dans un premier temps coder ces objets (pour les inscrire sur les rubans).
Ce problème se pose de façon encore plus cruciale dans la réalité : pour fonctionner, l'ordinateur dispose seulement des symboles 0 et 1 (appelés *bits*). Ainsi, tous ces objets doivent être codés par une suite de bits.
Nous devons donc choisir un type de codage, c'est à dire une fonction de I dans $\{0,1\}^*$ qui soit injective pour permettre de construire une bijection entre l'ensemble I et l'ensemble des images des éléments de I. Ainsi, à chaque élément correspond un unique codage et si une suite de bits correspond au codage d'un élément, celui-ci est unique.
Usuellement pour coder un entier, la machine utilise son écriture en base 2 : $n = \sum_{i=0}^{k} n_i 2^i$ avec n_i dans $\{0,1\}$ et $k = \lfloor \log_2(n) \rfloor$.
Ce choix de codage induit alors une taille pour les données du ruban. Pour chaque donnée, elle est définie par le nombre de bits utilisés pour la coder. Par exemple dans le cas précédent, un entier n est codé par une suite de $k + 1 = \lfloor \log_2(n) \rfloor + 1$ bits.

Ainsi, dans cette section, dans un premier temps nous précisons ce que nous entendons par codage (au travers de la notion d'ensemble effectif) puis nous détaillons les différents ensembles utilisés dans cette thèse ainsi que leur taille sous-jacente.
Commençons par définir la notion d'ensemble effectif. Dans la définition 1.3.2, nous indiquons qu'il s'agit d'un ensemble sur lequel a été choisi un type de codage. Celui-ci induit alors une

notion de taille. Elle est primordiale dans le sens où, représentant la taille de paramètres des problèmes considérés, elle est l'"unité de mesure" de la complexité.
Nous en mesurerons plus précisément l'intérêt dans la partie 1.3.5 puis au cours des différents exemples exposés dans cette thèse.
Introduisons, sans plus tarder, cette notion d'ensemble muni d'un codage (déterminant une taille associée) que nous appelons ensemble effectif :

Définition 1.3.2 *Notons l la fonction définie de $\{0,1\}^*$ dans $\mathbb{N}^*$ qui à une suite de bits associe sa longueur.*
On appelle ensemble effectif *tout couple (I,ϕ) où I est un ensemble et ϕ est une injection de I dans $\{0,1\}^*$ appelée* codage de I. *Soit t la fonction définie par le diagramme suivant :*

$$\begin{array}{ccc} I & \overset{\phi}{} & \{0,1\}^* \\ & {\scriptstyle t}\searrow & \downarrow{\scriptstyle l} \\ & & \mathbb{N}^* \end{array}$$

La fonction t est alors appelée la taille sur l'ensemble effectif (I,ϕ).

Un ensemble peut être muni de différents types de codage et le couple obtenu donne alors différents ensembles effectifs de même ensemble sous-jacent, de codages et éventuellement de tailles différents.
En algorithmique, la notion de taille formalise le codage choisi pour enregistrer les données. Elle devient en quelque sorte plus importante que le type de codage : elle permet de mesurer le temps d'exécution d'un algorithme et le lecteur avisé peut retrouver rapidement un type de codage sous-jacent naturel à une taille donnée (s'il est utilisé dans la communauté bien évidemment).
Ainsi, dans les exemples suivants, nous exposons uniquement les ensembles sous-jacents et leur taille associée au codage choisi.

Détaillons les différents ensembles utilisés dans cette thèse ainsi que leurs tailles sous-jacentes. Voici pour commencer les tailles définies sur les ensembles $\mathbb{N}^*$ et $\mathbb{Z}$.

Ensembles effectifs infinis. Débutons par l'ensemble $\mathbb{N}^*$. Dans cette thèse, l'ensemble des entiers naturels est muni de tailles différentes t_1 et t_2 correspondant aux codages de base 1 et 2 respectivement. L'exemple 1.3.1 expose ces définitions dans le détail.

Exemple 1.3.1 *Précisons ici les tailles qui nous servent par la suite.*

L'ensemble $\mathbb{N}^*$: *nous utilisons deux tailles différentes :*

- $t_1 : \begin{array}{ccc} \mathbb{N}^* & \longrightarrow & \mathbb{N}^* \\ k & \mapsto & k \end{array}$
  Cette notion de taille correspond à un codage en base 1 : un entier k est alors codé par une répétition de k fois le chiffre 1. Ainsi, on note $\mathbb{1}^*$ l'ensemble effectif $(\mathbb{N}^*,\phi_1)$ de taille t_1 et 1^k l'entier naturel de taille k correspondant. Cette taille est peu utilisée pour mesurer la complexité d'un algorithme. Elle sert en cryptographie à définir le paramètre de sécurité que nous introduisons dans la partie 1.4 ;
- $t_2 : \begin{array}{ccc} \mathbb{N}^* & \longrightarrow & \mathbb{N}^* \\ k & \mapsto & \lfloor \log_2(k) \rfloor + 1 \end{array}$

Cette notion de taille correspond à un codage en base 2. C'est la notion de taille utilisée couramment en informatique. Ainsi, on note $\mathbb{N}^*$ *l'ensemble effectif* $(\mathbb{N}^*, \phi_2)$ *de taille* t_2.

Nous allons aussi parler de l'ensemble $\mathbb{Z}$ comme ensemble sous-jacent d'un ensemble effectif infini. Nous n'utilisons sur cet ensemble qu'une seule taille correspondant au codage usuel des entiers en base 2. Nous détaillons cet ensemble effectif dans l'exemple 1.3.2.

Exemple 1.3.2 L'ensemble $\mathbb{Z}$ **:** *Nous utilisons la taille suivante :*

$$t_{\mathbb{Z}} : \begin{array}{ccc} \mathbb{Z} & \longrightarrow & \mathbb{N}^* \\ k & \mapsto & \begin{cases} \lfloor \log_2(|k|) \rfloor + 2 & si\ k \neq 0 \\ 1 & sinon \end{cases} \end{array}$$

Cette taille formalise le fait qu'un entier relatif est un entier naturel auquel a été rajouté un signe. Celui-ci est codé par un bit supplémentaire.

Traitons maintenant le cas des ensembles effectifs finis.

Ensembles effectifs finis. Nous allons définir une taille sur un ensemble fini. Celle que nous avons choisie est une taille constante. Elle correspond à un codage des éléments de cet ensemble par des données toutes de même taille.
Par exemple, tout entier entre 0 et $p-1$ sera codé sur $\lfloor \log_2(p) \rfloor + 1$ bits, même 0. Ce type de codage de 0 suppose que l'on sache "à l'avance" que l'élément que l'on va lire appartient à cet ensemble. Nous ne traiterons pas des méthodes pour en informer la machine et nous assimilerons ce type de codage à un codage de taille constante exposé dans l'exemple 1.3.3 .

Exemple 1.3.3 Un ensemble fini E *peut être muni d'un codage* ϕ *tel que* (E, ϕ) *soit un ensemble effectif de taille constante. Soit* t_E *une constante positive, la taille associée est alors :* $\begin{array}{ccc} E & \longrightarrow & \mathbb{N}^* \\ x & \mapsto & t_E \end{array}$
Nous dirons alors des éléments de E *qu'*ils sont de taille t_E *et que* E est de taille t_E. *Cela ne signifie pas que* E *est de cardinal* t_E *et il ne faut alors pas confondre ces deux notions.*

Nous utilisons principalement ce type d'ensemble effectif fini pour représenter les éléments d'un corps fini.

Exemple 1.3.4 Le corps fini $\mathbb{F}_p$ **:** *il existe une bijection entre le corps fini* $\mathbb{F}_p$ *et l'intervalle d'entiers* $[\![0, p-1]\!]$. *Les entiers entre* 0 *et* $p-1$ *sont codés en base 2 par des éléments de taille inférieure ou égale à* $t_2(p) = \lfloor \log_2(p) \rfloor + 1$.
Ainsi, on choisit pour p *premier,* $t_{\mathbb{F}_p} = t_2(p) = \lfloor \log_2(p) \rfloor + 1$.

Les notions d'ensemble effectif et de taille sous-jacente sont primordiales pour comprendre les objets développés en théorie de la complexité. En effet, le temps d'exécution d'un algorithme s'exprime la plupart du temps en fonction de la taille des entrées, et elle permet ainsi d'en apprécier l'efficacité. Nous introduisons ces notions dans la section 1.3.3.

1.3.3 Temps d'exécution : complexité

Dans cette section, nous abordons les notions de temps de calcul, de complexité et d'efficacité d'un algorithme.
Il s'agit en pratique de pouvoir évaluer la durée (en unités de temps) que va nécessiter l'exécution d'un algorithme sur une entrée d'une certaine taille par un ordinateur d'une certaine puissance de calcul et disposant d'une certaine capacité mémoire.

Nous ne traiterons pas ici de la capacité mémoire (ou complexité en espace) que peut nécessiter un algorithme même si elle reste un paramètre important (cf par exemple l'algorithme de Schoof et ses améliorations [Fou01]).

Quant au temps de calcul, la modélisation du fonctionnement d'un algorithme par une machine de Turing nous en donne les interprétations suivantes :

1. il est lié au nombre de configurations nécessaires avant que la machine s'arrête ;
2. il dépend de la taille des entrées : par exemple le nombre de configurations nécessaires pour lire un entier sur un ruban dépend de sa taille sur celui-ci ;
3. la taille de la sortie ne peut être plus grande que ce nombre de configurations puisqu'à chacune d'entre elles la machine ne peut écrire qu'un seul symbole.

Ainsi pour la suite, nous donnons la définition 1.3.3 d'un algorithme.

Définition 1.3.3 *Un algorithme* A *est une machine de Turing à laquelle on peut associer :*
- *un entier* n *;*
- $n+1$ *ensembles effectifs, notés, pour* i *de* 1 *à* n, $Input^i(\mathsf{A}) = (I^i_{\mathsf{A}}, \phi_{I^i_{\mathsf{A}}})$ *et* $Output(\mathsf{A}) = (O_{\mathsf{A}}, \phi_{O_{\mathsf{A}}})$ *de tailles associées notées respectivement* $t_{I^1}, ..., t_{I^n}$ *et* t_O *;*
- I_{A} *une partie de* $I^1_{\mathsf{A}} \times ... \times I^n_{\mathsf{A}}$ *;*
- *une surjection* f_{A} *de* I_{A} *dans* O_{A}*, représentant la fonction calculée par l'algorithme* A*, telle que : pour tout* $x = (x_1, ..., x_n)$ *dans* I_{A} *:*
  - *l'exécution de l'algorithme* A *sur l'entrée* $\left(\phi_{I^1_{\mathsf{A}}}(x_1), ..., \phi_{I^n_{\mathsf{A}}}(x_n)\right)$ *s'arrête,*
  - $\phi_{O_{\mathsf{A}}}(f_{\mathsf{A}}(x_1, ..., x_n))$ *représente la sortie issue de cette exécution.*

On note alors $T_{\mathsf{A}}(x_1, ..., x_n)$ *le nombre de configurations construites pour obtenir ce résultat. Nous appelons :*
- $I_{\mathsf{A}} \subseteq I^1_{\mathsf{A}} \times ... \times I^n_{\mathsf{A}}$ *:* l'ensemble des entrées de A *;*
- $x = (x_1, ..., x_n)$ *dans* I_{A} *: une* entrée de A
- $Input(\mathsf{A}) = \left(I_{\mathsf{A}}, \left(\phi_{I^1_{\mathsf{A}}}, ..., \phi_{I^n_{\mathsf{A}}}\right)\right)$ *:* l'ensemble des entrées effectives de A *;*
- $f_{\mathsf{A}}(x)$ *:* la sortie de A sur x*, notée par la suite* $\mathsf{A}(x)$ *;*
- O_{A} *est alors* l'ensemble des sorties de A *;*
- $Output(\mathsf{A})$ *:* l'ensemble effectif des sorties de A *;*
- $T_{\mathsf{A}}(x)$ *:* le temps d'exécution de A sur x *;*
- *la fonction* T_{A} *définie sur* $\mathbb{N}$ *par :*

$$\mathsf{T}_{\mathsf{A}}(k) = \max\left(T_{\mathsf{A}}(x_1, .., x_n) : (x_1, ..., x_n) \in I_{\mathsf{A}}, t_{I^1}(x_1) \leqslant k, ..., t_{I^n}(x_n) \leqslant k\right)$$

est appelée la complexité de A*. Elle représente le plus long temps de calcul qui puisse être obtenu lors de l'exécution de l'algorithme sur des entrées de taille* k*.*

Introduisons enfin la notion de polynomialité (en temps). La taille des entrées intervient dans l'expression du temps d'exécution d'un algorithme. Cette notion permet ainsi d'en mesurer l'efficacité.
Dans ce cadre, la notation asymptotique O est utilisée pour donner un ordre de grandeur ou une majoration du temps de calcul. Cette notion est adaptée au cadre des ensembles effectifs. Rappelons-en brièvement le sens général dans la définition 1.3.4.

Définition 1.3.4 *Soit n un entier et $\left(I^1, \phi_1\right) ... \left(I^n, \phi_n\right)$ des ensembles effectifs de taille respective $t_{I^1}, ..., t_{I^n}$. Soit f et g deux fonctions réelles positives définies sur $I \subseteq I^1 \times ... \times I^n$, on dit que f* est dominée par *g si et seulement si :*

$$\exists C > 0, \exists N \in \mathbb{N}, \forall\, (x_1, ..., x_n) \in I,$$
$$\max\left(t_{I^1}(x_1), ..., t_{I^n}(x_n)\right) \geqslant N \Rightarrow f(x_1, ..., x_n) \leqslant C g(x_1, .., x_n).$$

On note alors $f = O(g)$.

Ainsi, on peut dire que le temps d'exécution d'un algorithme est en $O(g)$ si asymptotiquement (sur la taille des entrées) ce temps est majoré par la fonction réelle positive g. La définition 1.3.5 introduit cette notion.

Définition 1.3.5 *Considérons un algorithme* A. *Gardons les notations de la définition 1.3.3. Soit g une fonction réelle positive définie sur I_{A}, on dit que le* temps d'exécution de A est en $O\left(g(x_1, ..., x_n)\right)$ *si et seulement si :*

$$\exists C > 0, \exists N \in \mathbb{N} : \forall (x_1, .., x_n) \in I_{\mathsf{A}},$$
$$\max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) \geqslant N \Rightarrow T_{\mathsf{A}}(x_1, ..., x_n) \leqslant C g\left(x_1, ..., x_n\right).$$

Les algorithmes sont alors distingués par le fait que leur temps d'exécution est polynomial ou non (en la taille des entrées). La notion de polynomialité est largement développée dans des ouvrages comme [GJ79], [CLR94] ou [Pap94]. Voici, dans la définition 1.3.6, celle que nous utilisons par la suite.

Définition 1.3.6 *On dit que l'algorithme* A *est* polynomial (en temps) *si et seulement si :*

$$\exists \alpha \geqslant 0, \exists C > 0, \exists N \in \mathbb{N} : \forall (x_1, ..x_n) \in I_{\mathsf{A}},$$
$$\max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) \geqslant N \Rightarrow T_{\mathsf{A}}(x_1, ..., x_n) \leqslant C\left[\max\left(t_{I^i}(x_i), i = 1..n\right)\right]^{\alpha}.$$

On peut alors montrer le lemme 1.3.1.

Lemme 1.3.1 *Si l'algorithme* A *est* polynomial (en temps), *alors :*

$$\exists \alpha \geqslant 0, \exists C > 0, \exists N \in \mathbb{N} : \forall k \geqslant N \Rightarrow \mathsf{T}_{\mathsf{A}}(k) \leqslant C.k^{\alpha}.$$

Preuve. Soit k un entier, la complexité de l'algorithme A est définie par :

$$\mathsf{T}_{\mathsf{A}}(k) = \max\left(T_{\mathsf{A}}(x_1, .., x_n) : (x_1, .., x_n) \in I_{\mathsf{A}}, t_{I^1}(x_1) \leqslant k, ..., t_{I^n}(x_n) \leqslant k\right).$$

$$\begin{array}{rcl} \text{Considérons } D(k) & = & \{(x_1, ..., x_n) \in I_{\mathsf{A}} : t_{I^1}(x_1) \leqslant k, ..., t_{I^n}(x_n) \leqslant k\} \\ & = & \{(x_1, ..., x_n) \in I_{\mathsf{A}} : \max\left(t_{I^1}(x_1), ..., t_{I^n}(x_n)\right) \leqslant k\}. \end{array}$$

L'algorithme A est polynomial en temps, donc il existe α et C constantes strictement positives, et un entier N tels que :

$$\forall (x_1, ..., x_n) \in I_{\mathsf{A}},$$
$$\max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) \geqslant N \Rightarrow \mathsf{T}_{\mathsf{A}}(x_1, ..., x_n) \leqslant C\left[\max\left(t_{I^i}(x_i), i = 1..n\right)\right]^{\alpha}.$$

On écrit alors l'ensemble $D(k)$ sous la forme : $D(k) = D_{<N}(k) \bigcup D_{\geqslant N}(k)$, où :

$$D_{<N}(k) = \{(x_1, ..., x_n) \in D(k) : \max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) < N\}$$
et
$$D_{\geqslant N}(k) = \{(x_1, ..., x_n) \in D(k) : \max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) \geqslant N\}$$

On considère alors le maximum du temps d'exécution sur chacun de ces deux ensembles :

$$M_{<N}(k) = \max\left(T_{\mathsf{A}}(x_1, ..x_n) : (x_1, ..x_n) \in D_{<N}(k)\right)$$

et

$$M_{\geqslant N}(k) = \max\left(T_{\mathsf{A}}(x_1, ..x_n) : (x_1, ..x_n) \in D_{\geqslant N}(k)\right).$$

L'algorithme A étant polynomial :
– il existe un réel strictement positif M tel que : $M_{<N}(k) < M$;
– il existe α et C constantes positives telles que :

$$\forall\, (x_1, .., x_n) \in D_{\geqslant N}(k), \mathsf{T}_{\mathsf{A}}(x_1, ..., x_n) \leqslant C\left[\max\left(t_{I^i}(x_i), i = 1..n\right)\right]^{\alpha}.$$

Ainsi, on a :

$$M_{\geqslant N}(k) \leqslant C.k^{\alpha}$$

Or la complexité de A est :

$$\mathsf{T}_{\mathsf{A}}(k) = \max\left(M_{<N}(k), M_{\geqslant N}(k)\right) \leqslant \max(M, C.k^{\alpha}).$$

Donc, quitte à choisir une constante C plus grande :

$$\exists \alpha \geqslant 0, \exists C > 0, \exists N \in \mathbb{N} : \forall k \geqslant N \Rightarrow \mathsf{T}_{\mathsf{A}}(k) \leqslant C \cdot k^{\alpha}. \qquad \square$$

Un algorithme est dit alors polynomial en temps, si son temps d'exécution peut être majoré par un polynôme (éventuellement multivarié) dont les variables représentent la taille des différentes entrées.

Lemme 1.3.2 *S'il existe un polynôme P dans $\mathbb{R}[X_1, ..., X_n]$ tel que le temps d'exécution de l'algorithme A soit en $O(P(t_1(x_1), ..., t_n(x_n)))$ alors A est polynomial (en temps).*

Preuve.

$$\exists \alpha \geqslant 0, \exists C > 0, \exists N \in \mathbb{N} : \forall (x_1, ..., x_n) \in I_1 \times .. \times I_n,$$
$$\max\left(t_{I^1}(x_1), .., t_{I^n}(x_n)\right) \geqslant N \Rightarrow P(t_1(x_1), ..., t_n(x_n)) \leqslant C \cdot \left[\max\left(t_i(x_i), i = 1..n\right)\right]^{\alpha}. \qquad \square$$

Traitons maintenant du cas des algorithmes probabilistes.

1.3.4 Algorithme probabiliste polynomial

Au sens usuel, un algorithme probabiliste est un algorithme qui fait appel lors de son exécution à une suite de tirages aléatoires. La particularité de ce type d'algorithme est qu'il peut y avoir plusieurs sorties associées à une même entrée ; on considèrera alors l'ensemble des sorties possibles pour une entrée fixée. Par exemple, à une question à caractère décisionnel (notion que nous précisons dans la section 1.3.5), nous pouvons choisir de répondre aléatoirement par "oui" ou "non". Se pose alors la question de l'efficacité d'un tel algorithme ou plus précisément quelle est la probabilité qu'il renvoie le bon résultat ?

Cela dit, un algorithme probabiliste reste déterministe au sens où si les tirages aléatoires renvoient les mêmes résultats, la sortie restera la même.
Ainsi, nous pouvons encore modéliser ce type d'algorithme par une machine de Turing à plusieurs rubans. Il suffit pour cela de réserver un ruban aux tirages aléatoires que l'on peut considérer alors comme partie intégrante des entrées.

Soit A un algorithme probabiliste. Notons r le ruban représentant les entrées aléatoires de A et x les rubans représentant les entrées non aléatoires. Afin de différencier ces deux types de données, on note, s'il existe, $\mathsf{A}(x;r)$ le résultat du calcul de A sur les entrées x et de tirages aléatoires r.
Le caractère aléatoire des algorithmes probabilistes nous incite à redéfinir les notions sous-jacentes aux algorithmes exposées page 17, en tenant compte des réflexions suivantes :

1. On s'intéresse uniquement aux entrées x telles que l'algorithme s'arrête quels que soient les tirages aléatoires r effectués.
2. Pour des entrées x fixées, A renvoie plusieurs sorties $\mathsf{A}(x;r)$. Celles-ci forment un ensemble que l'on peut munir d'une loi de distribution mesurant la probabilité d'obtenir telle sortie plutôt que telle autre.
3. Le temps d'exécution de A sur x est alors le maximum des temps d'exécution d'entrées x sur tous les tirages aléatoires possibles.

Ainsi pour la suite, nous donnons la définition 1.3.7 d'un algorithme probabiliste.

Définition 1.3.7 *Un algorithme probabiliste* A *est une machine de Turing à laquelle on peut associer :*
- *un entier* n *;*
- $n+2$ *ensembles effectifs, notés pour* i *de* 1 *à* n $Input^i(\mathsf{A}) = (I^i_{\mathsf{A}}, \phi_{I^i_{\mathsf{A}}})$, $Random(\mathsf{A}) = (R_{\mathsf{A}}, \phi_{R_{\mathsf{A}}})$, *et* $Output(\mathsf{A}) = (O_{\mathsf{A}}, \phi_{O_{\mathsf{A}}})$. *De plus, on a* $R_{\mathsf{A}} = \{0,1\}^*$ *et* $\phi_{R_{\mathsf{A}}} = Id$ *;*
- I_{A} *une partie de* $I^1_{\mathsf{A}} \times ... \times I^n_{\mathsf{A}}$ *;*
- *une surjection* f_{A} *de* $I_{\mathsf{A}} \times R_{\mathsf{A}}$ *dans* O_{A} *tels que : pour tout* $x = (x_1, ..., x_n)$ *dans* I_{A} *et* r *dans* R_{A} *:*
  - *l'exécution de l'algorithme* A *sur l'entrée* $\left(\phi_{I^1_{\mathsf{A}}}(x_1), ..., \phi_{I^n_{\mathsf{A}}}(x_n); \phi_{R_{\mathsf{A}}}(r)\right)$ *s'arrête,*
  - $\phi_{O_{\mathsf{A}}}(f_{\mathsf{A}}(x_1, ..., x_n; r))$ *représente la sortie issue de cette exécution.*

On note alors $T_{\mathsf{A}}(x_1, ..., x_n; r)$ *le nombre de configurations construites pour obtenir ce résultat.*

Nous appelons :

- $I_{\mathsf{A}} \subseteq I_{\mathsf{A}}^1 \times ... \times I_{\mathsf{A}}^n$ *:* l'ensemble des entrées de A *;*
- $x = (x_1, ..., x_n)$ *dans* I_{A} *: une* entrée de A
- $Input(\mathsf{A}) = \left(I_{\mathsf{A}}, \left(\phi_{I_{\mathsf{A}}^1}, ..., \phi_{I_{\mathsf{A}}^n}\right)\right)$ *:* l'ensemble des entrées effectives de A *;*
- r *: un* tirage aléatoire de A *et par conséquent* R_{A} *est* l'ensemble des tirages aléatoires de A *;*

- $f_{\mathsf{A}}(x;r)$ *:* la sortie de A sur x et de tirage aléatoire r*, notée par la suite* $\mathsf{A}(x;r)$ *;*
- $\mathsf{A}(x) = \{\mathsf{A}(x;r) : r \in R_{\mathsf{A}}\}$ *est l'ensemble des sorties de* A *d'entrée* x*.*
  Dans la proposition 1.3.1, on munira cet ensemble d'une loi de probabilité ;
- $O_{\mathsf{A}} = \bigcup_{x \in I_{\mathsf{A}}} \mathsf{A}(x)$ *est* l'ensemble des sorties de A *;*
- $Output(\mathsf{A})$ *:* l'ensemble effectif des sorties de A *;*
- $T_{\mathsf{A}}(x) = \max\left(T_{\mathsf{A}}(x;r) : r \in R_{\mathsf{A}}\right)$ *:* le temps d'exécution de A sur x*.*
  Cette notion mesure le plus long temps de calcul que l'on puisse obtenir lors de l'exécution de l'algorithme A *sur une entrée* x*. Dans le cas des algorithmes probabilistes, on l'appelle aussi* complexité probabiliste de A *;*
- *la fonction* T_{A} *définie sur* $\mathbb{N}$ *par :*

$$\mathsf{T}_{\mathsf{A}}(k) = \max\left(T_{\mathsf{A}}(x) : t_{I^1}(x_1) \leqslant k, ..., t_{I^n}(x_n) \leqslant k\right)$$

  est appelée la complexité de A*.*
  Là aussi, elle représente le plus long temps de calcul qui puisse être obtenu lors de l'exécution de l'algorithme sur des entrées de taille k*.*

Nous définissons alors de la même façon la polynomialité (en temps) d'un algorithme probabiliste.

Définition 1.3.8 *On dit que l'algorithme* A *est* polynomial (en temps) *si et seulement si :*

$$\exists \alpha \geqslant 0, \exists C > 0, \exists N \in \mathbb{N} : \forall (x_1, ..., x_n) \in I_1 \times ... \times I_n,$$
$$\max\left(t_{I^1}(x_1), ..., t_{I^n}(x_n)\right) \geqslant N \Rightarrow T_{\mathsf{A}}(x_1, ..., x_n) \leqslant C\left[\max\left(t_{I^i}(x_i), i = 1...n\right)\right]^{\alpha}.$$

Dans ce contexte, un algorithme (déterministe) est un cas particulier d'algorithme probabiliste. Il suffit de supposer que R_{A} ne contienne qu'un seul élément pour indiquer qu'aucun tirage aléatoire n'intervient dans le fonctionnement de l'algorithme.
Un algorithme probabiliste ne renvoie pas systématiquement le même résultat et on ne peut pas l'assimiler aussi facilement à une fonction (dont l'image d'un point ne peut varier).
Lorsque l'on choisit une entrée, l'algorithme renvoie plusieurs sorties qui n'ont pas nécessairement la même fréquence d'apparition. Si on voulait assimiler un algorithme probabiliste à une fonction, celle-ci renverrait pour chaque entrée possible un espace probabilisé mesurant la probabilité d'obtenir telle sortie plutôt que telle autre.
Précisons dans la proposition 1.3.1, comment un algorithme probabiliste et une entrée définissent un espace probabilisé.

Proposition 1.3.1 *Soit* A *un algorithme probabiliste polynomial et* x *une entrée de* A*, alors* $\mathsf{A}(x) = \{\mathsf{A}(x;r) : r \in \{0,1\}^{T_{\mathsf{A}}(x)}\}$.
La fonction définie sur $\mathsf{A}(x)$ *par :*

$$\forall y \in \mathsf{A}(x), P_{\mathsf{A},x}(y) = \frac{\#\{r : r \in \{0,1\}^{T_{\mathsf{A}}(x)}, \mathsf{A}(x;r) = y\}}{2^{T_{\mathsf{A}}(x)}}$$

définit une mesure sur $\mathcal{P}\left(\mathsf{A}(x)\right)$*. On appelle alors* $\left(\mathsf{A}(x), P_{\mathsf{A},x}\right)$ *l'*espace probabilisé par (l'exécution de) A sur x*.*

Preuve. Considérons une machine de Turing modélisant A. Nous lui avons choisi pour alphabet l'ensemble $\{0,1\}$. La notation $T_{\mathsf{A}}(x)$ modélise un majorant du nombre de configurations construites lors de l'exécution de A sur une entrée modélisant x. Elle majore le nombre de tirages à pile ou face requis pour exécuter l'algorithme. Donc les tirages aléatoires r appartiennent à l'ensemble $\{0,1\}^{T_{\mathsf{A}}(x)}$. Ces remarques expliquent le choix de cet ensemble.

Montrons que $P_{\mathsf{A},x}$ est une mesure de probabilité sur $\mathsf{A}(x)$.
Il est évident que :
- pour tout y dans $\mathsf{A}(x)$, $P_{\mathsf{A},x}(y)$ positif,
- la somme $\Sigma_{y\in\mathsf{A}(x)}P_{\mathsf{A},x}(y)$ vaut 1 et les ensembles $\{r : r \in \{0,1\}^{T_{\mathsf{A}}(x)}, \mathsf{A}(x;r) = y\}$ où y appartient à $\mathsf{A}(x)$ forment une partition de $\{0,1\}^{T_{\mathsf{A}}(x)}$ de cardinal $2^{T_{\mathsf{A}}(x)}$. □

Cette loi de probabilité a pour support $\mathsf{A}(x)$, elle mesure la fréquence d'apparition de chaque sortie sur tous les résultats de tirages aléatoires possibles.
Dès lors, nous pouvons appréhender le choix d'un élément dans un ensemble fini de loi uniforme comme l'exécution d'un algorithme probabiliste dont l'espace probabilisé est fini uniforme.
Nous traitons de cet exemple dans la définition 1.3.9.

Définition 1.3.9 *Soit E un ensemble effectif fini, considérons l'algorithme 1.*
Algorithme 1 *: Tir_E*

Entrées :	
Sortie :	$x \in E$
1.	$x \longleftarrow E$
2.	***Return*** (x)

On pose :
- *l'ensemble des sorties de $\mathsf{Tir}_{\mathsf{E}}$ d'entrée vide est E, soit :*
$$\mathsf{Tir}_E = E\,;$$
- *la mesure P_{Tir_E}, est une loi uniforme sur E, soit :*
$$\text{pour tout } x \text{ dans } E,\ P_{\mathsf{Tir}_E}(x) = \frac{1}{|E|}\,;$$
- *le temps d'exécution de Tir_E sur une entrée vide est constant, soit :*
$$T_{\mathsf{Tir}_E}() = O(1).$$

L'algorithme 1 de tir uniforme exposé ici est un algorithme idéalisé de par son temps de calcul (ne dépendant pas de la taille de l'ensemble) et de par l'espace probabilisé qu'il renvoie. En effet, la construction d'algorithme de tir uniforme idéal reste un sujet de recherche à part entière (cf [Lub96] ou [Vie03] par exemple).

Les algorithmes probabilistes interviennent en cryptographie, notamment en cryptanalyse pour définir un adversaire (cf 1.4) ; et de manière plus générale en algorithmique pour résoudre des problèmes de nature décisionnelle (cf 1.3.5). Nous les retrouvons dès la fin de la prochaine partie.

1.3.5 Les algorithmes pour résoudre un problème calculatoire

Dans cette section, nous formalisons la notion de problème en informatique. Il en existe de plusieurs types, chacun donnant lieu à des hiérarchies de classes de complexité différentes (cf.

par exemple [GJ79], [Joh90] ou [Pap94] pour les caractères calculatoire et décisionnel, [MR95] pour les caractères probabilistes). Il ne s'agit pas pour nous de discuter de ces différentes classes.

Nous pouvons préciser que par la suite, nous traitons de problèmes calculatoires et décisionnels.

Nous donnons dans une première partie la notion de problème que nous utilisons dans le chapitre 3. Puis nous introduisons la notion d'équivalence entre deux problèmes paramétrés. Enfin, dans une dernière partie, nous illustrons ces notions par des exemples connus.

Problèmes et réductions déterministes

De façon générale, un problème se définit au moyen d'une question. Celle-ci porte sur certains objets qui déterminent les entrées acceptées par cette question ; on les appelle les instances du problème. Par exemple, à la question " Soit n un entier naturel, n est-il premier ? " les instances du problème sont les entiers naturels. La réponse à la question posée peut être du type "oui" ou "non", (comme pour la question précédente), ou plus complexe (comme la réponse à la question "Soit n un entier naturel, quelle est sa décomposition en facteurs premiers ?").
Dans cette partie, nous formalisons ce type de notions. Les définitions de problème et de réduction exposées dans cette section sont inspirées du mémoire de thèse de K. Gjøsteen [Gjø04, p.7-8]. On les retrouve également dans le livre de Garey et Jonhson [GJ79, p.4].
On distingue les problèmes de nature décisionnelle (pour lesquels la sortie est limitée aux réponses "oui" ou "non") des problèmes de nature calculatoire (pour lesquels la sortie est une valeur numérique). La première formalise un problème de décision, la seconde un problème de calcul.

Dans ma thèse, les algorithmes utilisés pour les différentes réductions sont déterministes et portent sur des problèmes calculatoires admettant chacun une unique solution. Ainsi, nous pouvons assimiler la notion de problème à résoudre à celle de fonction à calculer. De plus, ces définitions ne tiennent pas compte de la distribution que l'on peut associer à un ensemble d'instances d'un problème.

On obtient ainsi les définitions 1.3.10 et 1.3.11.

Définition 1.3.10 *Un* problème *est la donnée d'un triplet $\mathcal{P} = (P, S, f)$ où P est un ensemble appelé* ensemble d'instances*, S est un ensemble appelé* ensemble des solutions*, et f est une application de P dans S appelée* solution du problème.

Dans ce cadre, cette définition n'est rien d'autre que la définition d'une fonction appelée solution, dont l'ensemble de définition est appelé ensemble des instances et l'ensemble d'arrivée est appelé ensemble des solutions.

La solution formalise la question posée. Par exemple, si la solution est une application renvoyant la décomposition en produit de facteurs premiers d'un nombre, il s'agira bien d'un problème de décomposition en produit de facteurs premiers. Résoudre le problème consiste alors à savoir calculer efficacement la solution.

Dans ce contexte, l'ensemble des instances permet de savoir quel exemple de problème est considéré. On peut choisir de restreindre le problème de décomposition en produit de facteurs premiers aux entiers de la forme $p \times q$ où p et q sont deux nombres premiers distincts.
On peut aussi se concentrer sur une seule instance. Par exemple, le problème RSA-640 consiste à calculer la décomposition en produit de facteurs premiers du nombre :

$$3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609.$$

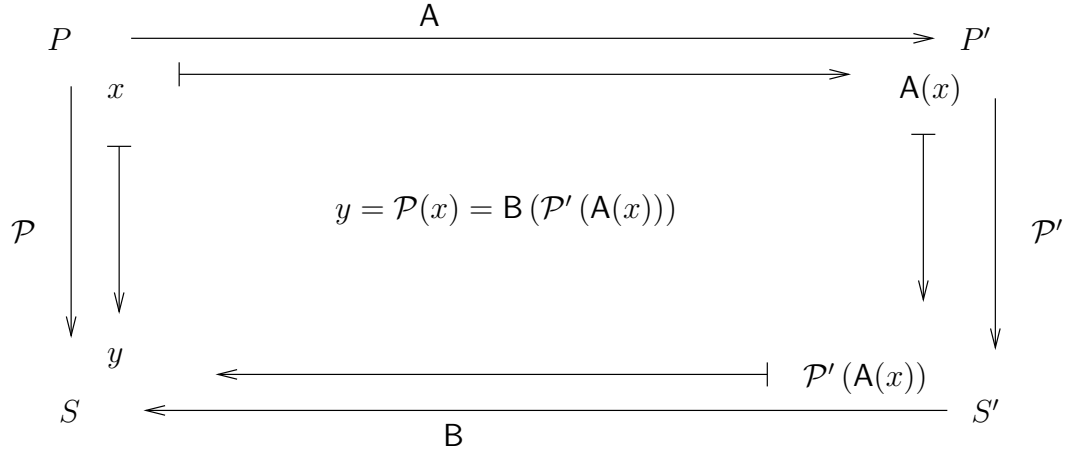
Il a été résolu le 2 novembre 2005 par Bahr, Boehm, Franke et Kleinjung après 5 mois de calculs (cf. [BBFK05]).

Par la suite, on assimile un problème à sa solution et on note $\mathcal{P}(x)$ la solution d'instance x au lieu de $f(x)$.
Un outil important dans la résolution de problème est la réduction (définition 1.3.11), elle permet de classer les problèmes en affirmant qu'un problème est au moins aussi difficile à résoudre qu'un autre problème. Nous introduisons cette notion dans la définition 1.3.11.

Définition 1.3.11 *Soient $\mathcal{P} = (P, S, f)$ et $\mathcal{P}' = (P', S', f')$ deux problèmes, soient A et B deux algorithmes déterministes tels que :*
– A prend en entrée un élément de P et renvoie un élément de P',
– B prend en entrée un élément de S' et renvoie un élément de S.
On dit que (A, B) est une réduction de $\mathcal{P}$ vers $\mathcal{P}'$ *si et seulement si :*

$$\textit{pour tout } x \textit{ dans } P,\ \mathsf{B}(\mathcal{P}'(\mathsf{A}(x))) = \mathcal{P}(x).$$

Elle donne lieu au diagramme commutatif suivant :



Nous nous intéressons alors aux problèmes de tailles de plus en plus grandes : ce sont des problèmes paramétrés.

Problème paramétré et équivalence

Pour nous, l'intérêt d'une réduction est qu'elle soit polynomiale ; c'est à dire que les deux algorithmes qui la composent le soient. Cela suppose que la taille des instances puisse être de plus en plus élevée et tendre vers l'infini.

Ainsi, nous introduisons la notion de réduction pour un problème paramétré par une variable i appartenant à un ensemble effectif infini (I, ϕ_I) de taille t_I (cf. définition 1.3.12).
Nous pourrons alors faire tendre $t_I(i)$ vers l'infini et étudier le temps d'exécution asymptotique des algorithmes composant la réduction dans ce contexte (cf. définition 1.3.14).
Nous pouvons remarquer que l'ensemble (I, ϕ_I) de taille t_I n'a d'intérêt que s'il est du même ordre que la taille des données du problème. On définit d'ailleurs par $t_I(i)$ la taille du problème $\mathcal{P}_i$ (cf. définition 1.3.13).

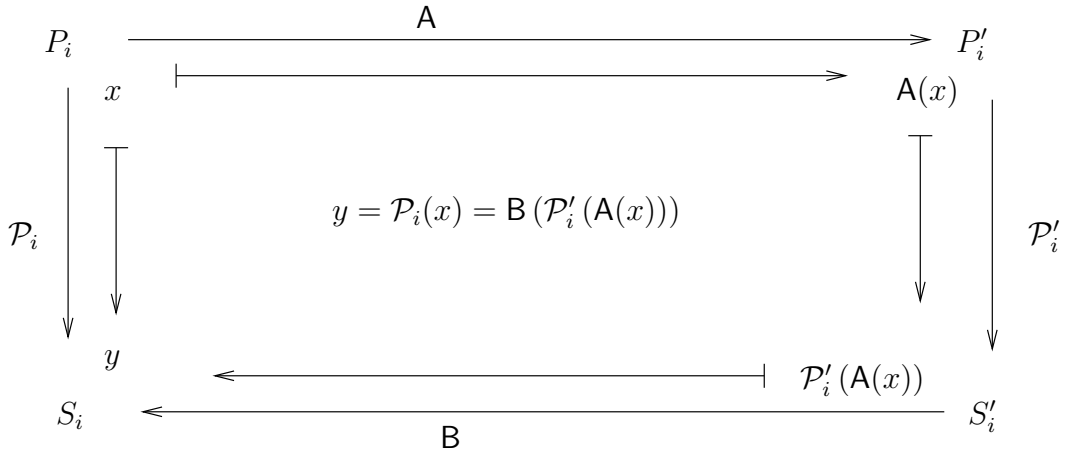
Pour commencer, la définition 1.3.12 étend la notion de réduction entre deux problèmes (définie en 1.3.11) à celle de réduction entre deux problèmes paramétrés. Il est à noter cependant que dans la définition 1.3.12 les algorithmes de cette réduction doivent traiter tous les problèmes indexés par la variable i. La voici :

Définition 1.3.12 *Soit (I, ϕ_I) un ensemble effectif infini de taille t_I, soient $\mathcal{P} = \{\mathcal{P}_i\}_{i \in I}$ et $\mathcal{P}' = \{\mathcal{P}'_i\}_{i \in I}$ deux problèmes paramétrés par I avec $\mathcal{P}_i = (P_i, S_i, f_i)$ et $\mathcal{P}'_i = (P'_i, S'_i, f'_i)$. Soient* A *et* B *deux algorithmes déterministes :*
- A *prend en entrée un élément de l'union des P_i et renvoie un élément de l'union des P'_i, pour i dans I ;*
- B *prend en entrée un élément de l'union des S'_i et renvoie un élément de l'union des S_i, pour i dans I.*

On dit que (A, B) est une réduction de $\mathcal{P}$ vers $\mathcal{P}'$ *si et seulement si :*

$$\textit{pour tout } i \textit{ dans } I, \textit{ pour tout } x \textit{ dans } P_i,\ \mathsf{B}(\mathcal{P}'_i(\mathsf{A}(x))) = \mathcal{P}_i(x).$$

qui donne lieu au diagramme commutatif suivant :



La variable i permet de mesurer la taille des instances d'un problème. On introduit alors la définition 1.3.13 :

Définition 1.3.13 *Soit (I, ϕ_I) un ensemble effectif infini de taille t_I, soit $\{\mathcal{P}_i\}_{i \in I}$ un problème paramétré par i, la* taille du problème $\mathcal{P}_i$ *est $t_I(i)$.*

Ainsi, on dira qu'un problème paramétré se réduit à un autre problème paramétré s'il existe une réduction polynomiale du premier vers le second.

Définition 1.3.14 *Soit (I, ϕ_I) un ensemble effectif infini de taille t_I, soit $\mathcal{P} = \{\mathcal{P}_i\}_{i\in I}$ et $\mathcal{P}' = \{\mathcal{P}'_i\}_{i\in I}$ deux problèmes paramétrés par i avec $\mathcal{P}_i = (P_i, S_i, f_i)$ et $\mathcal{P}'_i = (P'_i, S'_i, f'_i)$. On dit que $\mathcal{P}$* se réduit à *$\mathcal{P}'$ si et seulement si :*
il existe une réduction (A, B) de $\mathcal{P}$ vers $\mathcal{P}'$ telle que A et B soient polynomiaux en la taille des problèmes $\mathcal{P}_i$.

Enfin, on dit de deux problèmes paramétrés se réduisant l'un à l'autre qu'ils sont équivalents comme le précise la définition 1.3.15.

Définition 1.3.15 *Soit (I, ϕ_I) un ensemble effectif infini de taille t_I, soient $\mathcal{P}$ et $\mathcal{P}'$ deux problèmes paramétrés par i dans I. On dit que $\mathcal{P}$* et $\mathcal{P}'$ sont équivalents *si et seulement si $\mathcal{P}$ se réduit à $\mathcal{P}'$ et $\mathcal{P}'$ se réduit à $\mathcal{P}$.*

Exemples de problèmes paramétrés

Les problèmes exposés dans cette partie apparaissent naturellement lors de l'étude de la sécurité du cryptosystème ElGamal. Commençons par le plus difficile d'entre eux : le problème du logarithme discret. Ensuite nous présenterons le problème calculatoire puis décisionnel de Diffie-Hellman.

–Le problème du logarithme discret– Exposons ce problème dans le cadre général d'un groupe abélien fini noté additivement. Grâce à la méthode de Pohlig-Hellman ([PH78]), la décomposition d'un groupe fini en produit de sous-groupes cycliques permet de restreindre l'étude de ce problème aux groupes cycliques finis.
En fixant un générateur P d'un groupe cyclique, nous pouvons donner une définition mathématique du logarithme discret d'un élément Q en base P.

Définition 1.3.16 *Soit G un groupe cyclique fini de générateur P, soit Q un élément de G ;* le logarithme discret de Q en base P *est l'unique entier m dans $[\![0, |\mathsf{G}| - 1]\!]$ tel que $Q = mP$. On note alors $m = \log_P(Q)$.*

Ainsi, le logarithme discret d'un élément dépend aussi du générateur choisi comme référence de base. Voici dans ce cadre général, la définition 1.3.17 du problème du logarithme discret, en reprenant les notations de la définition 1.3.10 :

Définition 1.3.17 *Soit G un groupe cyclique fini de générateur P,* le problème du logarithme discret dans (G, P) *est $\mathcal{L}og(G, P) = (G, [\![0, |G| - 1]\!], \log_P)$.*

Le choix du groupe G, puis de la famille de groupes G_i sous-jacente, permet de définir des problèmes de logarithme discret paramétrés sur des exemples plus concrets.
Dans un premier temps, le cryptosystème ElGamal a été présenté sur un groupe multiplicatif de cardinal $p - 1$. La cryptographie s'est alors penchée sur le problème du logarithme discret sur les groupes $\mathbb{F}_p^*$

Exemple 1.3.5 ***Problème du logarithme discret sur un groupe multiplicatif $\mathbb{F}_p^*$.***
Pour p un nombre premier, soit x un générateur du groupe multiplicatif $\mathbb{F}_p^*$, le problème du logarithme discret dans $(\mathbb{F}_p^*, x)$ consiste à calculer pour tout y dans $\mathbb{F}_p^*$, l'unique entier n dans $[\![0, (p-1) - 1]\!]$ tel que $y = x^n$.
On le note plus succinctement $\mathcal{L}og(\mathbb{F}_p^*, x) = (\mathbb{F}_p^*, [\![0, p-2]\!], \log_x)$.

Exemple 1.3.6 ***Problème du logarithme discret sur les groupes multiplicatifs*** $\mathbb{F}_p^*$**.**
Considérons $I = \{(p,x) : p \text{ premier}, x \in \mathbb{F}_p^*, \langle x\rangle = \mathbb{F}_p^*\}$ *et* $t_I(p,x) = t_2(p)$*. On appelle* problème du logarithme discret dans les $\mathbb{F}_p^*$ *le problème de logarithme discret dans* $(\mathbb{F}_p^*, x)$ *paramétré par* (p,x) *dans* I*.*

Brassard a montré dans [Bra79] que, plus largement, le problème décisionnel associé à $\mathcal{L}og(\mathbb{F}_p^*, x)$ paramétré par (p,x) (pour p premier) est dans la classe de problèmes $NP \cap CoNP$ (voir par exemple [Pap94] ou [vzGG99] pour l'introduction de ces notions).
Je ne connais à ce jour aucun algorithme polynomial (en $t_2(p)$) résolvant ce problème paramétré avec une probabilité non négligeable.
Je ne connais non plus aucune preuve de l'existence ou de l'absence d'un tel algorithme.

Exemple 1.3.7 ***Problème du logarithme discret sur une courbe elliptique sur*** $\mathbb{F}_p$**.**
Considérons une courbe elliptique cyclique $E_{a,b}(\mathbb{F}_p)$ *de cardinal* $\mathbf{N}$ *et* P *un générateur de* $E_{a,b}(\mathbb{F}_p)$*, le* problème du logarithme discret dans $(E_{a,b}(\mathbb{F}_p), P)$ *consiste à calculer pour tout* Q *dans* $E_{a,b}(\mathbb{F}_p)$*, l'unique entier* n *dans* $[\![0, \mathbf{N}-1]\!]$ *tel que* $Q = nP$*.*
On le note plus succinctement $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), P) = (E_{a,b}(\mathbb{F}_p), [\![0, \mathbf{N}-1]\!], \log_P)$*.*

Exemple 1.3.8 ***Problème du logarithme discret sur les courbes elliptiques paramétrées par*** I**.** *Considérons un ensemble effectif infini* (I, ϕ) *tel que :*

$$I \subset \{(p,a,b,P) : p \text{ premier}, (a,b) \in \mathbb{F}_p^2, 4a^3 + 27b^2 \neq 0, P \in E_{a,b}(\mathbb{F}_p), \langle P\rangle = E_{a,b}(\mathbb{F}_p)\}.$$

On appelle problème du logarithme discret sur les courbes elliptiques paramétrées par I *le problème du logarithme discret dans* $(E_{a,b}(\mathbb{F}_p), P)$ *paramétré par* (p,a,b,P) *dans* I*.*

Dans la section 3.3, nous allons montrer que le problème du logarithme discret sur les courbes elliptiques sur $\mathbb{F}_p[\varepsilon]$ (un anneau de nombres duaux de grande caractéristique) dont la projection sur $\mathbb{F}_p$ est cyclique et de cardinal non divisible par p, est équivalent au problème du logarithme discret sur les courbes elliptiques sur un corps cyclique $\mathbb{F}_p$ de grande caractéristique et dont le cardinal est non divisible par p.
Plus précisément, nous définirons un ensemble I_ε tel que le problème de logarithme discret sur les courbes elliptiques $(E_{a,b}(\mathbb{F}_p), \overline{P})$ et $(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ paramétrées par (p,a,b,P) dans I_ε sont équivalents.
Cette équivalence fait l'objet du corollaire 3.3.1 de la sous-section 3.3.1.

–**Le problème calculatoire de Diffie-Hellman**– Considérons encore une fois un groupe cyclique de générateur P. Ce problème consiste à calculer le composé de Diffie-Hellman en base P de deux éléments du groupe, dont la définition mathématique est exposée ci-dessous.

Définition 1.3.18 *Soit* G *un groupe cyclique fini de générateur* P*, soient* A *et* B *deux éléments de* G*, le* composé de A et B en base P dit de Diffie-Hellman *est l'unique élément* C *dans* G *tel que :*

$$C = \alpha\beta P \text{ avec } A = \alpha P \text{ et } B = \beta P.$$

On note alors $C = CDH_P(A,B)$*.*

Là aussi, le composé de Diffie-Hellman de deux éléments dépend du générateur choisi comme référence de base. Voici dans ce cadre général, la définition 1.3.19 du problème calculatoire de Diffie-Hellman (noté par la suite CDH) :

Définition 1.3.19 *Soit G un groupe cyclique fini de générateur P,* le problème calculatoire de Diffie-Hellman dans (G, P) *est noté* $C_{DH}(G, P) = (G^2, G, CDH_P)$.

Illustrons ce problème dans le cadre des courbes elliptiques définies sur un corps fini, en commençant par une courbe fixée puis en continuant par une famille de courbes paramétrées par la caractéristique du corps et les coefficients de l'équation de Weierstrass.

Exemple 1.3.9 ***Problème CDH sur une courbe elliptique sur*** $\mathbb{F}_p$**.** *Considérons une courbe elliptique cyclique $E_{a,b}(\mathbb{F}_p)$ de cardinal $\mathbf{N}$ et P un générateur de $E_{a,b}(\mathbb{F}_p)$, le* problème calculatoire de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p), P)$ *consiste à calculer pour tout couple (A, B) de points de $E_{a,b}(\mathbb{F}_p)$, l'unique point C de G tel que $C = \alpha\beta P$ avec $A = \alpha P$ et $B = \beta P$.*
On le note plus succinctement $C_{DH}(E_{a,b}(\mathbb{F}_p), P) = (E_{a,b}(\mathbb{F}_p)^2, E_{a,b}(\mathbb{F}_p), CDH_P)$.

Exemple 1.3.10 ***Problème CDH sur les courbes elliptiques paramétrées par*** I**.** *Considérons un ensemble effectif infini (I, ϕ) tel que :*

$$I \subset \{(p, a, b, P) : p \text{ premier}, (a, b) \in \mathbb{F}_p^2, P \in E_{a,b}(\mathbb{F}_p), \langle P \rangle = E_{a,b}(\mathbb{F}_p)\}.$$

On appelle problème calculatoire de Diffie-Hellman sur les courbes elliptiques paramétrées par I *le problème calculatoire de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p), P)$ paramétré par (p, a, b, P) dans I.*

Un algorithme qui s'attaque au problème CDH prend donc en entrée un couple de points d'une courbe elliptique (et un générateur de cette courbe), il renvoie un point de cette même courbe. On mesure sa probabilité de succès à sa capacité de renvoyer le bon point c'est à dire le composé de Diffie Hellman du couple d'entrée. Précisons ces idées avec la définition 1.3.20.

Définition 1.3.20 *Considérons le problème calculatoire de Diffie-Hellman sur les courbes elliptiques paramétrées par un ensemble effectif infini (I, ϕ) de taille t, un* algorithme $\mathcal{A}$ qui traite ce problème *est un algorithme probabiliste prenant en entrées :*
– une courbe elliptique $E_{a_i,b_i}(\mathbb{F}_{p_i})$ de générateur P_i pour i dans I paramétrée par i ;
– deux points A et B de $E_{a_i,b_i}(\mathbb{F}_{p_i})$;
et renvoyant un point de $E_{a_i,b_i}(\mathbb{F}_{p_i})$.
Sa probabilité de succès *est une fonction définie sur* $\mathbb{N}$ *par :*

$$\mathsf{Succ}_{\mathcal{A}}^{CDH}(k) =$$

$$Pr\left(C = CDH(A, B) : i \xleftarrow{u} t^{-1}(k), (A, B) \xleftarrow{u} E_{a_i,b_i}(\mathbb{F}_{p_i})^2, C \leftarrow \mathcal{A}(p_i, a_i, b_i, P_i, A, B)\right)$$

$$et$$

$$\mathsf{Succ}_{\mathcal{A}}^{CDH}(k) = 0 \text{ si } t^{-1}(k) = \varnothing.$$

Le problème restera difficile si aucun algorithme ne parvient à le résoudre avec une probabilité de succès non négligeable.

Rappelons ce que signifie qu'une fonction est négligeable.

Définition 1.3.21 *Une fonction f positive définie sur* $\mathbb{N}$ *est* négligeable *si et seulement si :*

$$\forall n \in \mathbb{N}, \exists k_n \in \mathbb{N}, k > k_n \Rightarrow f(k) < \frac{1}{k^n}$$

On obtient ainsi la notion de problème difficile, en particulier pour le problème CDH.

Définition 1.3.22 *Soit I un ensemble effectif infini. On dira que* le problème CDH sur les courbes elliptiques paramétré par I est difficile à résoudre *si et seulement si tout algorithme probabiliste polynomial $\mathcal{A}$ traitant ce problème a une probabilité de succès négligeable, soit :*

$$\forall n \in \mathbb{N}, \exists k_n \in \mathbb{N}, k > k_n \Rightarrow \mathsf{Succ}_{\mathcal{A}}^{CDH}(k) < \frac{1}{k^n}.$$

Le problème CDH, comme celui du logarithme discret, est un problème calculatoire. La nature de ce problème nous permettra d'établir le même type d'équivalence entre les problèmes de logarithme discret ; à savoir, l'équivalence des problèmes CDH :
- sur les courbes elliptiques sur les corps premiers $\mathbb{F}_p$ de grande caractéristique et dont le cardinal est non divisible par p,
- et sur les courbes elliptiques sur $\mathbb{F}_p[\varepsilon]$ correspondantes.

Cette équivalence fait l'objet du corollaire 3.3.2 de la sous-section 3.3.1.

1.3.6 Les distingueurs pour résoudre un problème décisionnel

Les notions développées dans cette section sont inspirées des notes de cours de Katz [Kat04] (ou du livre qu'il a co-écrit [KL07]) et proviennent des travaux présentés dans l'article [Gol99] de Goldreich.
Nous traitons là de problèmes de décision.
Il ne s'agit plus ici de répondre à une question calculatoire mais de répondre par "oui" ou par "non". Un problème décisionnel est alors un problème dont les solutions sont repésentées par exemple par 0 pour "oui " et 1 pour "non" comme le formalise la définition 1.3.23.

Définition 1.3.23 *Un* problème décisionnel *est un problème dont l'ensemble des solutions est* $\{0,1\}$.

En exemple, nous définissons le problème décisionnel de Diffie Hellman (noté DDH par la suite).

–Le problème décisionnel de Diffie-Hellman– La question est la suivante :

> " Voici un triplet (A,B,C) d'éléments d'un groupe G de générateur P,
> est ce que C est le composé de Diffie-Hellman de A et B en base P ? "

Voici présentée en 1.3.24, la définition formelle du problème DDH sur un groupe abélien fini cyclique.

Définition 1.3.24 *Soit G un groupe cyclique fini de générateur P,* le problème décisionnel de Diffie-Hellman dans (G,P) *est $D_{DH}(G,P) = (G^3, \{0,1\}, DDH_P)$ où :*

$$\begin{array}{rccl} DDH_P : & G^3 & \longrightarrow & \{0,1\} \\ & (A,B,C) & \mapsto & \left\{ \begin{array}{ll} 1 & si : \exists(\alpha,\beta) \in \mathbb{N}^2 : A = \alpha P, B = \beta P, C = \alpha\beta P \\ 0 & sinon \end{array} \right. \end{array}$$

Illustrons ce problème dans le cadre des courbes elliptiques définies sur un corps fini, en commençant par une courbe fixée puis en continuant par une famille de courbes paramétrées par la caractéristique du corps et les coefficients de l'équation de Weierstrass.

Exemple 1.3.11 ***Problème DDH sur une courbe elliptique sur*** $\mathbb{F}_p$. *Considérons une courbe elliptique cyclique* $E_{a,b}(\mathbb{F}_p)$ *de cardinal* $\mathbf{N}$ *et* P *un générateur de* $E_{a,b}(\mathbb{F}_p)$, *le* problème décisionnel de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p), P)$ *consiste à déterminer pour tout triplet* (A, B, C) *de points de* $E_{a,b}(\mathbb{F}_p)$ *s'il existe deux entiers* α *et* β *tels que* $A = \alpha P$, $B = \beta P$ *et* $C = \alpha\beta P$. *On le note plus succinctement* $D_{DH}(E_{a,b}(\mathbb{F}_p), P) = (E_{a,b}(\mathbb{F}_p)^3, \{0,1\}, DDH_P)$.

Un algorithme résolvant ce type de problème est un algorithme (pouvant être) probabiliste et renvoyant 1 ou 0, représentant respectivement "oui" ou "non". On appelle ce type d'algorithme un distingueur introduit de façon plus générale par la définition 1.3.25.
Nous utilisons ensuite cette notion pour définir un distingueur du problème DDH.

Définition 1.3.25 *Soit* (X, p) *et* (X, p') *deux espaces probabilisés finis. Un* distingueur des distributions (X, p) et (X, p') *est un algorithme probabiliste :*
- *qui prend en entrée un élément de* X *;*
- *qui renvoie* 0 *ou* 1.

L'avantage de ce distingueur est alors le réel $\mathsf{Adv}_{\mathcal{D}}$ *défini par :*

$$\mathsf{Adv}_{\mathcal{D}} = \left| Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{p} X\right) - Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{p'} X\right)\right|$$

L'avantage mesure la différence de probabilité que le distingueur renvoie 1 lorsque un élément est issu de la première distribution plutôt que de la seconde.

Ainsi, s'il renvoie 1 avec la même probabilité lorsque un élément est issu de la première distribution que de la seconde, cela signifie bien qu'il ne différencie pas les deux espaces probabilisés et son avantage est nul.
Par contre, s'il renvoie 1 avec une probabilité égale à 1 lorsqu'un élément est issu de la première distribution et avec une probabilité nulle lorsqu'il est issu de la seconde, cela signifie bien qu'il différencie parfaitement ces deux espaces probabilisés et son avantage vaut 1.
On remarque que l'avantage d'un distingueur est nécessairement compris entre 0 et 1.

Illustrons cette notion sur un exemple simple :

Exemple 1.3.12 *Soit* $X = [\![1, 6]\!]$ *où la première distribution* u *est uniforme sur* X *et où la seconde* p' *est nulle pour les nombre impairs et égale à* $\frac{1}{3}$ *pour les nombres pairs.*
Considérons le distingueur $\mathcal{D}$ *renvoyant* 1 *pour un entier pair et* 0 *pour un entier impair, son avantage est alors :*

$$\begin{aligned} \mathsf{Adv}_{\mathcal{D}} &= \left| Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{u} [\![1, 6]\!]\right) - Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{p'} [\![1, 6]\!]\right)\right| \\ &= |0,5 - 1| = 0,5 \end{aligned}$$

Son avantage est donc constant égal à $0,5$.

On peut remarquer qu'un distingueur de ces deux distributions ne peut pas avoir un avantage supérieur à $0,5$ *dans la mesure où exactement la moitié des éléments de* X *ont une probabilité nulle de provenir de la seconde distribution.*

Cette notion de distingueur de deux distributions permet de définir un distingueur d'un problème décisionnel en choisissant les espaces probabilisés adéquats.

Concernant le problème décisionnel DDH sur un groupe G de générateur P, nous choisissons les triplets d'éléments de ce groupe.
Il s'agit alors de distinguer parmi ces triplets ceux qui sont de la forme (A, B, C) avec C composé de Diffie Hellman de A et B en base P des autres triplets. Ces triplets sont appelés triplets de Diffie Hellman, comme l'indique la définition 1.3.26

Définition 1.3.26 *Soit G un groupe cyclique de générateur P, un triplet (A, B, C) dans G^3 est un* triplet de Diffie Hellman de (G, P) *si et seulement s'il existe deux entiers α et β tels que $A = \alpha P$, $B = \beta P$ et $C = \alpha\beta P$.*

La définition 1.3.27 précise alors ce qu'est un distingueur pour le problème DDH sur un groupe G de générateur P. Son objectif sera de reconnaître un triplet de Diffie-Hellman d'un triplet quelconque. On définit alors :

Définition 1.3.27 *Soit G un groupe cyclique fini de générateur P. Un* distingueur $\mathcal{D}$ du problème DDH sur G de base P *est un algorithme probabiliste polynomial qui prend en entrée un triplet (A, B, C) avec A, B, C trois éléments de G, et qui renvoie un bit.*

On mesure alors son avantage à sa capacité à reconnaître un triplet de Diffie-Hellman d'un triplet quelconque comme le précise la définition 1.3.28.

Définition 1.3.28 *Soit G un groupe fini cyclique de générateur P et $\mathcal{D}$ un distingueur du problème DDH sur G de base P. Considérons les ensembles finis suivants :*
- *$Rand(G, P) = \{(\alpha P, \beta P, \gamma P) : (\alpha, \beta, \gamma) \in [\![0, |G| - 1]\!]^3\}$*
- *$DH(G, P) = \{(\alpha P, \beta P, \alpha\beta P) : (\alpha, \beta) \in [\![0, |G| - 1]\!]^2\}$.*

*L'*avantage de $\mathcal{D}$ sur le problème DDH dans G de base P *est*

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{DDH}(G,P)} = \left| Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{u} Rand(G, P)\right) - Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{u} DH(G, P)\right)\right|$$

Pour pouvoir effectuer une étude asymptotique de la résolution d'un problème décisionnel par un distingueur, nous devons considérer toutes ces notions pour des entrées de taille croisssante. On considère alors qu'un distingueur (de famille de distributions) est efficace si son avantage n'est pas négligeable (en la taille des entrées).
Pour cela nous devons définir un distingueur d'une famille de distributions et définir son avantage comme une fonction en la taille des données.

Définition 1.3.29 *Soit $((X_k, p_k))_{k\in\mathbb{N}}$ et $((X_k, p'_k))_{k\in\mathbb{N}}$ deux suites d'espaces probabilisés finis. Un* distingueur des familles de distributions $((X_k, p_k))_{k\in\mathbb{N}}$ et $((X_k, p'_k))_{k\in\mathbb{N}}$ *est un algorithme probabiliste polynomial :*
- *qui prend en entrée un élément de l'union disjointe des X_k de taille associée $t(x) = k$ pour x dans X_k ;*
- *qui renvoie 0 ou 1.*

L'avantage de ce distingueur est alors la fonction $\mathsf{Adv}_{\mathcal{D}}$ définie sur $\mathbb{N}$ par :

$$\mathsf{Adv}_{\mathcal{D}}(k) = \left| Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{p_k} X_k\right) - Pr\left(\mathcal{D}(x) = 1 : x \xleftarrow{p'_k} X_k\right)\right|$$

La notion d'avantage négligeable permet alors de définir des ensembles calculatoirement indiscernables comme résistants à tout distingueur.
Ce type de propriété appliquée à un distingueur du problème DDH sur une famille de groupes paramétrés est introduit dans la définition 1.3.30.

Définition 1.3.30 *Soit (I, ϕ_I) un ensemble effectif infini de taille t paramétrant une famille $((G_i, P_i))_{i\in I}$ de groupes finis cycliques G_i de générateur P_i.*
Notons $(Rand_k)_{k\in\mathbb{N}}$ et $(DDH_k)_{k\in\mathbb{N}}$ les suites d'espaces probabilisés suivantes :

$$Rand_k = \left\{(\alpha P_i, \beta P_i, \gamma P_i) : i \leftarrow t^{-1}(k), (\alpha, \beta, \gamma) \xleftarrow{u} [\![0, |G_i| - 1]\!]^3\right\}$$

$$DDH_k = \left\{(\alpha P_i, \beta P_i, \alpha\beta P_i) : i \leftarrow t^{-1}(k), (\alpha, \beta) \xleftarrow{u} [\![0, |G_i| - 1]\!]^2\right\}.$$

Un distingueur du problème DDH paramétré par I *est un distingueur dont l'ensemble des entrées est l'union disjointe des couples (G_i, P_i) pour i dans I de taille associée t.*
*L'*avantage de $\mathcal{D}$ sur ce problème *est :*

$$\mathsf{Adv}_{\mathcal{D}}^{DDH}(k) = |Pr(\mathcal{D}(x) = 1 : x \leftarrow Rand_k) - Pr(\mathcal{D}(x) = 1 : x \leftarrow DDH_k)|.$$

On dit que ces suites sont calculatoirement indiscernables *si et seulement si :*
pour tout distingueur $\mathcal{D}$ du problème DDH paramétré par I, on a :

$$\forall n \in \mathbb{N}, \exists k_n \in \mathbb{N}, k > k_n \Rightarrow \left|\mathsf{Adv}_{\mathcal{D}}^{DDH}(k)\right| < \frac{1}{k^n}.$$

On dit alors que le problème DDH paramétré par I est difficile à résoudre

Si aucun distingueur n'est efficace pour déterminer avec une probabilité non négligeable si un triplet est de Diffie-Hellman, nous dirons que le problème est difficile.
On dit alors d'un problème DDH paramétré qu'il est *facile à résoudre* si et seulement s'il n'est pas difficile à résoudre. Autrement dit, s'il existe un distingueur qui discerne avec une probabilité non négligeable un triplet de Diffie-Hellman d'un triplet quelconque.

Nous cherchons à montrer dans la section 3.3 que le problème $D_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ paramétré par (p, a, b, P) est facile à résoudre. Cette propriété est valable pour (p, a, b, P) dans un ensemble effectif infini que nous définirons le moment voulu (page 73). La preuve consiste alors :
– à exhiber un distingueur de ce problème paramétré ;
– à vérifier qu'il s'exécute en temps polynomial ;
– à en étudier l'avantage pour $(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$;
– et enfin à montrer que cet avantage est non négligeable.

1.4 Cryptographie

La cryptographie est la science du secret [Ste98]. Elle voit le jour de façon pragmatique, bien avant Jésus Christ (cinq siècles avant [Sin99, p.20,p.25]), avec la transmission d'informations militaires et la protection de secrets industriels. Il est alors question de communiquer entre différents états-majors sur les stratégies à mettre en œuvre. Bien entendu, à cette époque les moyens restent rudimentaires, mais certains problèmes se posent déjà :
– comment s'assurer que les informations ne tombent pas aux mains de l'ennemi ;
– comment s'assurer qu'elles arrivent à bon port ;
– comment s'assurer que les informations obtenues proviennent bien de son expéditeur ?

La sécurité des données est donc déjà à l'ordre du jour. L'expéditeur doit trouver un moyen de cacher l'information et se mettre d'accord au préalable avec les destinataires éventuels sur le mode opératoire. Ensuite, il faut trouver un mode de transmission de l'information et réfléchir à son intégrité et sa confidentialité en cas d'interception. La sécurité de l'information dépend alors des individus qui y ont accès : expéditeur, coursiers et destinataires. De plus, si on souhaite garantir le secret d'une information à un seul destinataire, l'expéditeur doit trouver autant de modes opératoires qu'il a de correspondants et en trouver de nouveaux s'il soupçonne qu'un procédé (ou une clé) est divulgué(e).
Ainsi, la confiance accordée aux individus détenant l'information à un moment donné et la quantité de correspondants et d'informations posent déjà un problème de protection du secret. Ce second critère va bien évidemment prendre des proportions énormes avec les inventions successives de l'ordinateur puis d'Internet.

Au fil du temps des spécialistes vont se pencher sur ces questions et inventer des procédés de plus en plus élaborés, qui finiront par être révélés un jour ou l'autre. Cette histoire est développée par exemple dans [Ste98] ou [Sin99] ([Dub00] pour une vision globale).
Face à ces contraintes, les scientifiques cherchent alors des chiffres difficiles à inverser même si l'on connait le procédé permettant de cacher l'information (les prémisces [Ker83]). C'est ainsi qu'apparaît, dans [DH76], l'idée de système de chiffrement à clé publique où :
- les modes de chiffrement et de déchiffrement dépendent d'un paramètre appelé la clé ;
- il est facile d'utiliser les processus de chiffrement et de déchiffrement lorsque l'on se donne une clé ;
- par contre, on ne peut pas retrouver le processus de déchiffrement lorsque l'on connait seulement celui de chiffrement ; en particulier il est difficile de retrouver la clé.

Ce dernier critère va aussi permettre de rendre public le processus de chiffrement, chaque utilisateur va disposer d'une clé publique et d'une clé secrète et privée : pour établir une connexion sécurisée avec un autre utilisateur, il lui suffira de créer une clé partagée à l'aide de sa clé privée et de la clé publique de son interlocuteur.
Pour rendre impossible une recherche exhaustive (qui consiste à tester toutes les clés), le nombre de clés doit être suffisament grand pour assurer une sécurité calculatoire ; ce type de cryptosystème va alors devenir, avec l'apparition d'Internet, un moyen infatiguable de créer des canaux sécurisés entre les utilisateurs.
La cryptographie est tombée dans le domaine public : on la retrouve désormais dans tous les foyers pour peu que l'on ait un téléphone portable, une carte bancaire, un ordinateur, un décodeur...

La section 1.4 introduit dans la partie 1.4.1 cette notion de cryptosystème à clef publique ; puis, dans la partie 1.4.2, expose différents scénarii d'attaques.

1.4.1 Cryptographie à clef publique

On trouve dans la littérature beaucoup de définitions voisines de système de chiffrement à clé publique ou de cryptosystème à clé publique. Le principe y est toujours le même : utiliser une paire d'algorithmes calculables en un temps polynomial dont l'un est tenu secret pour le seul destinataire et l'autre est rendu public pour tous les expéditeurs potentiels. De plus, pour que le système soit efficace, il faut :

- que l'algorithme secret inverse facilement l'algorithme public : cela permet de retrouver le message d'origine ;
- qu'il soit (idéalement) impossible de trouver l'algorithme secret à partir de l'algorithme public : cela permet d'assurer la confidentialité de l'information. Si l'on considère que le temps de recherche de cet algorithme est illimité, le caractère fini des ensembles manipulés par un ordinateur rend intraitable cette condition. Elle est alors remplacée par : il est très difficile de trouver l'algorithme secret à partir de l'algorithme public en temps limité.

Whitfield Diffie et Martin E. Hellman sont les premiers à introduire cette idée auprès de la communauté scientifique en 1976 (cf. [DH76] et [Sin99]). Les paires d'algorithmes sont indexées par des clés qui sont générées aléatoirement. Ainsi les algorithmes publics et secrets laissent place aux couples de clés publique et privée. Dans [DH76], Diffie et Hellman proposent déjà un exemple de système public de distribution de clés, bien connu aujourd'hui sous le nom de protocole de Diffie-Hellman.

On retrouve ensuite des définitions différentes selon le point de vue de l'auteur, l'usage qu'il veut en faire, le degré de précision qu'il donne de certains paramètres. Ainsi, par exemple un système de chiffrement est défini par l'intermédiaire tantôt d'algorithmes, tantôt de machines de Turing (cf. [BSS05] et [Ver06]). Certains auteurs détaillent les dépendances entre certains paramètres, comme par exemple dans [GM84] les clairs sont engendrés par une machine de Turing probabiliste prenant pour entrée seulement le paramètre de sécurité.

La définition 1.4.1 formalise la notion que nous utiliserons par la suite ; elle précise tous les termes sous-jacents à cette notion ainsi que les dépendances des paramètres. Pour chaque algorithme, nous avons détaillé l'ensemble de ses entrées.

Définition 1.4.1 *Un* système de chiffrement à clé publique *est la donnée de trois algorithmes polynomiaux* $\mathsf{K}, \mathsf{E}, \mathsf{D}$ *tels que :*

- K *est un algorithme probabiliste :*
  - *qui prend en entrée un entier k de taille k (cf. page 15), soit*
    $$\mathit{Input}(\mathsf{K}) = \mathbb{1}^* ;$$
  - *qui renvoie un couple* $(\mathsf{pk}, \mathsf{sk})$*, dont les éléments sont appelés respectivement* clé publique *et* clé privée de sécurité k.

  L'algorithme K *est appelé* algorithme de génération de clés.
  L'entrée k est appelée paramètre de sécurité.

  On note alors $\mathsf{P_K}$ *et* $\mathsf{S_K}$ *les algorithmes de génération de clés publiques et privées correspondants.*
  Il s'agit des premières et secondes projections de l'algorithme de génération de clés K.

- E *est un algorithme probabiliste ou déterministe :*
  - *qui prend en entrée un paramètre de sécurité k, une clé publique* pk *de sécurité k et un élément m, appelé* clair*. Pour* pk *clé publique de sécurité k, on note* $M_{\mathsf{E}}(k, \mathsf{pk})$ *l'ensemble des clairs que l'algorithme* E *peut chiffrer avec en entrée k pour paramètre de sécurité et* pk *pour clé publique. Cet ensemble est appelé l'*ensemble des clairs associés à la clé publique pk de sécurité k. *On a ainsi :*
    $$I(\mathsf{E}) = \left\{(k, \mathsf{pk}, m) : k \in \mathbb{N}^*, \mathsf{pk} \in \mathsf{P_K}(1^k), m \in M_{\mathsf{E}}(k, \mathsf{pk})\right\}.$$
  - *qui renvoie un élément appelé* chiffré de m par la clé publique pk.
    On note également :

$$C_{\mathsf{E}}(k, \mathsf{pk}) = \bigcup_{m \in M_{\mathsf{E}}(k,\mathsf{pk})} \mathsf{E}(1^k, \mathsf{pk}, m).$$

*Cet ensemble est appelé l'*ensemble des chiffrés associés à la clé publique pk de sécurité k.

L'algorithme E *est appelé* algorithme de chiffrement.

- D *est un algorithme déterministe :*
  - *qui prend en entrée un paramètre de sécurité* k*, une clé privée* sk *de sécurité* k *de clé publique associée* pk *et un chiffré* c *de* $C_{\mathsf{E}}(k, \mathsf{pk})$*, soit :*
    $$I(\mathsf{D}) = \left\{(k, \mathsf{sk}, c) : k \in \mathbb{N}^*, \exists \mathsf{pk} \in \mathsf{P_K}(1^k) : (\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}(1^k), c \in C_{\mathsf{E}}(k, \mathsf{pk})\right\} ;$$
  - *qui renvoie un clair* m *de* $M_{\mathsf{E}}(k, \mathsf{pk})$.

L'algorithme D *est appelé* algorithme de déchiffrement.

De plus, les algorithmes K, E *et* D *doivent vérifier :*

$$\forall k \in \mathbb{N}^*, \forall (\mathsf{pk}, \mathsf{sk}) \in \mathsf{K}(1^k), \forall m \in M_{\mathsf{E}}(k, \mathsf{pk}), \forall c \in \mathsf{E}(1^k, \mathsf{pk}, m), \mathsf{D}(1^k, \mathsf{sk}, c) = m.$$

La dernière propriété exprime le fait que l'algorithme de déchiffrement muni de la clé privée sk inverse bien l'algorithme de chiffrement muni de la clé publique correspondante pk.
Par contre, cette définition ne tient aucun compte de la sécurité d'un tel cryptosystème. Par exemple, elle ne spécifie pas qu'il doit être difficile de retrouver la clé privée à partir de la clé publique en temps limité. Pourtant, un cryptosystème ne vérifiant pas cette propriété devient inutile. Il s'agit là d'une première condition nécessaire à l'obtention d'un bon système de chiffrement ; il en existe d'autres plus précises et plus fines, comme la difficulté de retrouver le clair à partir d'un chiffré ou d'en obtenir une quelconque information.
Nous développons ces différents critères de sécurité dans la section 1.4.2.

1.4.2 Critères de sécurité

En effet, les objectifs d'un cryptanalyste lorqu'il s'attaque à un cryptosystème peuvent être différents. Il peut, par exemple, :
- vouloir être capable de déchiffrer tous les messages chiffrés par ce système ;
- s'intéresser au déchiffrement des messages émis par un seul utilisateur ;
- ou aux messages reçus par cet utilisateur ;
- ou même uniquement à la correspondance de deux utilisateurs ;
- chercher à déchiffrer un seul message ;
- ou chercher une information quelconque sur un message donné. Par exemple,

- "Y a-t-il des consonnes dans ce message ?"

- "Non !"

- "Alors, elle a dû répondre par "oui" ou par "non" et elle lui a sûrement répondu "oui"."

Les conditions dans lesquelles ce type d'attaque a lieu peuvent être différentes. Puisqu'il s'agit de cryptosystème à clef publique, le cryptanalyste a accès au système de chiffrement et aux clés publiques des utilisateurs. Ainsi, il peut par exemple engendrer des clés ou chiffrer des messages destinés à un utilisateur. Mais on peut supposer qu'il peut aussi faire déchiffrer un certain nombre de messages. C'est ce qui peut se produire par exemple si un utilisateur s'absente de son écran lors d'une pause sans vérouiller sa session : un individu peut usurper temporairement son identité.
Ainsi, on définit un type d'attaquant par :

– l'objectif qu'il s'est fixé ;
– les moyens dont il dispose pour l'atteindre.

Objectifs d'un adversaire

Pour définir l'objectif d'un adversaire, on décrit le scénario correspondant. Celui-ci ressemble à une confrontation entre le concepteur du cryptosystème et l'attaquant où le premier soumet un "défi " au second. Cette mise en scène est formalisée par une expérience aléatoire (définie de manière générale dans la section 1.2). Selon le scénario, un calcul de probabilité permet alors de mesurer l'efficacité de l'adversaire à atteindre son objectif.
Dans cette partie, nous allons décrire deux types d'objectifs différents :
– l'inversion d'un message quelconque. Un cryptosystème résistant à ce type d'adversaire est alors appelé One-Way (signifiant *à sens unique* en anglais, car l'adversaire s'attaque à cette propriété du chiffrement) et l'on dit qu'un adversaire cherchant à inverser un message quelconque est de type OW ;
– la recherche d'une information quelconque sur le clair. Un cryptosystème résistant à ce type d'adversaire vérifie la propriété d' Indistinguability (appelée indistiguabilité ou indiscernabilité en français) et l'on dit qu'un adversaire cherchant une information quelconque sur le clair est de type IND.

One-Wayness : Voici le défi lancé par le concepteur du cryptosystème à tout attaquant :

" A partir d'une clé publique quelconque,
et d'un cryptogramme quelconque,
es-tu capable de me donner le clair correspondant ?"

Il s'agit bien là de l'objectif : "inverser un message quelconque". Le scénario est alors le suivant : pour un paramètre de sécurité k,
– le générateur de clés fournit un couple de clés de sécurité k ;
– un message est choisi uniformément dans l'ensemble des clairs chiffrables par ce couple de clés ;
– ce message est chiffré par le cryptosystème avec la clé secrète obtenue précédemment ;
– le cryptogramme et la clé publique sont communiqués à l'adversaire qui renvoie un clair correspondant.
L'adversaire est alors efficace si le clair qu'il renvoie correspond avec une probabilité non négligeable au clair choisi uniformément dans la deuxième étape.

La définition 1.4.2 donne alors une formalisation d'un adversaire OW.

Définition 1.4.2 *Un* adversaire OW $\mathcal{A}$ contre un système de chiffrement à clef publique PKC *est un algorithme probabiliste polynomial :*
– *qui prend en entrée un paramètre de sécurité k, une clé publique* pk *de sécurité k et un chiffré associé à la clé publique* pk *;*
– *qui renvoie un clair chiffrable avec cette clé publique, autrement dit un élément de* $M_{\mathsf{E}}(k, \mathsf{pk})$.

Le scénario associé à ce type d'adversaire est formalisé dans la définition 1.4.3.

Définition 1.4.3 *Soit* $\mathsf{PKC} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *un cryptosystème à clé publique, soit $\mathcal{A}$ un adversaire OW contre* PKC*, soit k un entier. L'expérience aléatoire* $OW^{\mathcal{A}}_{\mathsf{PKC}}(k)$ *est :*

$$\begin{array}{rrcl} OW^{\mathcal{A}}_{\mathsf{PKC}}(k) : & (\mathsf{pk},\mathsf{sk}) & \longleftarrow & \mathsf{K}(1^k) \\ & m_\star & \longleftarrow & M_{\mathsf{E}}(k,\mathsf{pk}) \\ & C & \longleftarrow & \mathsf{E}(1^k,\mathsf{pk},m_\star) \\ & m^\star & \longleftarrow & \mathcal{A}\left(1^k,\mathsf{pk},C\right) \end{array}$$

On peut remarquer que la clé secrète générée dans la première étape n'intervient pas dans la suite de l'expérience. Nous pourrions alors remplacer la première ligne par :

$$\mathsf{pk} \longleftarrow \mathsf{P_K}(1^k).$$

Mais nous garderons la première notation qui permet, par exemple, de ne pas redéfinir plus loin la clé secrète sk correspondant à la clé publique pk ; et de garder à l'esprit la provenance de la clé publique pk.

Ensuite, on mesure l'efficacité d'un adversaire OW à inverser le système par sa probabilité de succès définie en 1.4.4.

Définition 1.4.4 *Soit $\mathcal{A}$ un adversaire OW contre un système de chiffrement à clé publique* PKC*,* sa probabilité de succès sur le système de chiffrement PKC *est la fonction* $\mathsf{Succ}^{\mathsf{OW}}_{\mathcal{A}/\mathsf{PKC}}$ *définie sur* $\mathbb{N}$ *par :*

$$\mathsf{Succ}^{\mathsf{OW}}_{\mathcal{A}/\mathsf{PKC}}(k) = Pr\left(m^\star = m_\star : ((\mathsf{pk},\mathsf{sk}), m_\star, C, m^\star) \leftarrow OW^{\mathcal{A}}_{\mathsf{PKC}}(k)\right).$$

Indistinguability : Voici le défi lancé par le concepteur du cryptosystème à tout attaquant :

"A partir d'une clé publique quelconque,
es-tu capable de me donner deux clairs
dont tu saurais déterminer
au vu du cryptogramme lequel des deux j'ai chiffré ?"

Le challenge pour l'adversaire est donc le suivant : il doit produire deux messages clairs (par exemple les messages “oui” et “non”) dont il sait qu'il aura une chance (non négligeable) de reconnaitre lequel a été chiffré.
Pour formaliser ce type d'attaque, on fait appel à deux algorithmes différents : le premier produit les deux clairs au vu de la clé publique et le second détermine lequel des deux a été chiffré au vu du cryptogramme, en renvoyant un bit indiquant sa réponse : par exemple si le bit est 0 ; cela signifie qu'à mon avis tu as chiffré le premier message ; si le bit est 1 ; c'est que je pense que tu as chiffré le second. Pour prendre en compte le fait qu'il ne s'agisse que d'un seul attaquant, le premier algorithme produit également une chaîne de bits (s) qui est une entrée du second algorithme. Ainsi le premier algorithme "communique" des données (celles de son choix) au second, comme par exemple les messages qu'il a choisis. On obtient ainsi la définition 1.4.5 du type d'attaquant qui nous intéresse :

Définition 1.4.5 *Un* adversaire IND $\mathcal{A}$ contre un système de chiffrement à clé publique PKC *est un couple $(\mathcal{A}_1,\mathcal{A}_2)$ d'algorithmes probabilistes polynomiaux, tels que :*
- *$\mathcal{A}_1$ prend en entrée un paramètre de sécurité k, une clé publique* pk *de sécurité k ;*
- *$\mathcal{A}_1$ renvoie deux clairs associés à la clé publique* pk *et une chaîne de bits s ;*
- *$\mathcal{A}_2$ prend en entrée la chaîne de bits s, un chiffré c associé à la clé publique* pk *;*
- *$\mathcal{A}_2$ renvoie un bit :*
  - *0 signifiant : “ Mon choix est le premier clair” ;*

– 1 *signifiant : " Mon choix est le second clair".*

Le scénario d'attaque est alors le suivant : pour un paramètre de sécurité k,
– le cryptosystème génère un couple de clés ;
– la clé publique pk est transmise à l'adversaire par l'intermédiaire de l'algorithme n° 1 ;
– celui-ci envoie deux clairs m_0, m_1 chiffrables par cette clé, ainsi qu'une chaîne de bits s qui sera transmise à l'algorithme n° 2 ;
– un bit δ_* est choisi uniformément ;
– le clair m_{δ_*} est chiffré par le cryptosystème avec la clé publique pk ;
– le cryptogramme correspondant est transmis à l'algortihme n° 2, ainsi que la chaîne de bits s. Il en déduit un second bit δ^*.
Le but de l'adversaire est alors que δ^* soit égal à δ_*.
Ce scénario est formalisé dans la définition 1.4.6.

Définition 1.4.6 *Soit* $\mathsf{PKC} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *un cryptosystème à clé publique, soit* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *un adversaire IND contre* PKC*, soit* k *un entier non nul, l'expérience aléatoire* $IND^{\mathcal{A}}_{\mathsf{PKC}}(k)$ *est :*

$$
\begin{array}{rrcl}
IND^{\mathcal{A}}_{\mathsf{PKC}}(k) : & (\mathsf{pk}, \mathsf{sk}) & \leftarrow & \mathsf{K}(1^k) \\
& (m_0, m_1, s) & \leftarrow & \mathcal{A}_1(1^k, \mathsf{pk}) \\
& \delta_\star & \stackrel{u}{\leftarrow} & \{0,1\} \\
& C & \leftarrow & \mathsf{E}(1^k, \mathsf{pk}, m_{\delta_\star}) \\
& \delta^\star & \leftarrow & \mathcal{A}_2(1^k, C, s)
\end{array}
$$

Dans le cas de l'indistinguabilité, la mesure de l'efficacité de l'adversaire n'est pas si simple que dans le cas de l'inversion. En effet, le choix uniforme d'un bit donne lieu à un espace équiprobable de cardinal 2 et en choisissant uniformément un second bit, il y a une chance sur deux que l'on obtienne le même résultat que précédemment.
L'efficacité d'un adversaire IND est alors mesurée par la différence entre la probabilité que $\delta^* = \delta_*$ et $0,5$, comme le précise la définition 1.4.7.

Définition 1.4.7 *Soit* $\mathcal{A}$ *un IND-adversaire contre un système de chiffrement à clé publique* PKC*,* son avantage sur le cryptosystème $\mathsf{PKC} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ *est la fonction* $\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A}/\mathsf{PKC}}$ *définie sur* $\mathbb{N}$ *par :*

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A}/\mathsf{PKC}}(k) = \left|2Pr\left(\delta^\star = \delta_\star : ((\mathsf{pk}, \mathsf{sk}), (m_0, m_1, s), \delta_\star, C, \delta^\star) \leftarrow IND^{\mathcal{A}}_{\mathsf{PKC}}(k)\right) - 1\right|.$$

Cette définition de l'avantage mesure bien l'efficacité de l'adversaire car si celui-ci renvoie uniformément 0 ou 1 indépendamment des résultats précédents, son avantage sera nul.
Autrement dit, s'il n'a aucune idée du résultat, il n'a aucun avantage sur le système.
Cette propriété est établie dans le lemme 1.4.1.

Lemme 1.4.1 *Soit* $\mathcal{A}$ *un IND-adversaire contre un système de chiffrement à clé publique* PKC*, soit* k *un entier non nul et* $((\mathsf{pk}, \mathsf{sk}), (m_0, m_1, s), \delta_\star, C, \delta^\star)$ *sortie de l'expérience aléatoire* $IND^{\mathcal{A}}_{\mathsf{PKC}}(k)$*, on a :*

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A}/\mathsf{PKC}}(k) = \left|Pr\left(\delta^\star = 0 | \delta_\star = 0\right) - Pr\left(\delta^\star = 0 | \delta_\star = 1\right)\right|.$$

L'avantage d'un IND-adversaire $\mathcal{A}$ *contre un cryptosystème* PKC *est nul si et seulement si les variables aléatoires* $\delta_\star$ *et* $\delta^\star$ *issues de* $IND^{\mathcal{A}}_{\mathsf{PKC}}$ *sont indépendantes.*

Preuve Soit k un entier non nul, tous les évènements suivants sont considérés suite à l'expérience aléatoire $IND^{\mathcal{A}}_{\mathsf{PKC}}(k)$. On a $\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A}/\mathsf{PKC}}(k) = |2Pr(\delta^\star = \delta_\star) - 1|$ avec :

$$\begin{aligned} Pr(\delta^\star = \delta_\star) &= Pr(\delta^\star = 0 \cap \delta_\star = 0) + Pr(\delta^\star = 1 \cap \delta_\star = 1) \\ &= Pr(\delta^\star = 0 | \delta_\star = 0) Pr(\delta_\star = 0) + Pr(\delta^\star = 1 | \delta_\star = 1) Pr(\delta_\star = 1) \\ &= Pr(\delta^\star = 0 | \delta_\star = 0) \frac{1}{2} + (1 - Pr(\delta^\star = 0 | \delta_\star = 1) \frac{1}{2} \end{aligned}$$

Ainsi on a :

$$\begin{aligned} &\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{A}/\mathsf{PKC}}(k) \\ &= \left| 2 \left(Pr(\delta^\star = 0 | \delta_\star = 0) \frac{1}{2} + (1 - Pr(\delta^\star = 0 | \delta_\star = 1) \frac{1}{2} \right) - 1 \right| \\ &= |Pr(\delta^\star = 0 | \delta_\star = 0) - Pr(\delta^\star = 0 | \delta_\star = 1)| \end{aligned}$$

Alors l'avantage de $\mathcal{A}$ est nul si et seulement si $Pr(\delta^\star = 0 | \delta_\star = 0) = Pr(\delta^\star = 0 | \delta_\star = 1)$.
Or on a : $Pr(\delta^\star = 0) = \frac{1}{2} Pr(\delta^\star = 0 | \delta_\star = 0) + \frac{1}{2} Pr(\delta^\star = 0 | \delta_\star = 1)$.
Donc l'avantage de $\mathcal{A}$ est nul si et seulement si :

$$Pr(\delta^\star = 0) = Pr(\delta^\star = 0 | \delta_\star = 0) = Pr(\delta^\star = 0 | \delta_\star = 1).$$

Cela signifie que les variables $\delta_\star$ et $\delta^\star$ sont indépendantes. □

Cette notion d'indistinguabilité est bien reliée à l'idée d'obtenir de l'information sur le message. Par exemple si un adversaire est capable de déceler à partir d'un cryptogramme si le clair correspondant contient des consonnes, il saura distinguer un clair comportant des consonnes d'un clair n'en contenant pas. L'algorithme n° 1 renverra alors deux clairs (le premier contenant des consonnes et le second n'en contenant pas) et une chaîne de bits s indiquant cette information (par exemple le couple de clairs). Puis, au vu du cryptogramme, l'algorithme n° 2 désignera quel clair a été chiffré.
La sécurité sémantique, qui traite de ces questions, a été introduite en 1984 dans [GM84]. Cet article établit déjà l'importance de la notion d'indistinguabilité. Elle donnera lieu à la notion de sécurité prouvée introduite dans [Bel98], insistant sur l'importance des temps de calcul algorithmiques (nous en parlerons dans la section 1.5). Des notions plus précises encore donneront lieu aux sécurités exacte, concrète et pratique développées respectivement dans [BR96], [Gal04] et [Poi02c].

Moyens d'un adversaire

Définissons maintenant les différents types de moyens pouvant être à la disposition de l'attaquant. Nous en distinguons deux :
- ceux qui peuvent faire déchiffrer les messages de leur choix (sauf bien entendu ceux qui leur sont donnés dans le scénario d'attaque) ;
- ceux qui ne peuvent pas.

Par contre, l'un comme l'autre peuvent faire chiffrer les messages qu'ils souhaitent puisque le système de chiffrement est public.
On parle alors d'attaque à chiffrés choisis (CCA : Chosen Ciphertext Attack) dans le premier cas et d'attaque à clairs choisis (CPA : Chosen Plaintext Attack) dans le second cas.

Il existe encore des distinctions plus précises : par exemple on détermine le nombre de déchiffrements permis à l'attaquant, ou l'on permet à l'attaquant d'attendre le déchiffrement d'un chiffré pour demander le déchiffrement du second. Nous n'aborderons pas ce type de précisions que le lecteur pourra retrouver par exemple dans [PP04] ou [RS92].

On appelle alors :
- CPA : un attaquant qui ne peut faire déchiffrer un message. Il s'agit du type d'attaquant le moins puissant. Il peut tout de même chiffrer les clairs de son choix par l'intermédiaire de l'algorithme de chiffrement public E ;
- CCA : un attaquant qui peut faire déchiffrer les messages de son choix. Pour formaliser ce type d'attaquant on peut supposer qu'il ait accès à un algorithme permettant de déchiffrer un message (autre que ceux qui lui sont donnés) en temps constant. Il s'agit alors plus précisément d'un attaquant CCA2 ; le type d'attaquant le plus puissant.

A partir de l'objectif et des moyens de l'adversaire, sont définis différents types d'attaquants du plus puissant IND-CCA2 au moins puissant OW-CPA. D'autres types d'adversaire, une synthèse de ces différentes notions et de leurs relations est exposée dans [BDPR98].

Dans la section 1.5, nous allons introduire la notion de preuves de sécurité qui permet de montrer qu'un système de chiffrement est robuste (ou non) face à un adversaire de type fixé.

1.5 Preuves de sécurité

La notion de sécurité sémantique a été introduite pour la première fois par Goldwasser et Micali dans l'article [GM84] : elle exprime le fait qu'aucune information sur le clair ne doit être décelable à partir du chiffré (hormis sa longueur).
Un tel critère impose, en particulier, que le système de chiffrement soit probabiliste : le chiffrement d'un clair ne doit pas systématiquement donner le même résultat, bien que le déchiffrement (à l'aide de la clé secrète appropriée) de tous les cryptogrammes correspondants doit retourner toujours le même clair. Idéalement pour qu'aucune information ne "transpire" d'un chiffré, il faudrait que tout message de l'ensemble des cryptogammes ait la même probabilité de provenir d'un clair plutôt que d'un autre. C'est l'analogue de la condition du secret parfait exprimée pour les chiffrés à clé secrète par C. E. Shannon dans [Sha49].

La sécurité prouvée (appelée aussi sécurité réductionniste) introduite dans [Bel98] a pour principe de ramener l'attaque d'un cryptosystème à un problème sous-jacent supposé difficile à résoudre. En pratique, il s'agit de construire des réductions algorithmiques successives entre le scénario d'attaque d'une part et la résolution de ce problème sous-jacent d'autre part. Le caractère calculatoire ou décisionnel de ces réductions dépend de la nature de l'attaque : par exemple, un scénario OW conduira à un problème calculatoire tandis qu'un scénario IND conduira à un problème décisionnel. Ainsi les réductions seront de type calculatoire dans le premier cas et décisionnel dans le second.
Pour qu'il soit possible de résoudre le problème en un temps raisonnable à partir du scénario d'attaque, les coûts de ces réductions doivent être en temps polynomial et avec une probabilité non négligeable de succès (dans le cas de réductions probabilistes).
Ainsi, si un adversaire est capable de mener une attaque contre un cryptosystème en temps polynomial et avec une probabilité non négligeable, les réductions nous permettront de l'utiliser pour résoudre un problème a priori difficile.

Une preuve de sécurité consiste alors à construire une réduction entre le scénario d'attaque et la résolution du problème sous-jacent. Ces preuves restent difficiles à formaliser et plusieurs types de rédaction ont vu le jour. La technique de "suite de jeux" a été largement développée ces dernières années, (voir par exemple [Poi02a], [Gal04],[Lag05]).

Plus précisément, elle consiste, dans un premier temps, à formaliser l'attaque de l'adversaire et la résolution du problème sous-jacent par des "jeux" auxquels est associé à chacun une probabilité de réussite. Ces jeux faisant appel à des algorithmes probabilistes, déterminant chacun des espaces probabilisés et des variables aléatoires, nous les assimilerons à des expériences aléatoires.
Il s'agit ensuite d'écrire une suite finie d'expériences aléatoires et d'évènements associés, dont on peut comparer entre elles les probabilités de succès. De plus, la première expérience doit formaliser l'attaque et la dernière la résolution du problème sous-jacent.
Toutefois, il est difficile de s'assurer de la rigueur de telles preuves comme le souligne D. Vergnaud dans [Ver06, p.91].

Pour obtenir plus de rigueur dans ce type de rédaction, dans [Sho06] V. Shoup détermine en quelque sorte les règles du jeu. Il y introduit trois types de *transitions* possibles pour passer d'une expérience aléatoire à une autre :

1. Transition de type 1 : la transition supposée indiscernable.
   Le premier type de transition consiste à changer le mode d'obtention d'une variable : par exemple, lors de l'expérience $\mathsf{Exp_i}$ la variable x s'obtient en exécutant l'algorithme $\mathcal{A}$, alors qu'au cours de l'expérience $\mathsf{Exp_{i+1}}$, x s'obtient en exécutant l'algorithme $\mathcal{B}$.
   Il faut alors comparer la distribution de cette variable dans chacune des deux expériences. Le cas idéal voudrait que ces distributions soient identiques. Dans ce cas, les probabilités d'un évènement S suite à $\mathsf{Exp_i}$ et $\mathsf{Exp_{i+1}}$ sont égales et il s'agit d'une transition de type 3.
   Si ce n'est pas le cas, on peut construire un distingueur $\mathcal{D}$ de ces deux distributions :

   $$\text{il renvoie 1 si l'évènement } S \text{ est vérifié et 0 sinon.}$$

   Ainsi, il renvoie 1 avec une probabilité $P(S : \mathsf{Exp_i})$ si la variable x est issue de $\mathcal{A}$ et $P(S : \mathsf{Exp_{i+1}})$ si elle est issue de $\mathcal{B}$ (cf. notation page 11). Son avantage est donc

   $$\mathsf{Adv}_{\mathcal{D}} = \left| P(S : \mathsf{Exp_i}) - P(S : \mathsf{Exp_{i+1}}) \right|.$$

   En supposant qu'il n'existe aucun algorithme probabiliste polynomial qui puisse déterminer avec une probabilité non négligeable si un élément provient de la première distribution plutôt que de la seconde, les probabilités $P(S : \mathsf{Exp_i})$ et $P(S : \mathsf{Exp_{i+1}})$ sont "quasi-identiques" ou indiscernables : leur différence est négligeable ;
2. Transition de type 2 : la transition à défaillance exceptionnelle.
   Le deuxième type de transition sert à ne pas tenir compte d'"erreurs mineures". C'est le cas lorsque l'on remplace le mode d'obtention d'une variable par un autre mode qui peut être défaillant.
   On considère F cet évènement, si l'évènement S que l'on étudie suite à chacune de ces deux expériences Exp_i et Exp_{i+1} est identique lorsque F ne se produit pas ; alors on a, d'après le lemme de la différence (cf [Sho06, p.3]) :

   $$\left| Pr(S : \mathsf{Exp_i}) - Pr(S : \mathsf{Exp}_{i+1}) \right| \leqslant Pr(F).$$

Ainsi pour que les probabilités de S suite à Exp_i et Exp_{i+1} soient proches, il suffit que la probabilité que l'évènement F se produise soit négligeable. Autrement dit, l'évènement F ne doit arriver "presque jamais" ;

3. Transition de type 3 : la transition invisible.

   Le dernier type de transition peut prendre plusieurs tournures. Il concerne toutes les transitions ne modifiant pas les lois des variables aléatoires.

   Il s'agit, par exemple, du changement de nom d'une variable, du changement de mode d'obtention d'une variable par une méthode équivalente ou bien encore du déplacement au cours d'une expérience aléatoire de l'affectation d'une variable dont la loi est indépendante des variables aléatoires définies précédemment. Ces modifications ont pour but de faciliter la compréhension de la preuve. Pour ce type de transition, les variables introduites au cours des deux expériences successives doivent être introduites de manière équivalente et suivre la même loi. Si l'on considère le même évènement suite à ces expériences, les probabilités que celui-ci se réalise sont alors identiques.

Nous utiliserons des suites d'expériences aléatoires afin d'établir les preuves de sécurité de la section 4.3. Remarquons enfin que ce type de rédaction ne permet pas d'exposer certaines preuves (cf. [Ver06, p.92]).

Chapitre 2

Géométrie Algébrique

L'objectif de cette section est d'introduire et étudier les courbes elliptiques définies par une cubique de Weierstrass à coefficients dans l'anneau $\mathbb{F}_q[\varepsilon]$.

Dans la partie 2.1, nous étudions ces courbes dans le cas d'un anneau local et nous montrons qu'elles sont munies d'une structure de groupe dont nous donnons explicitement les lois d'addition.

Ensuite, dans la partie 2.2 nous en déduisons quelques résultats généraux concernant l'anneau local $\mathbb{F}_q[\varepsilon]$ étudié en 1.1.2.

Enfin, dans la partie 2.3, nous établissons les formules d'addition sur une courbe elliptique sur $\mathbb{F}_q[\varepsilon]$ qui nous serviront à calculer effectivement la somme de deux points dans les sections suivantes.

2.1 Courbes elliptiques dans le cas d'un anneau local

Proposition 2.1.1 *Soit A un anneau dans lequel 6 est inversible, soient a et b deux éléments de A tels que $4a^3 + 27b^2$ soit inversible dans A ; la courbe elliptique E d'équation*

$$y^2z = x^3 + axz^2 + bz^3$$

est munie d'une unique structure de schéma en groupes sur $Spec(A)$ dont l'élément neutre est $O = [0 : 1 : 0]$.

Preuve. Les éléments 6 et $4a^3 + 27b^2$ sont inversibles dans l'anneau A, donc l'équation $y^2z = x^3 + axz^2 + bz^3$ définit une courbe elliptique E sur A. Le théorème 2.1.2 de Katz et Mazur [KM85, p.63] assure que le choix d'une section permet de munir $E|A$ d'une structure de schéma en groupes et son unicité est assurée par le corollaire 6.6 de Mumford, Fogarty et Kirwan [MFK94, p.117]. □

Notations. Dans cette section R désigne un anneau local de corps résiduel K de caractéristique différente de 2 et 3. Notons π la projection canonique de R dans K. Nous désignons par a et b des éléments de R tels que $\pi(4a^3 + 27b^2) \neq 0$ et nous noterons $E_{a,b}$ le schéma en groupes sur $Spec(R)$ d'équation $y^2z = x^3 + axz^2 + bz^3$ et d'élément neutre O.

Les R-schémas $Spec(R)$ et $Spec(K)$ sont déterminés par $Id : R \to R$ et $\pi : R \to K$ respectivement. Ainsi, $E_{a,b}(R)$ et $E_{a,b}(K)$ sont respectivement les ensembles des R-morphismes de schémas de $Spec(R)$ dans $E_{a,b}$ et de $Spec(K)$ dans $E_{a,b}$.
Ceux-ci sont déterminés par :
- la donnée d'un point $[x : y : z]$ de $\mathbb{P}^2(R)$ vérifiant $y^2z - x^3 - axz^2 - bz^3 = 0$ pour les éléments de $E_{a,b}(R)$,
- la donnée d'un point $[x : y : z]$ de $\mathbb{P}^2(K)$ vérifiant $y^2z - x^3 - \pi(a)xz^2 - \pi(b)z^3 = 0$ pour les éléments de $E_{a,b}(K)$.

Ainsi, pour a et b dans R, on a :

$$E_{a,b}(K) = E_{\pi(a),\pi(b)}(K). \tag{2.1}$$

Lemme 2.1.1 *L'application*

$$\begin{array}{rccc} \pi_{E_{a,b}(R)} : & E_{a,b}(R) & \to & E_{a,b}(K) \\ & [x : y : z] & \mapsto & [\pi(x) : \pi(y) : \pi(z)] \end{array}$$

est un morphisme de groupes.

Preuve. On utilise le foncteur contravariant qui, à un $R-$schéma T, associe le groupe $E_{a,b}(T)$. En particulier, celui-ci envoie le $R-$morphisme de schéma déterminé par π sur l'application $\pi_{E_{a,b}(R)}$. Ainsi, $\pi_{E_{a,b}(R)}$ est un morphisme de groupes. □

L'objet de la proposition 2.1.2 est de donner des relations explicites permettant de calculer la somme de deux éléments de $E_{a,b}(R)$. Dans le cas particulier où R est le corps K ces relations diffèrent selon les valeurs des deux éléments (cf. [Sil85]). Pour P et P' dans $E_{a,b}(K)$:
- si $P = O$ ou $P' = O$: $P + P' = P'$ ou P respectivement ;
- si $P' = -P$ (soit $P = [x : y : z]$ et $P' = [x : -y : z]$) : $P + P' = O$;
- si $P' = P$: on utilise les relations obtenues par construction de la tangente ;
- si $P' \neq P$ et $P' \neq -P$: on utilise les relations obtenues par construction de la corde.

La proposition 2.1.2 généralise ces relations dans le cas où l'anneau R de corps résiduel K est une algèbre locale. De plus, elle restreint le nombre de cas à deux selon que les projections des deux éléments dans $E_{a,b}(K)$ sont égales ou pas.

Proposition 2.1.2 *Si R est une algèbre locale, pour tout $P = [x : y : z]$, $P' = [x' : y' : z']$ dans $E_{a,b}(R)$, on a :*

1. *si $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$, alors $P + P' = [p_1 : q_1 : r_1]$ avec*

I.
$$\begin{array}{rl} p_1 = & y^2x'\,z' - z\,x\,y'^2 - a\,(z\,x' + x\,z')\,(z\,x' - x\,z') + (2\,y\,y' - 3\,b\,z\,z')\,(z\,x' - x\,z') \\ q_1 = & y\,y'\,(z'\,y - z\,y') - a\left(x\,y\,z'^2 - z^2x'\,y'\right) + (-2\,a\,z\,z' - 3\,x\,x')\,(x'\,y - x\,y') \\ & -3\,b\,z\,z'\,(z'\,y - z\,y') \\ r_1 = & (z\,y' + z'\,y)\,(z'\,y - z\,y') + (3\,x\,x' + a\,z\,z')\,(z\,x' - x\,z') \end{array}$$

2. *soit U le sous-ensemble de $E_{a,b}(R) \times E_{a,b}(R)$ défini par $(\pi(p_2), \pi(q_2), \pi(r_2)) \neq (0, 0, 0)$ où*

II.
$$\begin{array}{rl} p_2 = & (y\,y' - 6\,b\,z\,z')\,(x'\,y + x\,y') + \left(a^2z\,z' - 2\,a\,x\,x'\right)(z\,y' + z'\,y) - 3\,b\left(x\,y\,z'^2 + z^2x'\,y'\right) \\ & -a\left(y\,z\,x'^2 + x^2y'\,z'\right) \\ q_2 = & y^2y'^2 + 3\,ax^2x'^2 + \left(-a^3 - 9\,b^2\right)z^2z'^2 - a^2\,(z\,x' + x\,z')^2 - 2\,a^2z\,x\,z'\,x' \\ & + (9\,b\,x\,x' - 3\,a\,b\,z\,z')\,(z\,x' + x\,z') \\ r_2 = & (y\,y' + 3\,b\,z\,z')\,(z\,y' + z'\,y) + (3\,x\,x' + 2\,a\,z\,z')\,(x'\,y + x\,y') + a\left(x\,y\,z'^2 + z^2x'\,y'\right). \end{array}$$

L'ensemble U contient les couples (P, P') tels que $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P')$; et pour tout (P, P') dans U, on a $P + P' = [p_2 : q_2 : r_2]$.

Preuve. D'après la proposition 2.1.1, la courbe elliptique $E_{a,b}$ est munie d'une structure de schéma en groupes d'élément neutre O. L'anneau R est local, donc il vérifie les conditions i.) et ii.) données dans l'article de Lenstra [Len86, p.104]. En particulier, la somme de deux éléments de $E(R)$ est donnée par les relations I, II, III données à la fin de l'article de Lange et Ruppert [LR85]. Les polynômes p_1, q_1, r_1 et p_2, q_2, r_2 sont obtenus à partir des relations I et III respectivement.

1. Montrons que si $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$, alors $\pi(q_1) \neq 0$ ou $\pi(r_1) \neq 0$.

   Nous devons distinguer les cas selon les valeurs de $\pi_{E_{a,b}(R)}(P)$ et $\pi_{E_{a,b}(R)}(P')$:

   – Cas n^o 1 : si $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$ avec $\pi_{E_{a,b}(R)}(P) = O$ ou $\pi_{E_{a,b}(R)}(P') = O$.

   Traitons le cas $\pi_{E_{a,b}(R)}(P) = O$, alors on a

   $$\pi(x) = 0,\ \pi(y) \neq 0,\ \pi(z) = 0 \text{ et } \pi(z') \neq 0.$$

   Ainsi, on obtient le résultat attendu :
   $$\pi(r_1) = (\pi(z')\pi(y))^2 \neq 0.$$
   L'autre cas s'obtient de la même manière ;

   – Cas n^o 2 : si $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$ avec $\pi_{E_{a,b}(R)}(P) \neq O$ et $\pi_{E_{a,b}(R)}(P') \neq O$ et :

   $$\pi_{E_{a,b}(R)}(P') = -\pi_{E_{a,b}(R)}(P).$$

   Alors, on a :
   $$\pi(z) \neq 0 \text{ et } \pi(z') \neq 0,$$
   ainsi que les relations :
   $$\pi\left(\frac{x}{z}\right) = \pi\left(\frac{x'}{z'}\right) \text{ et } \pi\left(\frac{y}{z}\right) = -\pi\left(\frac{y'}{z'}\right).$$

   En les utilisant, on obtient :
   $$\pi(q_1) = 8\pi(y')\pi(z')\pi(y)^2.$$
   Or $\pi(y)$ et $\pi(y')$ sont non nuls sinon la condition $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$ n'est plus vérifiée. Ainsi, on a :
   $$\pi(q_1) \neq 0\,;$$

   – Cas n^o 3 : si $\pi_{E_{a,b}(R)}(P) \neq \pi_{E_{a,b}(R)}(P')$ avec $\pi_{E_{a,b}(R)}(P) \neq O$ et $\pi_{E_{a,b}(R)}(P') \neq O$ et :

   $$\pi_{E_{a,b}(R)}(P') \neq -\pi_{E_{a,b}(R)}(P),$$

   on obtient alors $\pi(z) \neq 0$,$\pi(z') \neq 0$ et :
   $$\pi(r_1)\pi(z)\pi(z') = \left(\pi(z')\pi(x) - \pi(x)\pi(z')\right)^3 .$$
   Or $\pi\left(\frac{x}{z}\right) \neq \pi\left(\frac{x'}{z'}\right)$ sinon $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P')$ ou $\pi_{E_{a,b}(R)}(P) = -\pi_{E_{a,b}(R)}(P')$.
   Ainsi, on a :
   $$\pi(r_1) \neq 0.$$

2. Montrons que si $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P')$, alors $\pi(q_2) \neq 0$ ou $\pi(r_2) \neq 0$.

   Nous devons distinguer les cas selon les valeurs de $\pi_{E_{a,b}(R)}(P)$ et $\pi(y)$:

   – Cas n° 1 : si $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P') = O$ alors on a :

   $$\pi(x) = \pi(x') = \pi(z) = \pi(z') = 0,\ \pi(y) \neq 0 \text{ et } \pi(y') \neq 0,$$

   d'où

   $$\pi(q_2) = \pi(y)^2\pi(y')^2 \neq 0\,;$$

   – Cas n° 2 : si $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P') \neq O$ avec $\pi(y) = 0$, alors on a :

   $$\pi(y') = 0,\ \pi(z) \neq 0 \text{ et } \pi(z') \neq 0.$$

   En utilisant la relation $\pi(x)^3 = -\pi(a)\pi(x)\pi(z)^2 - \pi(b)\pi(z)^3$, on obtient :

   $$\pi(q_2) = \pi(z)^4\left(-9\pi(a)^2\pi\left(\frac{x}{z}\right)^2 - 27\pi(a)\pi(b)\pi\left(\frac{x}{z}\right) - 27\pi(b)^2 - \pi(a)^3\right).$$

   Or le résultant suivant Z des polynômes

   $$Z^3 + \pi(a)Z + \pi(b)$$
   et
   $$9\pi(a)^2Z^2 + 27\pi(a)\pi(b)Z + 27\pi(b)^2 + \pi(a)^3$$

   est égal à $\pi(4a^3 + 27b^2)^3$. Donc il est non nul et l'élément

   $$-9\pi(a)^2\pi\left(\frac{x}{z}\right)^2 - 27\pi(a)\pi(b)\pi\left(\frac{x}{z}\right) - 27\pi(b)^2 - \pi(a)^3$$

   est inversible dans K. Ainsi, on a :

   $$\pi(q_2) \neq 0\,;$$

   – Cas n° 3 : si $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P') \neq O$ avec $\pi(y) \neq 0$, alors on a :

   $$\pi(y') \neq 0,\ \pi(z) \neq 0 \text{ et } \pi(z') \neq 0.$$

   En utilisant la relation $\pi(x)^3 = -\pi(a)\pi(x)\pi(z)^2 - \pi(b)\pi(z)^3$, on obtient :

   $$\pi(r_2) = 8\pi(y)^3\pi(z) \neq 0.$$

   Ainsi, le sous-ensemble U contient les couples (P, P') tels que $\pi_{E_{a,b}(R)}(P) = \pi_{E_{a,b}(R)}(P')$ et pour (P, P') dans U, on a $P + P' = [p_2 : q_2 : r_2]$. □

2.2 Le cas de l'anneau local $\mathbb{F}_q[\varepsilon]$

Dans cette partie, nous utilisons les résultats généraux énoncés dans la section 2.1 dans le cas de l'algèbre locale $\mathbb{F}_q[\varepsilon]$ étudiée dans la sous-section 1.1.2 (cf. corollaire 1.1.1). Rappelons que pour a et b éléments de $\mathbb{F}_q[\varepsilon]$, les ensembles $E_{a,b}(\mathbb{F}_q)$ et $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ sont identiques (cf. equation 2.1 page 44). Les éléments $\pi(a)$ et $\pi(b)$ appartenant à $\mathbb{F}_q$, nous préfèrerons noter cet ensemble $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$.

2.2.1 Les applications fondamentales

Dorénavant, q représente une puissance d'un nombre premier p différent de 2 et 3.
Considérons l'anneau $\frac{\mathbb{F}_q[X]}{X^2}$ noté $\mathbb{F}_q[\varepsilon]$. C'est un anneau local d'idéal maximal (ε) et de corps résiduel $\mathbb{F}_q$ dans lequel 6 est inversible (cf. lemme 1.1.3).

La projection canonique π de $\mathbb{F}_q[\varepsilon]$ dans $\mathbb{F}_q$ envoie 1 sur 1 et ε sur 0. Autrement dit, pour tout a_0, a_1 dans $\mathbb{F}_q$, on a $\pi(a_0 + a_1\varepsilon) = a_0$.

Dorénavant, sauf mention contraire, a et b représentent des éléments de $\mathbb{F}_q[\varepsilon]$ tels que $\pi(4a^3 + 27b^2) \neq 0$. On notera $\mathbf{N}$ le cardinal $Card\left(E_{\pi(a),\pi(b)}(\mathbb{F}_q)\right)$.

Lemme 2.2.1 *L'application*

$$\begin{array}{rccc} \Theta : & \mathbb{F}_q & \to & E_{a,b}(\mathbb{F}_q[\varepsilon]) \\ & k & \mapsto & [k\varepsilon : 1 : 0] \end{array}$$

est un morphisme de groupes injectif.
Les éléments de $\Theta(\mathbb{F}_q)$ sont appelés points à l'infini de $E_{a,b}(\mathbb{F}_q[\varepsilon])$.

Preuve. On vérifie aisément que les points $[k\varepsilon : 1 : 0]$ avec k dans $\mathbb{F}_q$ satisfont l'équation $y^2z = x^3 + axz^2 + bz^3$ quels que soient a et b. La somme de deux éléments $[k\varepsilon : 1 : 0]$ et $[k'\varepsilon : 1 : 0]$ pour k et k' dans $\mathbb{F}_q$ est donnée par les relations II de la proposition 2.1.2 (l'utilisation des relations I ne conduit pas à un point de $\mathbb{P}^2(R)$).
Ainsi, on a bien $\Theta(k) + \Theta(k') = \Theta(k + k')$. L'injection est immédiate. □
Notation. On note $\lceil . \rfloor$ la fonction réciproque de Θ définie sur $\Theta(\mathbb{F}_q)$. Ainsi, pour tout point à l'infini P, $\lceil P \rfloor$ est l'unique élément de $\mathbb{F}_q$ tel que $P = \Theta(\lceil P \rfloor)$.

Lemme 2.2.2 *Le morphisme de groupes*

$$\begin{array}{rccc} \pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])} : & E_{a,b}(\mathbb{F}_q[\varepsilon]) & \to & E_{\pi(a),\pi(b)}(\mathbb{F}_q) \\ & [X : Y : Z] & \mapsto & [\pi(X) : \pi(Y) : \pi(Z)] \end{array}$$

est surjectif.

Preuve. Cette application est un morphisme de groupes d'après le lemme 2.1.1.
Soit $[X_0 : Y_0 : Z_0]$ un élément de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$. Cherchons X_1, Y_1 et Z_1 dans $\mathbb{F}_q$ tels que le point projectif $[X_0 + X_1\varepsilon : Y_0 + Y_1\varepsilon : Z_0 + Z_1\varepsilon]$ appartienne à $E_{a,b}(\mathbb{F}_q[\varepsilon])$.
Décomposons alors les coefficients a et b sous la forme $a = a_0 + a_1\varepsilon$ avec a_0 et a_1 dans $\mathbb{F}_q$, autrement dit $a_0 = \pi(a)$ et $a_1\varepsilon = a - \pi(a)$ (et b de manière identique).
Les valeurs $X_0, X_1, Y_0, Y_1, Z_0, Z_1, a_0, a_1, b_0, b_1$ vérifient alors :

$$\left\{ \begin{array}{l} Y_0^2Z_0 = X_0^3 + a_0X_0Z_0^2 + b_0Z_0^3 \\ (3X_0^2 + a_0Z_0^2)X_1 - 2Y_0Z_0Y_1 + (2a_0X_0Z_0 + 3b_0Z_0^2 - Y_0^2)Z_1 + a_1X_0Z_0^2 + b_1Z_0^3 = 0 \end{array} \right. \quad (2.2)$$

La première équation signifie que le point $[X_0 : Y_0 : Z_0]$ appartient à $E_{a_0,b_0}(\mathbb{F}_q)$, ce que nous savons déjà. De plus, les coefficients $3X_0^2 + a_0Z_0^2$, $2Y_0Z_0$ et $2a_0X_0Z_0 + 3b_0Z_0^2 - Y_0^2$ correspondant aux dérivées partielles de l'équation $y^2z - x^3 - axz^2 - bz^3$ calculées en $[X_0 : Y_0 : Z_0]$ ne peuvent être tous les trois nuls.
Alors, la seconde équation admet bien des solutions et l'application $\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}$ est surjective.
□
Remarque. Ce résultat est valable pour tout anneau local R, (cf. [Len86]).

Définition 2.2.1 *Avec les notations du lemme 2.2.2, soit P un point de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$, les éléments de $\pi^{-1}_{E_{a,b}(\mathbb{F}_q[\varepsilon])}(P)$ sont appelés relevés de P dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$.*

Notations. Pour P dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$, on note $\overline{P} = \pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}(P)$.

Lemme 2.2.3 *Avec les notations des lemmes 2.2.1 et 2.2.2, on a : $Ker(\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}) = \Theta(\mathbb{F}_q)$. Autrement dit, $\Theta(\mathbb{F}_q)$ est l'ensemble des relevés de O dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$*

Preuve. Il est évident que $\Theta(\mathbb{F}_q) \subset \mathrm{Ker}(\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])})$.
Soit P dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ tel que : $\overline{P} = O$. Il existe donc (X_1, Y_1, Z_1) dans $\mathbb{F}_q^3$ tel que

$$P = [X_1\varepsilon : 1 + Y_1\varepsilon : Z_1\varepsilon].$$

De plus, ce triplet doit vérifier $-Z_1 = 0$ (d'après les relations 2.2). D'où

$$P = [X_1\varepsilon : 1 + Y_1\varepsilon : 0] = [X_1\varepsilon : 1 : 0] = \Theta(X_1).$$

Le résultat suit. □

2.2.2 Quand p ne divise pas le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$

Notations. Pour m entier relatif, on note $[m]$ le morphisme :

$$\begin{array}{rccc} [m] : E_{a,b}(\mathbb{F}_q[\varepsilon]) & \longrightarrow & E_{a,b}(\mathbb{F}_q[\varepsilon]) \\ P & \mapsto & mP \end{array}$$

Pour un entier relatif m, on note $\overline{m}$ la classe de m dans $\mathbb{F}_p$ où p est la caractéristique du corps $\mathbb{F}_q$.
Pour un élément k de $\mathbb{F}_p$, on note $[\![k]\!]$ l'unique entier entre 0 et $p-1$ tel que $\overline{[\![k]\!]} = k$.

Lemme 2.2.4 *Si p ne divise pas $\mathbf{N}$, alors le morphisme $[p]$ se factorise de façon unique à travers $\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}$. On note ν le morphisme obtenu :*

$$\begin{array}{ccc} E_{a,b}(\mathbb{F}_q[\varepsilon]) & \overset{[p]}{} & E_{a,b}(\mathbb{F}_q[\varepsilon]) \\ {\scriptstyle \pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}} & {\scriptstyle \nu} & \\ E_{\pi(a),\pi(b)}(\mathbb{F}_q) & & \end{array}$$

Preuve. Soit $P \in \Theta(\mathbb{F}_q)$, on a :

$$pP = p\Theta(\lceil P \rfloor) = \Theta(p\lceil P \rfloor) = \Theta(0).$$

D'après le lemme 2.2.3, $\mathrm{Ker}(\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}) = \Theta(\mathbb{F}_q)$, donc on a $\mathrm{Ker}(\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}) \subset \mathrm{Ker}([p])$. Par passage au quotient, il existe un unique morphisme de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ rendant le diagramme précédent commutatif. □

Rappelons maintenant que $\mathbf{N}$ représente le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$; on note alors $\mathbf{N}'$ l'unique entier naturel inférieur à p tel que $\overline{\mathbf{N}'} = \overline{\mathbf{N}}^{-1}$; soit $\mathbf{N}' = [\![\overline{\mathbf{N}}^{-1}]\!]$.
Le lemme 2.2.4 permet de définir, de façon immédiate, l'application λ du corollaire 2.2.1.

Corollaire 2.2.1 *Si p ne divise pas $\mathbf{N}$, alors le morphisme $[1-\mathbf{NN}']$ se factorise de façon unique à travers $\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}$. On note λ le morphisme obtenu :*

$$\begin{array}{ccc} E_{a,b}(\mathbb{F}_q[\varepsilon]) & \overset{[1-\mathbf{NN}']}{} & E_{a,b}(\mathbb{F}_q[\varepsilon]) \\ {\scriptstyle \pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}} & \overset{\lambda}{} & \\ E_{\pi(a),\pi(b)}(\mathbb{F}_q) & & \end{array}$$

Preuve. On a $\mathbf{N}' = [\![\overline{\mathbf{N}}^{-1}]\!]$. Donc il existe c dans $\mathbb{Z}$ tel que $1-\mathbf{NN}' = cp$. Ainsi les morphismes $[1-\mathbf{NN}']$ et $[c]\circ[p]$ sont égaux. Et, il existe un unique morphisme de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ rendant le diagramme précédent commutatif. □

Lemme 2.2.5 *Avec les notations du corollaire 2.2.1, on a :*

$$\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])} \circ \lambda = Id_{E_{\pi(a),\pi(b)}(\mathbb{F}_q)}.$$

Preuve. Soit P un point de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$, d'après le corollaire 2.2.1, il existe un élément P' de $E_{a,b}(\mathbb{F}_q[\varepsilon])$, tel que :

$$\overline{P'} = P \text{ et } \lambda(P) = (1-\mathbf{NN}')P'.$$

D'où la relation :

$$\lambda(P) = P' - \mathbf{N}'\mathbf{N}P'.$$

Or, d'après les lemmes 2.1.1 ou 2.2.2, $\mathbf{N}P'$ est un point à l'infini, donc on a :

$$\mathbf{N}P' = \Theta(\lceil \mathbf{N}P' \rfloor) \text{ et } \lambda(P) = P' - \mathbf{N}'\Theta(\lceil \mathbf{N}P' \rfloor).$$

On obtient ainsi :

$$\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])} \circ \lambda(P) = \overline{P'} - \mathbf{N}'O = P.$$

□

Corollaire 2.2.2 *Si p ne divise pas $\mathbf{N}$, alors on a l'isomorphisme de groupes :*

$$E_{a,b}(\mathbb{F}_q[\varepsilon]) \cong E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q.$$

Preuve. Des lemmes 2.2.1, 2.2.2, 2.2.3 et 2.2.5, on obtient la suite exacte scindée :

$$\begin{array}{ccccccccc} 0 & & \mathbb{F}_q & \overset{\Theta}{} & E_{a,b}(\mathbb{F}_q[\varepsilon]) & \overset{\pi}{} & E_{\pi(a),\pi(b)}(\mathbb{F}_q) & & 0 \\ & & & & & \overset{\lambda}{} & & & \end{array}$$

D'où l'isomorphisme annoncé. □

Le lemme 2.2.6 donne explicitement un isomorphisme entre les groupes $E_{a,b}(\mathbb{F}_q[\varepsilon])$ et $E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$.

Lemme 2.2.6 *L'isomorphisme*

$$\begin{array}{rcl} E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q & \to & E_{a,b}(\mathbb{F}_q[\varepsilon]) \\ (P,k) & \mapsto & \lambda(P) + \Theta(k) \end{array}$$

admet pour application réciproque :

$$\begin{array}{rcl} \Lambda : E_{a,b}(\mathbb{F}_q[\varepsilon]) & \longrightarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q \\ P & \mapsto & (\overline{P}, \lceil \mathbf{N}\mathbf{N}'P \rfloor) \end{array}$$

Preuve. Notons temporairement f la première application. D'après le corollaire 2.2.2, f est un isomorphisme de groupes.
Soit P dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$, on a les relations suivantes d'après le lemme 2.2.1 et le corollaire 2.2.1 :

$$f \circ \Lambda(P) = f(\overline{P}, \lceil \mathbf{N}\mathbf{N}'P \rfloor) = \lambda(\overline{P}) + \Theta(\lceil \mathbf{N}\mathbf{N}'P \rfloor) = (1 - \mathbf{N}\mathbf{N}')P + \mathbf{N}\mathbf{N}'P = P$$

Soit $(P,k) \in E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$, on a les relations suivantes :

$$\Lambda \circ f(P,k) = \Lambda(\lambda(P) + \Theta(k)) = \left(\overline{\lambda(P) + \Theta(k)}, \lceil \mathbf{N}\mathbf{N}'(\lambda(P) + \Theta(k)) \rfloor\right).$$

D'après le lemme 2.2.5, on a $\overline{\lambda(P)} = P$. De plus, soit P' un relèvement de P et c l'entier tel que : $\mathbf{N}\mathbf{N}' = 1 + cp$, alors on a :

$$\mathbf{N}\mathbf{N}'\lambda(P) = \mathbf{N}\mathbf{N}'(1 - \mathbf{N}\mathbf{N}')P' = \mathbf{N}'cp\mathbf{N}P'.$$

Or l'élément est $\mathbf{N}P'$ est un point à l'infini et, d'après le lemme 2.2.1 on a les relations suivantes :

$$\mathbf{N}\mathbf{N}'\lambda(P) = \mathbf{N}'cp\Theta(\lceil \mathbf{N}P' \rfloor) = \Theta(\mathbf{N}'cp\lceil \mathbf{N}P' \rfloor) = \Theta(0).$$

On obtient ainsi :

$$\Lambda \circ f(P,k) = (P + O, \lceil \Theta(0) + (1 + cp)\Theta(k) \rfloor) = (P, \lceil \Theta((1 + cp)k) \rfloor) = (P,k).$$

Et f est bien l'application réciproque de Λ. □

Ainsi, pour a et b paramétrant une courbe elliptique $E_{a,b}$ sur $\mathbb{F}_q$, on note Λ^{-1} le morphisme de $E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ défini par :

$$\Lambda^{-1}(P,k) = \lambda(P) + \Theta(k)$$

et λ est défini page 49.

Finissons cette section, en montrant le lemme 2.2.7 dont on se servira dans la section 4.3.

Lemme 2.2.7 *Si p ne divise pas $\mathbf{N}$, pour tout P dans $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$, on a :*

$$\Lambda(pP) = \left(p\overline{P}, 0\right) \text{ et } \Lambda \circ \nu(P) = (pP, 0)$$

Preuve. Soit P dans $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ et P' dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ tel que $\overline{P'} = P$, on a :

$$\Lambda \circ \nu(P) = \Lambda(pP') = (p\overline{P'}, \lceil \mathbf{N}\mathbf{N}'pP' \rfloor) = (pP, p\lceil \mathbf{N}\mathbf{N}'P' \rfloor) = (pP, 0). \qquad \square$$

2.2.3 Quand p ne divise pas le cardinal de $E_{a,b}(\mathbb{F}_q)$ avec a et b dans $\mathbb{F}_q$

Dans cette section, nous donnons une autre expression de l'isomorphisme Λ dans le cas où les coefficients a et b de l'équation de Weierstrass définissant $E_{a,b}(\mathbb{F}_q[\varepsilon])$ sont dans $\mathbb{F}_q$. Ce résultat (lemme 2.2.9) permet de calculer Λ sans utiliser les lois de groupe de $E_{a,b}(\mathbb{F}_q[\varepsilon])$ et $E_{a,b}(\mathbb{F}_q)$. Pour cela, commençons par montrer le lemme 2.2.8, qui donne une condition nécessaire et suffisante pour qu'un élément de $E_{a,b}(\mathbb{F}_q[\varepsilon]) \setminus \Theta(\mathbb{F}_q)$ soit dans $E_{a,b}(\mathbb{F}_q)$.

Lemme 2.2.8 *Soit a et b dans $\mathbb{F}_q$, pour tout P dans $E_{a,b}(\mathbb{F}_q[\varepsilon]) \setminus \Theta(\mathbb{F}_q)$ avec*

$$P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$$

et x_0, x_1, y_0, y_1 dans $\mathbb{F}_q$, on a :

$$\mathbf{N}P = \Theta(0) \text{ si et seulement si } x_1 = y_1 = 0.$$

Preuve Dans un premier temps, supposons $x_1 = y_1 = 0$ alors $P = [x_0 : y_0 : 1]$ appartient à $E_{a,b}(\mathbb{F}_q)$. Donc pour tout entier n, nP est dans $E_{a,b}(\mathbb{F}_q)$. Ainsi $\mathbf{N}P = \Theta(k)$ avec $\Theta(k)$ dans $E_{a,b}(\mathbb{F}_q)$. Donc on a $k = 0$ et $\mathbf{N}P = \Theta(0)$.
Pour la réciproque, supposons que l'on ait : $\mathbf{N}[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] = \Theta(0)$. L'élément $[x_0 : y_0 : 1]$ appartient à $E_{a,b}(\mathbb{F}_q[\varepsilon])$, donc on a $\mathbf{N}[x_0 : y_0 : 1] = \Theta(0)$. Alors on a :

$$\mathbf{N}[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] - \mathbf{N}[x_0 : y_0 : 1] = \mathbf{N}\left([x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] - [x_0 : y_0 : 1]\right) = \Theta(0).$$

Or il existe k dans $\mathbb{F}_q$ tel que $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] - [x_0 : y_0 : 1] = \Theta(k)$ et $\mathbf{N}k = 0$. Or $\mathbf{N}$ est premier avec p, donc on a $k = 0$, d'où $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] = [x_0 : y_0 : 1]$ et $x_1 = y_1 = 0$. □
On en déduit le lemme 2.2.9.

Lemme 2.2.9 *Soit P un élément de $E_{a,b}(\mathbb{F}_q[\varepsilon])$, on a :*

- *si $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ avec x_0, x_1, y_0, y_1 dans $\mathbb{F}_q$, $\Lambda(P) = \left([x_0 : y_0 : 1], -\dfrac{x_1}{2y_0}\right)$;*
- *si $P = \Theta(k)$ avec k dans $\mathbb{F}_q$, $\Lambda(P) = (O, k)$.*

Preuve. Traitons séparément chacun des cas :
- si $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$: il est clair que $\overline{P} = [x_0 : y_0 : 1]$.
  Montrons que $\lceil \mathbf{N}\mathbf{N}'P \rfloor = -\dfrac{x_1}{2y_0}$. On a :

$$\Theta\left(\lceil \mathbf{N}\mathbf{N}'P \rfloor\right) = \mathbf{N}\mathbf{N}'P = P + \left(\mathbf{N}\mathbf{N}' - 1\right)P.$$

  Or $\mathbf{N}\mathbf{N}' - 1$ est un multiple de p, donc $\mathbf{N}\left(\mathbf{N}\mathbf{N}' - 1\right)P = \Theta(0)$. D'après le lemme 2.2.8, $\left(\mathbf{N}\mathbf{N}' - 1\right)P$ est alors dans $E_{a,b}(\mathbb{F}_p)$. De plus, on a $\overline{\left(\mathbf{N}\mathbf{N}' - 1\right)P} = -\overline{P}$ et on trouve : $\left(\mathbf{N}\mathbf{N}' - 1\right)P = [x_0 : -y_0 : 1]$.
  On obtient alors l'égalité : $\Theta\left(\lceil \mathbf{N}\mathbf{N}'P \rfloor\right) = P + [x_0 : -y_0 : 1]$ et d'après la loi de groupe établie dans la section 2.3, on établit l'égalité :

$$\lceil \mathbf{N}\mathbf{N}'P \rfloor = \frac{-x_1}{2y_0};$$

- si $P = \Theta(k)$, il est clair que $\overline{\Theta(k)} = O$. On a vu précédemment que $\mathbf{N}\mathbf{N}' - 1$ est un multiple de p, et on obtient $\mathbf{N}\mathbf{N}'\Theta(k) = \Theta(k) + \left(\mathbf{N}\mathbf{N}' - 1\right)\Theta(k) = \Theta(k)$.

Les égalités obtenues dans chaque cas achèvent la preuve. □

2.2.4 Quand p divise N

Quand la caractéristique du corps $\mathbb{F}_q$ divise le cardinal du groupe $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$, les groupes $E_{a,b}(\mathbb{F}_q[\varepsilon])$ et $E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$ ne sont pas nécessairement isomorphes.

Exemple 2.2.1 *$E_{3+\varepsilon,2+\varepsilon}(\mathbb{F}_5[\varepsilon])$ est isomorphe à $E_{3,2}(\mathbb{F}_5) \times \mathbb{F}_5$.*
Ces deux groupes sont d'ordre 25 et tous leurs éléments sont d'ordre 1 ou 5 (cf. Annexe B). Donc ils sont tous deux isomorphes à $\frac{\mathbb{Z}}{5\mathbb{Z}} \times \frac{\mathbb{Z}}{5\mathbb{Z}}$.

Exemple 2.2.2 *$E_{3+\varepsilon,2+2\varepsilon}(\mathbb{F}_5[\varepsilon])$ n'est pas isomorphe à $E_{3,2}(\mathbb{F}_5) \times \mathbb{F}_5$.*
Ces deux groupes sont d'ordre 25. Le premier est cyclique (de générateur $[1+4\varepsilon : 1+\varepsilon : 1]$ cf. Annexe B), le second ne l'est pas puisque tous ses éléments sont d'ordre 1 ou 5.

Dans ce cas, le lemme 2.2.10 permettra de résoudre le problème du logarithme discret sur ces courbes elliptiques en section 4.1.

Lemme 2.2.10 *Si p divise $\mathbf{N}$, alors le morphisme $\lceil . \rfloor \circ [\mathbf{N}]$ se factorise de façon unique à travers $\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}$. On note μ le morphisme obtenu :*

$$\begin{array}{ccc} E_{a,b}(\mathbb{F}_q[\varepsilon]) & \overset{\lceil . \rfloor \circ [\mathbf{N}]}{} & \mathbb{F}_q \\ {\scriptstyle \pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}} & {\scriptstyle \mu} & \\ E_{\pi(a),\pi(b)}(\mathbb{F}_q) & & \end{array}$$

Preuve. Le groupe $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ est d'ordre $\mathbf{N}$, donc $[\mathbf{N}](E_{a,b}(\mathbb{F}_q[\varepsilon])) \subset \Theta(\mathbb{F}_q)$ et l'application suivante est bien définie :

$$\begin{array}{rccc} \lceil . \rfloor \circ [\mathbf{N}] : E_{a,b}(\mathbb{F}_q[\varepsilon]) & \longrightarrow & \mathbb{F}_q \\ P & \mapsto & \lceil \mathbf{N}P \rfloor \end{array}$$

Le nombre premier p divise $\mathbf{N}$. Soit c l'entier naturel tel que $\mathbf{N} = cp$, soit P un élément de $\Theta(\mathbb{F}_q)$, on obtient les relations suivantes :

$$\lceil \mathbf{N}P \rfloor = \lceil cpP \rfloor = cp\lceil P \rfloor = 0.$$

Donc, d'après le lemme 2.2.3, on a : $\mathrm{Ker}(\pi_{E_{a,b}(\mathbb{F}_q[\varepsilon])}) \subset \mathrm{Ker}(\lceil . \rfloor \circ [\mathbf{N}])$. Le résultat suit. □

2.3 Loi d'addition sur $E_{a,b}(\mathbb{F}_q[\varepsilon])$

D'après les lemmes 2.2.1 et 2.2.2, les éléments de $E_{a,b}(\mathbb{F}_q[\varepsilon])$ admettent tous un unique représentant de la forme $[k\varepsilon : 1 : 0]$ ou $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ avec k, x_0, x_1, y_0, y_1 dans $\mathbb{F}_q$. La somme de deux éléments est alors donnée par les relations de la proposition 2.1.2.

L'objectif de cette section est d'obtenir des relations qui, étant donnés deux éléments de $E_{a,b}(\mathbb{F}_q[\varepsilon])$ représentés sous une de ces deux formes, renvoient la somme de ces éléments sous la forme adéquate.

Le lecteur pourra retrouver une synthèse de ces résultats à la fin de cette section par la table 2.1.

On doit alors traiter différents cas selon que les éléments à additionner sont de la forme $[k\varepsilon : 1 : 0]$ ou $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$. Pour P et P' dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$, on obtient ainsi les cas :
- $\overline{P} = O$ et $\overline{P'} = O$. Dans ce cas, on a $\overline{P + P'} = O$ d'après le lemme 2.1.1 et on obtient le résultat en utilisant les relations II de la proposition 2.1.2 ;
- ($\overline{P} = O$ et $\overline{P'} \neq O$) ou ($\overline{P} \neq O$ et $\overline{P'} = O$). Dans ce cas, on a $\overline{P + P'} \neq O$ d'après le lemme 2.1.1 et on obtient le résultat en utilisant les relations I de la proposition 2.1.2 ;
- $\overline{P} \neq O$ et $\overline{P'} \neq O$. Dans ce cas, la valeur de $\overline{P + P'}$ dépend de la condition $\overline{P'} = -\overline{P}$. Ainsi, on obtient deux autres cas :
  - $\overline{P'} = -\overline{P}$. Dans ce cas, on a $\overline{P + P'} = O$ d'après le lemme 2.1.1. Le résultat s'obtient en utilisant les relations I ou II de la proposition 2.1.2 selon que $\overline{P'}$ égale $\overline{P}$ ou pas.
  - $\overline{P'} \neq -\overline{P}$. Dans ce cas, on a $\overline{P + P'} \neq O$ d'après le lemme 2.1.1. Là encore nous devons considérer si $\overline{P'}$ égale $\overline{P}$ ou pas pour utiliser les relations adéquates.

On obtient ainsi six différents cas selon que $\overline{P} = O$, $\overline{P'} = O$, $\overline{P'} = -\overline{P}$, $\overline{P'} = \overline{P}$:

1. $\overline{P} = O$ et $\overline{P'} = O$, où $\overline{P + P'} = O$: on utilise les relations II de la proposition 2.1.2 ;
2. $\overline{P} \neq O$ et $\overline{P'} = O$, où $\overline{P + P'} \neq O$: on utilise les relations I de la proposition 2.1.2 ;
3. $\overline{P'} = -\overline{P}$ et $\overline{P'} = \overline{P}$, où $\overline{P + P'} = O$: on utilise les relations II de la proposition 2.1.2 ;
4. $\overline{P'} = -\overline{P}$ et $\overline{P'} \neq \overline{P}$, où $\overline{P + P'} = O$: on utilise les relations I de la proposition 2.1.2 ;
5. $\overline{P'} \neq -\overline{P}$ et $\overline{P'} = \overline{P}$, où $\overline{P + P'} \neq O$: on utilise les relations II de la proposition 2.1.2 ;
6. $\overline{P'} \neq -\overline{P}$ et $\overline{P'} \neq \overline{P}$, où $\overline{P + P'} \neq O$: on utilise les relations I de la proposition 2.1.2.

Les lemmes 2.3.1 à 2.3.6 traitent de ces différents cas. La figure 2.1 en page 56 représente chaque domaine de validité et les calculs nécessaires à chaque preuve sont en Annexe A.

Lemme 2.3.1 (Cas 1 : $\overline{P} = O$ et $\overline{P'} = O$) *Pour tout k et k' dans $\mathbb{F}_q$, on a :*

$$\Theta(k) + \Theta(k') = \Theta(k + k').$$

Preuve. Ce résultat a déjà été établi dans la preuve du lemme 2.2.1. Les calculs consistent à remplacer x par k, x' par k', les variables y et y' par 1 et z et z' par 0 dans les relations II de la proposition 2.1.2. □

Remarque. Dans ce cas, les relations I ne sont pas valables car elles aboutissent à un triplet d'éléments non inversibles.

Lemme 2.3.2 (Cas 2 : $\overline{P} \neq O$ et $\overline{P'} = O$) *Pour $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ avec x_0, x_1, y_0, y_1 dans $\mathbb{F}_q$ et k dans $\mathbb{F}_q$, on a :*

$$P + \Theta(k) = [x_0 + (x_1 - 2y_0k)\varepsilon : y_0 + (y_1 - k(3x_0^2 + a_0))\varepsilon : 1].$$

Preuve. Ces calculs consistent à poser $x = x_0 + x_1\varepsilon$, $y = y_0 + y_1\varepsilon$, $z = 1$, $x' = k$, $y' = 1$ et $z' = 0$ dans les relations I de la proposition 2.1.2. Il suffit ensuite de multiplier le résultat par -1 pour obtenir un élément du type $[X : Y : 1]$. □

Remarque. Dans ce cas, les relations II aboutissent au même résultat mais restent valables seulement si $y_0 \neq 0$.

Lemme 2.3.3 (Cas 3 : $\overline{P'} = -\overline{P}$ et $\overline{P'} = \overline{P}$, soit $(x'_0, y'_0) = (x_0, -y_0) = (x_0, 0)$)
Pour $P = [x_0 + x_1\varepsilon : y_1\varepsilon : 1]$ et $P' = [x_0 + x'_1\varepsilon : y'_1\varepsilon : 1]$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$, avec $x_0, x_1, x'_1, y_1, y'_1$ dans $\mathbb{F}_q$ on a :

$$P + P' = \Theta\left(-\frac{y'_1 + y_1}{3x_0^2 + a_0}\right) \text{ où } a_0 = \pi(a).$$

Preuve. Ces calculs consistent, dans un premier temps, à poser $x = x_0 + x_1\varepsilon$, $y = y_1\varepsilon$, $z = 1$, $x' = x_0 + x'_1\varepsilon$, $y = y'_1\varepsilon$, $z' = 1$, $a = a_0 + a_1\varepsilon$ et $b = b_0 + b_1\varepsilon$ dans les relations II de la proposition 2.1.2. On utilise ensuite la relation $x_0^3 + a_0x_0 + b_0 = 0$.
On obtient alors un point du type $[X_1\varepsilon : Y_0 + Y_1\varepsilon : 0]$ avec $Y_0 = 9a_0^2x_0^2 + 27a_0b_0x_0 + 27b_0^2 + a_0^3$. Or le résultant suivant Z des polynômes $Z^3 + a_0Z + b_0$ et $9a_0^2Z^2 + 27a_0b_0Z + 27b_0^2 + a_0^3$ est égal à $(4a_0^3 + 27b_0^2)^3 = \pi(4a^3 + 27b^2)^3$. Donc il est non nul et l'élément $9a_0^2x_0^2 + 27a_0b_0x_0 + 27b_0^2 + a_0^3$ est inversible dans $\mathbb{F}_q$. Ainsi, on a :

$$[X_1\varepsilon : Y_0 + Y_1\varepsilon : 0] = \left[\frac{X_1}{9a_0^2x_0^2 + 27a_0b_0x_0 + 27b_0^2 + a_0^3}\varepsilon : 1 : 0\right].$$

On peut alors factoriser le numérateur et le dénominateur de la première variable par $-3a_0x_0^2 - 9b_0x_0 + a_0^2$. Or cet élément est inversible dans $\mathbb{F}_q$ car le résultant suivant Z des polynômes $Z^3 + a_0Z + b_0$ et $-3a_0Z^2 - 9b_0Z + a_0^2$ est égal à $\pi(4a^3 + 27b^2)^2 \neq 0$. En simplifiant numérateur et dénominateur, on obtient le résultat énoncé. □

Lemme 2.3.4 (Cas 4 : $\overline{P'} = -\overline{P}$ et $\overline{P'} \neq \overline{P}$, soit $(x'_0, y'_0) = (x_0, -y_0)$ et $y_0 \neq 0$)
Pour $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ et $P' = [x_0 + x'1\varepsilon : -y_0 + y'_1\varepsilon : 1]$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ avec $x_0, x_1, x'_1, y_0, y_1, y'_1$ dans $\mathbb{F}_q$ tels que $y_0 \neq 0$, on a :

$$P + P' = \Theta\left(\frac{x'_1 - x_1}{2y_0}\right).$$

Preuve. Ces calculs consistent, dans un premier temps, à poser $x = x_0 + x_1\varepsilon$, $y = y_0 + y_1\varepsilon$, $z = 1$, $x' = x_0 + x'_1\varepsilon$, $y = -y_0 + y'_1\varepsilon$, $z' = 1$, $a = a_0 + a_1\varepsilon$ et $b = b_0 + b_1\varepsilon$ dans les relations I de la proposition 2.1.2. Puis on utilise les relations issues du système 2.2 :

$$\begin{cases} y_0^2 = x_0^3 + a_0x_0 + b_0 \\ 2y_0y_1 = (3x_0^2 + a_0)x_1 + a_1x_0 + b_1 \\ -2y_0y'_1 = (3x_0^2 + a_0)x'_1 + a_1x_0 + b_1 \end{cases} \tag{2.3}$$

On obtient alors un point du type $[X_1\varepsilon : Y_0 + Y_1\varepsilon : 0]$ avec $Y_0 = 8y_0^3$. De plus y_0 est non nul, ainsi on a : $[X_1\varepsilon : Y_0 + Y_1\varepsilon : 0] = \left[\frac{X_1}{8y_0^3}\varepsilon : 1 : 0\right]$. Cela nous donne le résultat recherché. □

Lemme 2.3.5 (Cas 5 : $\overline{P'} \neq -\overline{P}$ et $\overline{P'} = \overline{P}$, soit $(x'_0, y'_0) = (x_0, y_0)$ et $y_0 \neq 0$)
Pour $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ et $P' = [x_0 + x'_1\varepsilon : y_0 + y'_1\varepsilon : 1]$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ avec $x_0, x_1, x'_1, y_0, y_1, y'_1$ dans $\mathbb{F}_q$ tels que $y_0 \neq 0$, on a :

$$P + P' = [X : Y : 1]$$

$$\text{avec } X = \alpha^2 - 2x_0 - (x_1 + x'_1)\varepsilon,\ Y = \alpha(x_0 + x_1\varepsilon - X) - y_0 - y_1\varepsilon$$

$$\text{où } \alpha = \frac{3x_0^2 + a_0}{2y_0} + \frac{1}{2y_0}\left(a_1 + 3x_0(x_1 + x'_1) - \frac{3x_0^2 + a_0}{2y_0}(y_1 + y'_1)\right)\varepsilon,$$

$$\text{avec } a_0 = \pi(a),\ a - \pi(a) = a_1\varepsilon.$$

Preuve. Dans un premier temps, on utilise les relations II de la proposition 2.1.2 en posant $x = x_0 + x_1\varepsilon$, $x' = x_0 + x'_1\varepsilon$, $y = y_0 + y_1\varepsilon$, $y' = y_0 + y'_1\varepsilon$, $z' = z = 1$, $a = a_0 + a_1\varepsilon$ et $b = b_0 + b_1\varepsilon$. On multiplie le triplet par l'élément inversible $2y_0 \neq 0$ et on utilise les relations issues du système 2.2 :

$$\begin{cases} x_0^3 = y_0^2 - a_0x_0 - b_0 \\ 2y_0y_1 = (3x_0^2 + a_0)x_1 + a_1x_0 + b_1 \\ 2y_0y'_1 = (3x_0^2 + a_0)x'_1 + a_1x_0 + b_1 \end{cases} \tag{2.4}$$

On obtient ainsi un triplet Res_1 dont le dernier élément

$$c = 16y_0^4 + 12y_0^2(2b_1 + (3x_0^2 + a_0)(x_1 + x'_1) + 2a_1x_0)\varepsilon$$

est inversible dans $\mathbb{F}_q[\varepsilon]$.
On considère ensuite le triplet $c(X, Y, 1)$. On le développe et on utilise les relations

$$\begin{cases} 2y_0y_1 = (3x_0^2 + a_0)x_1 + a_1x_0 + b_1 \\ 2y_0y'_1 = (3x_0^2 + a_0)x'_1 + a_1x_0 + b_1 \end{cases} \tag{2.5}$$

On obtient ainsi un triplet Res_2.
Enfin, on trouve que la différence entre Res_1 et Res_2 est nulle en utilisant la relation $y_0^2 = x_0^3 + a_0x_0 + b_0$. Ainsi, les résultats de l'énoncé fournissent le même point projectif que les relations de la proposition 2.1.1. □

Lemme 2.3.6 (Cas 6 : $\overline{P'} \neq -\overline{P}$ et $\overline{P'} \neq \overline{P}$, soit $x'_0 \neq x_0$)
Pour $P = [x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ et $P' = [x'_0 + x'_1\varepsilon : y'_0 + y'_1\varepsilon : 1]$ dans $E_{a,b}(\mathbb{F}_q[\varepsilon])$ avec $x_0, x_1, x'_0, x'_1, y_0, y_1, y'_0, y'_1$ dans $\mathbb{F}_q$ tels que $x_0 \neq x'_0$ on a :

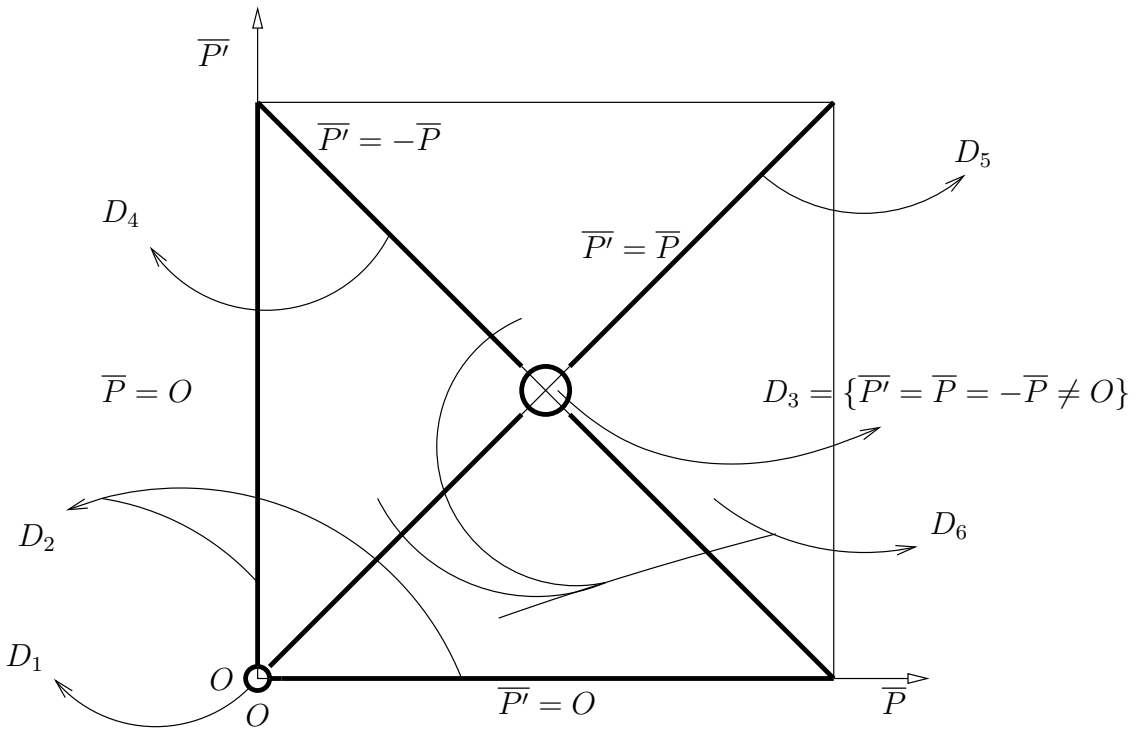
$$P + P' = [X : Y : 1]$$

$$\text{avec } X = \alpha^2 - x_0 - x'_0 - (x_1 + x'_1)\varepsilon,\ Y = \alpha(x_0 + x_1\varepsilon - X) - y_0 - y_1\varepsilon$$

$$\text{où } \alpha = \frac{y'_0 - y_0}{x'_0 - x_0} + \frac{(y'_1 - y_1)(x'_0 - x_0) - (y'_0 - y_0)(x'_1 - x_1)}{(x'_0 - x_0)^2}\varepsilon.$$

Preuve. Les relations de l'énoncé s'écrivent plus succinctement $X = \alpha^2 - x - x'$, $Y = \alpha(x - X) - y$ et $\alpha = \dfrac{y' - y}{x' - x}$ avec $x = x_0 + x_1\epsilon$, $y = y_0 + y_1\varepsilon$ et $x' = x_0 + x'_1\varepsilon$.
Elles sont obtenues par construction de la corde passant par les points P et P'. Elles correspondent également aux relations I de la proposition 2.1.2. On les retrouve en multipliant le triplet $(X, Y, 1)$ par l'élément inversible $(x - x')^3$ et en utilisant les relations $y^2 = x^3 + ax + b$ et $y'^2 = x'^3 + ax' + b$. □
Etant donnés deux éléments de $E_{a,b}(\mathbb{F}_q[\varepsilon])$ écrits sous une des formes $[k\varepsilon : 1 : 0]$ ou $[X : Y : 1]$, la somme de ces deux éléments est donnée par un des lemmes 2.3.1 à 2.3.6. Voici représentés sur la figure 2.1, les domaines d'application de chacun de ces lemmes selon les valeurs de $\overline{P}$ et $\overline{P'}$.

Ces résultats sont synthétisés dans la table 2.1, où $x_0, x_1, y_0, y_1, x'_0, x'_1, y'_0, y'_1, k$ et k' sont des éléments de $\mathbb{F}_q$ et où a_0, a_1, b_0, b_1 sont les uniques éléments de $\mathbb{F}_q$ tels que $a = a_0 + a_1\varepsilon$ et $b = b_0 + b_1\varepsilon$.



Domaine d'application du lemme 2.3.1 $D_1 = \{(P, P') \in E_{a,b}(\mathbb{F}_q[\varepsilon])^2 : (\overline{P}, \overline{P'}) = (O, O)\}$
Domaine d'application du lemme 2.3.2 $D_2 = \{(P, P') \in E_{a,b}(\mathbb{F}_q[\varepsilon])^2 : \overline{P} = O \text{ ou } \overline{P'} = O\} \setminus D_1$
Domaine d'application du lemme 2.3.3 $D_3 = \{(P, P') \in E_{a,b}(\mathbb{F}_q[\varepsilon])^2 : \overline{P'} = \overline{P} = -\overline{P}\} \setminus D_1$
Domaine d'application du lemme 2.3.4 $D_4 = \{(P, P') \in E_{a,b}(\mathbb{F}_q[\varepsilon])^2 : \overline{P'} = -\overline{P}\} \setminus (D_1 \bigcup D_3)$
Domaine d'application du lemme 2.3.5 $D_5 = \{(P, P') \in E_{a,b}(\mathbb{F}_q[\varepsilon])^2 : \overline{P'} = \overline{P}\} \setminus (D_1 \bigcup D_3)$
Domaine d'application du lemme 2.3.6 $D_6 = E_{a,b}(\mathbb{F}_q[\varepsilon])^2 \setminus (D_1 \bigcup D_2 \bigcup D_3 \bigcup D_4 \bigcup D_5)$

FIG. 2.1 – Domaines d'application des lemmes 2.3.1 à 2.3.6

	P	P'	$P+P'$
1.	$\mathcal{O}_k=[k\varepsilon:1:0]$	$\mathcal{O}_{k'}=[k'\varepsilon:1:0]$	$\mathcal{O}_{k+k'}=[(k+k')\varepsilon:1:0]$
2.	$[x_0+x_1\varepsilon:y_0+y_1\varepsilon:1]$	$\mathcal{O}_k=[k\varepsilon:1:0]$	$[X:Y:1]$ avec $X=x_0+(x_1-2y_0k)\varepsilon$ et $Y=y_0+(y_1-k(3x_0^2+a_0))\varepsilon$
	$[x_0+x_1\varepsilon:y_0+y_1\varepsilon:1]$	$[x'_0+x'_1\varepsilon:y'_0+y'_1\varepsilon:1]$	$\mathcal{O}_k=[k\varepsilon:1:0]$
3.	Si $x'_0=x_0$ et $y'_0=-y_0=0$		$k=-\dfrac{y_1+y'_1}{3x_0^2+a_0}$
4.	Si $x'_0=x_0$ et $y'_0=-y_0\neq 0$		$k=\dfrac{x'_1-x_1}{2y_0}$
	$[x_0+x_1\varepsilon:y_0+y_1\varepsilon:1]$	$[x'_0+x'_1\varepsilon:y'_0+y'_1\varepsilon:1]$	$[X:Y:1]$ avec $X=\lambda^2-(x_0+x'_0)-(x_1+x'_1)\varepsilon$ et $Y=\lambda(x_0+x_1\varepsilon-X)-y_0-y_1\varepsilon$
5.	Si $x'_0=x_0$ et $y'_0=y_0\neq 0$		$\lambda=\dfrac{3x_0^2+a_0}{2y_0}$ $+\dfrac{1}{2y_0}\left(a_1+3x_0(x_1+x'_1)-\dfrac{3x_0^2+a_0}{2y_0}(y_1+y'_1)\right)\varepsilon$
6.	Si $x'_0\neq x_0$		$\lambda=\dfrac{y'_0-y_0}{x'_0-x_0}$ $+\dfrac{(y'_1-y_1)(x'_0-x_0)-(y'_0-y_0)(x'_1-x_1)}{(x'_0-x_0)^2}\varepsilon$

TAB. 2.1 – Somme de deux éléments de $E_{a,b}(\mathbb{F}_q[\varepsilon])$.

Chapitre 3

Propriétés effectives de $E_{a,b}(\mathbb{F}_p[\varepsilon])$

Dans un premier temps, nous avons voulu identifier les différents algorithmes qui permettent d'utiliser un groupe (mathématique) dans un programme (informatique). Ce type de travail a été effectué dans la thèse de Damien Vergnaud [Ver06], avec l'introduction de représentation algorithmique d'un groupe. Notre approche est un peu différente. Ainsi, nous introduisons la notion de groupe effectif que nous illustrons par des exemples.
Dans un second temps, nous exposons explicitement un groupe effectif représentant $E_{a,b}(\mathbb{F}_p[\varepsilon])$ ainsi que d'autres algorithmes sous-jacents dont nous aurons l'utilité dans le chapitre 4.
Enfin, nous considérons le problème du logarithme discret et les problèmes calculatoires et décisionnels de Diffie Hellman sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ dont nous étudions les relations avec les problèmes correspondants sur $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$.

3.1 Notion de groupes effectifs

3.1.1 Groupe effectif - Cas particuliers

Dans cette section, nous donnons une notion de groupe effectif représentant un groupe (supposé commutatif). Cette notion permet, par exemple, de préciser la façon dont sont représentés les éléments du groupe ou de s'assurer que les algorithmes sous-jacents (notamment celui calculant la somme de deux éléments) sont efficaces.
Pour cela, commençons par rappeler la définition d'un groupe abélien.

Définition 3.1.1 *Un* groupe abélien $(G, +, -, e)$ *est un quadruplet composé d'un ensemble* G*, d'une loi de composition interne* $+$*, d'une involution* $-$ *de* G *et d'un élément* e *de* G *tels que :*

1. $\forall (a, b, c) \in G^3, (a + b) + c = a + (b + c)$;
2. $\forall a \in G, a + e = a$;
3. $\forall a \in G, a + (-a) = e$;
4. $\forall (a, b) \in G^2, a + b = b + a$.

Dans la suite, tous les groupes seront supposés abéliens.
Pour se servir d'un groupe dans une procédure informatique, nous devons pouvoir :
– déterminer si une chaîne de bits représente bien un élément du groupe ;
– déterminer si deux éléments sont égaux ;

– identifier quel élément est l'élément neutre (sans avoir à vérifier la propriété : pour tout a dans G, $a + e = a$) ;
– effectuer tous les calculs liés à la structure de groupe : addition, opposé.
La définition 3.1.2 expose notre idée d'un groupe effectif.

Définition 3.1.2 *Un* groupe effectif *est la donnée de cinq algorithmes polynomiaux*

$$\mathsf{G} = (\mathsf{App}, \mathsf{Neut}, \mathsf{Egal}, \mathsf{Som}, \mathsf{Opp})$$

vérifiant les conditions suivantes :
– App *(pour appartenance)*
  – *prend en entrée une chaîne finie de bits, soit*
$$\forall k \in \mathbb{N}, Input_{\mathsf{App}}(k) = \{0,1\}^k \simeq [\![0, 2^k - 1]\!] ;$$
  – *renvoie* 0 *ou* 1*, soit* $Output = \{0,1\}$.

  On note alors $\widehat{\mathsf{G}}$ *l'ensemble des entrées de* $\{0,1\}^*$ *pour lesquelles* App *renvoie* 1 ;
– Neut *(pour neutre)*
  – *ne prend aucune donnée en entrée, soit*
$$\forall k \in \mathbb{N}, Input_{\mathsf{Neut}}(k) = \varnothing ;$$
  – *renvoie une chaîne de bits* e *de* $\widehat{\mathsf{G}}$*, soit* $Output = \{e\} \subset \widehat{\mathsf{G}}$;
– Egal *(pour égalité)*
  – *prend en entrée* a *et* b *dans* $\widehat{\mathsf{G}}$*, soit*
$$\forall k \in \mathbb{N}, Input_{\mathsf{Egal}}(k) = \left(\widehat{\mathsf{G}} \bigcap \{0,1\}^k\right)^2 ;$$
  – *renvoie* 0 *ou* 1*, soit* $Output = \{0,1\}$.

  Cet algorithme vérifie les propriétés suivantes :
  1. $\forall a \in \widehat{\mathsf{G}}, \mathsf{Egal}(a,a) = 1$;
  2. $\forall (a,b) \in \widehat{\mathsf{G}}^2, \mathsf{Egal}(a,b) = \mathsf{Egal}(b,a)$;
  3. $\forall (a,b,c) \in \widehat{\mathsf{G}}^3, \mathsf{Egal}(a,b) = \mathsf{Egal}(b,c) = 1 \Rightarrow \mathsf{Egal}(a,c) = 1$;
– Som *(pour somme)*
  – *prend en entrée* a *et* b *dans* $\widehat{\mathsf{G}}$;
  – *renvoie un élément de* $\widehat{\mathsf{G}}$*, soit* $Output = \widehat{\mathsf{G}}$;
– Opp *(pour opposé)*
  – *prend un élément* a *dans* $\widehat{\mathsf{G}}$;
  – *renvoie un élément de* $\widehat{\mathsf{G}}$.
Ces algorithmes doivent vérifier les propriétés suivantes :
  1. $\forall (a,b,c) \in \widehat{\mathsf{G}}^3, \mathsf{Egal}(\mathsf{Som}(\mathsf{Som}(a,b),c), \mathsf{Som}(a, \mathsf{Som}(b,c))) = 1$;
  2. $\forall a \in \widehat{\mathsf{G}}, \mathsf{Egal}(\mathsf{Som}(a, \mathsf{Neut}()), a) = 1$;
  3. $\forall a \in \widehat{\mathsf{G}}, \mathsf{Egal}(\mathsf{Som}(a, \mathsf{Opp}(a)), \mathsf{Neut}()) = 1$;
  4. $\forall (a,b) \in \widehat{\mathsf{G}}^2, \mathsf{Egal}(\mathsf{Som}(a,b), \mathsf{Som}(b,a)) = 1$.

Les propriétés 1. à 3. vérifiées par les algorithmes Egal, Som, Opp et Neut sont similaires à celles d'un groupe et la dernière propriété (4.) nous assure la commutativité de la loi $+$.
Ainsi la définition 3.1.2 est associée à celle de groupe abélien à partir de la définition 3.1.3.

Définition 3.1.3 *Soit* $\mathsf{G} = (\mathsf{App}, \mathsf{Neut}, \mathsf{Egal}, \mathsf{Som}, \mathsf{Opp})$ *un groupe effectif. Le* groupe représenté par G *est défini par* $(\rho(\mathsf{G}), +, -, e)$ *où :*

- *l'ensemble* $\rho(\mathsf{G})$ *est l'ensemble quotient de* $\widehat{\mathsf{G}}$ *par la relation d'équivalence* $\mathcal{R}$*, définie sur* $\widehat{\mathsf{G}}$ *par :*

$$a\mathcal{R}b \Leftrightarrow \mathsf{Egal}(a,b) = 1$$

  Soit a *dans* $\widehat{\mathsf{G}}$*, notons* $\rho(a)$ *la classe d'équivalence de* a *dans* $\rho(\mathsf{G})$ *;*
- *la loi de composition interne* $+$ *est définie par :* $\rho(a) + \rho(b) = \rho(\mathsf{Som}(a,b))$ *;*
- *la bijection* $-$ *de* $\rho(\mathsf{G})$ *est définie par :* $-\rho(a) = \rho(\mathsf{Opp}(a))$ *;*
- *l'élément neutre* $\rho(e)$ *est défini par* $\rho(e) = \rho(\mathsf{Neut}())$*.*

Preuve. Les propriétés 1. à 3. verifiées par l'algorithme Egal impliquent que la relation $\mathcal{R}$ est bien une relation d'équivalence sur $\widehat{G}$.
Ainsi, l'ensemble quotient $\rho(G)$ est bien défini.
De plus, les propriétés 1. à 4. de la définition 3.1.2 fournissent les propriétés 1. à 4. de la définition 3.1.1 d'un groupe abélien. □

Nous utiliserons essentiellement des groupes abéliens finis.
Néanmoins, cette définition permet de considérer des groupes effectifs représentant des groupes abéliens infinis, comme par exemple $\mathbb{Z}$.

Les groupes effectivement finis

Nous souhaiterions pouvoir considérer aussi bien un groupe que sa représentation algorithmique. Mais un groupe peut être représenté par différents groupes effectifs.
Ainsi, pour un groupe effectif donné nous noterons de la même façon le groupe qu'il représente, et nous dirons qu'il vérifie une propriété quand le groupe qu'il représente la vérifiera.
Par exemple, nous dirons d'un groupe effectif qu'il est fini s'il représente un groupe fini. Cette terminologie ne nous assure pas que les représentants des éléments du groupe aient tous une taille inférieure à une valeur fixée, comme l'expose l'exemple 3.1.1.

Exemple 3.1.1 *Soit* n *un entier naturel non nul, le groupe abélien fini* $\dfrac{\mathbb{Z}}{n\mathbb{Z}}$ *admet plusieurs représentations, par exemple :*
- *soit* G_1 *avec* $\widehat{\mathsf{G}_1} \simeq \mathbb{Z}$ *et* $a\mathcal{R}b$ *si et seulement si* n *divise* $a - b$ *;*
- *soit* G_2 *avec* $\widehat{\mathsf{G}_2} \simeq [\![0, n-1]\!]$ *et* $a\mathcal{R}b$ *si et seulement si* $a = b$ *;*

Le choix de la représentation donne alors des algorithmes d'égalité et de calculs différents.
Mais remarquons que, bien que le groupe $\dfrac{\mathbb{Z}}{n\mathbb{Z}}$ *soit fini, le groupe effectif* G_1 *lui associe des représentants de taille non bornée.*

Pour s'assurer que les représentants soient de taille bornée, nous introduisons la notion de groupe effectivement fini dans la définition 3.1.4.

Définition 3.1.4 *Un groupe effectif* G *est un* groupe effectivement fini *si et seulement s'il existe un entier* n *tel que* $\widehat{G} \subset \{0,1\}^n$*.*

Il ne faut pas confondre cette notion avec celle de groupe effectif fini, comme l'expose l'exemple 3.1.2.

Exemple 3.1.2 *Dans l'exemple 3.1.1, les groupes effectifs* G_1 *et* G_2 *sont des groupes effectifs finis mais seul le groupe* G_2 *est un groupe effectivement fini.*

Alors un groupe effectivement fini représente nécessairement un groupe fini. Et cette notion nous permet de définir la taille d'un groupe effectivement fini.

Définition 3.1.5 *Soit* G *un groupe effectivement fini, la* taille de G *est alors :*

$$T(\mathsf{G}) = min\{k \in \mathbb{N} : \widehat{\mathsf{G}} \subset \{0,1\}^k\}.$$

On appelle cardinal de G *le cardinal du groupe qu'il représente. On le note* $|\mathsf{G}|$.

Preuve. D'après la définition 3.1.4, il existe un entier n tel que $\widehat{G} \subset \{0,1\}^n$ donc la taille d'un groupe effectivement fini est bien définie. □

Il ne faut pas confondre les notions de taille et de cardinal d'un groupe effectif : la taille d'un groupe effectif correspond à la taille de ses éléments alors que son cardinal correspond au nombre d'éléments du groupe qu'il représente.
Plus précisément, la taille d'un groupe effectif permet de mesurer le nombre de bits nécessaires pour coder les éléments du groupe qu'il représente et ces deux notions vérifient la propriéte 3.1.1 :

Proposition 3.1.1 *Pour tout groupe effectif fini* G*, on a* $|\mathsf{G}| \leqslant 2^{T(\mathsf{G})}$.

Les groupes effectivement cycliques

Dans la suite, notre étude portera sur des groupes cycliques. Dans le cas de problèmes algorithmiques comme le problème du logarithme discret ou les problèmes de Diffie-Hellman, les résultats dépendent du choix du générateur.
Ainsi nous introduisons dans la définition 3.1.6 la notion de groupe effectivement cyclique qui précise le genérateur que l'on considère pour un groupe effectif cyclique donné.

Définition 3.1.6 *Un* groupe effectivement cyclique *est un couple* (G, P) *où* G *est un groupe effectif et* P *est un élément de* $\widehat{\mathsf{G}}$ *tel que* $\rho(P)$ *engendre* $\rho(\mathsf{G})$.

Comme pour la notion de groupe effectivement fini, il ne faut pas confondre les notions de groupe effectivement cyclique et groupe effectif cyclique : un groupe effectif cyclique est un groupe effectif représentant un groupe cyclique alors qu'un groupe effectivement cyclique est la donnée d'un groupe effectif cyclique et d'un représentant d'un générateur du groupe associé.

Famille effective de groupes

Définition 3.1.7 *Soit* (I, ϕ_I) *un ensemble effectif de taille* t_I *et* $\{\mathsf{G}_i\}_{i\in I}$ *une famille de groupes effectivement finis. Nous dirons que c'est une* famille effective de groupes *si et seulement si :*
- *il existe un entier* α *tel que pour tout* i *dans* I, $T(\mathsf{G}_i) < t_I(i)^\alpha$;
- *il existe un entier* β *tel que pour tout* i *dans* I*, les temps d'exécution des algorithmes* $\mathsf{App}_i, \mathsf{Neut}_i, \mathsf{Egal}_i, \mathsf{Som}_i, \mathsf{Opp}_i$ *composant le groupe effectif* G_i *sont majorés par* $T(\mathsf{G}_i)^\beta$.

La définition 3.1.7 permet de considérer une suite de groupes effectifs de plus en plus grands, comme par exemple les groupes effectifs représentant $\mathbb{F}_p$ avec p nombre premier. Dans ce contexte, elle précise :
- que les temps d'exécution des algorithmes considérés sont polynomiaux en la taille du groupe (seconde condition) ;
- que la taille du groupe est polynomiale en la taille des paramètres (première condition) ;

– que l'on peut trouver des polynômes majorant ces tailles et ces temps d'exécution dont le degré est indépendant de i.

Le temps de calcul effectif concernant la loi de groupe est alors polynomial en la taille du paramètre i, comme l'indique le lemme 3.1.1.

Lemme 3.1.1 *Soit (I, ϕ_I) un ensemble effectif de taille t_I, on considère une famille effective de groupes paramétrée par i dans I. Alors le temps d'execution des algorithmes composant chaque groupe est majoré par un polynôme en $t_I(i)$.*
Par exemple, il existe un entier C et un entier N tels que pour tout i dans I tel que $t_I(i) > N$, et pour tout (a, b) dans $\widehat{\mathsf{G}}_i^2$, on ait : $T_{\mathsf{Som}_i}(a, b) < t_I(i)^C$.

Preuve. On peut choisir C égal à $\alpha\beta$. D'après la défintion 3.1.7, il existera un entier N vérifiant la propriété demandée. □

Ainsi, le paramètre i sert de référence pour mesurer la taille des éléments traités par les algorithmes ainsi que leurs temps d'exécution.

Exemple 3.1.3 *Soit p un nombre premier, les groupes $(\mathbb{F}_p, +)$ et $(\mathbb{F}_p^*, \times)$ sont représentés par une famille effective de groupe paramétrée par p de taille $t_2(p)$ (cf. page 15). Ces groupes sont de taille $\lfloor \log_2(p) \rfloor + 1$.*

Ces algorithmes sont exposés par exemple dans [vzGG99]. En particulier, on y trouve en quatrième page de couverture un récapitulatif des différentes méthodes utilisées et leurs temps de calcul respectifs.

3.1.2 L'algorithme d'exponentiation rapide

La donnée d'un groupe effectif représentant un groupe $(G, +)$ permet de construire un algorithme de multiplication d'un élément du groupe par un entier relatif ; il s'agit de l'algorithme 2.

Algorithme 2 : Mult

Entrées : $\mathsf{G} = (\mathsf{App}, \mathsf{Neut}, \mathsf{Egal}, \mathsf{Som}, \mathsf{Opp})$
$n \in \mathbb{Z}$
$P \in \widehat{\mathsf{G}}$
Sortie : $Q \in \widehat{\mathsf{G}}$ *tel que* $\overline{Q} = n\overline{P}$

1. ***If*** $n < 0$
2. | ***then*** $Q := \mathsf{Mult}(\mathsf{G}, -n, \mathsf{Opp}(P))$
3. | ***Goto*** *17.*
4. | ***endif***
5. $Q := \mathsf{Neut}()$
6. ***If*** $n = 0$
7. | ***then Goto*** *17.*
8. | ***endif***
9. $m := \lfloor \log_2(n) \rfloor$
10. ***For*** i ***from*** m ***downto*** 0 ***do***
11. | $n_i := i^{eme}$ *bit représentant* $n = (a_m ... a_0)$
12. | $Q := \mathsf{Som}(Q, Q)$
13. | ***if*** $n_i = 1$
14. | | ***then*** $Q := \mathsf{Som}(Q, P)$
15. | | ***endif***
16. | ***endfor***
17. ***Return*** Q

Proposition 3.1.2 *L'algorithme* Mult, *défini en 2, est polynomial (en temps). Plus précisément, soit* (G, n, P) *une entrée de* Mult *avec* $\mathsf{G} = (\mathsf{App}, \mathsf{Neut}, \mathsf{Egal}, \mathsf{Som}, \mathsf{Opp})$ *; si* Som *s'exécute en temps* T_{Som}*, le temps d'exécution de* Mult *sur* (G, n, P) *est*

$$T_{\mathsf{Mult}}(\mathsf{G}, n, P) = O\left(\log(n) T_{\mathsf{Som}}(X, Y)\right)) \text{ où } X, Y \text{ sont des éléments quelconques de } \widehat{G}.$$

Preuve : La notation O et la notion de polynomialité sont définies en 1.3.5 et 1.3.6. Les notions de tailles utilisées sont définies dans les exemples 1.3.1, 1.3.2 et 1.3.4. En particulier, la taille des éléments du type (G, P) avec P élément de $\widehat{\mathsf{G}}$ est $T(\mathsf{G})$.
Concernant le résultat, il s'agit d'une adaptation de l'algorithme d'exponentiation rapide. Le calcul de son temps d'exécution se trouve par exemple dans le livre [CLR94, p.816]. □

Un intérêt de la notion de groupe effectif est, par exemple, de préciser quel type de représentation est utilisé en pratique, ou bien encore de pouvoir comparer deux types de représentation. Il est important dans la pratique de trouver un groupe effectif performant (par la rapidité du temps de calcul et/ou la taille des entrées).

Concernant les courbes elliptiques sur des corps finis, plusieurs candidats ont été étudiés : les écritures affines et projectives mais aussi, par exemple, les représentations jacobiennes (cf. [CC87] [CMO98]). On trouve une synthèse de ce type d'étude dans le chapitre IV du livre [BSS99].
Nous concernant, nous choisirons une écriture affine des points d'une courbe elliptique, exposée plus précisément dans l'exemple 3.1.4 :

Exemple 3.1.4 *Soit p un nombre premier différent de 2 et 3, soit a et b dans $\mathbb{F}_p$ tels que $4a^3 + 27b^2 \neq 0$. Considérons l'ensemble $[\![0, p-1]\!] \bigcup \{\infty\} \times [\![0, p-1]\!]$. Les points du groupe $E_{a,b}(\mathbb{F}_p)$ peuvent être représentées par :*
– tout couple $(\infty, \star)$ où $\star$ appartient à $[\![0, p-1]\!]$, pour le point à l'infini $[0:1:0]$,
– (x, y) dans $[\![0, p-1]\!]^2$, pour $[x : y : 1]$ dans $E_{a,b}(\mathbb{F}_p)$.
L'opposé d'un point, noté (x, y) est alors représenté par l'élément $(x, -y)$ et les formules usuelles, obtenues par la méthode de la corde et la tangente (cf par exemple [Joy95] page 51) permettent de construire un algorithme efficace calculant la somme de deux points. De cette façon, les groupes $E_{a,b}(\mathbb{F}_p)$ sont représentés par une famille effective de groupes effectifs indexés par (p, a, b) de taille $t_2(p)$ (cf. page 15).
La taille de ces groupes peut être majorée par $2t_2(p) + 1 = 2\lfloor \log_2(p) \rfloor + 3$.
Nous noterons alors $\mathsf{EC}_{p,a,b}$ le groupe effectif représentant $E_{a,b}(\mathbb{F}_p)$ avec ce type d'écriture.

3.2 Le groupe $E_{a,b}(\mathbb{F}_p[\varepsilon])$ en pratique

Dans cette section, p désigne un nombre premier distinct de 2 et 3, et (a, b) deux éléments de $\mathbb{F}_p[\varepsilon]^2$ tels que $\pi(4a^3 + 27b^2) \neq 0$ et on note $\mathbf{N}$ le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$.
Nous avons vu dans la section 2.1 que $E_{a,b}(\mathbb{F}_p[\varepsilon])$ est muni d'une structure de groupe. Dans la section 3.2, nous montrons que ce groupe est utilisable en pratique. Plus précisément, dans la partie 3.2.1 nous donnons explicitement les algorithmes permettant d'obtenir un groupe effectif représentant $E_{a,b}(\mathbb{F}_p[\varepsilon])$, puis dans la partie 3.2.2 nous donnons d'autres algorithmes utilisant ce groupe effectif.

3.2.1 Un groupe effectif représentant $E_{a,b}(\mathbb{F}_p[\varepsilon])$

Dans cette partie, nous allons exhiber un groupe effectif représentant $E_{a,b}(\mathbb{F}_p[\varepsilon])$.
Pour commencer, nous devons choisir un mode de représentation de ses éléments. En effet, ils peuvent être représentés de différentes façons. Par exemple, l'écriture projective donne plusieurs représentants à chaque élément de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et tester l'égalité de deux éléments nécessite alors quatre multiplications dans $\mathbb{F}_p[\varepsilon]$ (soit douze multiplications dans $\mathbb{F}_p$).
Par ailleurs, d'après les lemmes 2.2.1 et 2.2.2 ces éléments peuvent s'écrire de façon unique sous la forme $[k\varepsilon : 1 : 0]$ pour un point à l'infini ou $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ pour les autres points avec k, x_0, x_1, y_0, y_1 dans $\mathbb{F}_p$. Nous choisirons cette écriture unique pour représenter les éléments de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ en distinguant le cas des points à l'infini des autres points.

Plus précisément, concernant les points du type $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ nous allons enregistrer les quatre éléments de $\mathbb{F}_p$, x_0, x_1, y_0 et y_1. Nous pourrions ne pas enregistrer y_1 dans la mesure où nous pouvons le calculer à partir des variables x_0, x_1 et y_0 (quand cette dernière est non nulle). Ce choix nous obligerait à recalculer la variable y_1 lors du calcul de la somme de deux éléments et notamment à effectuer une division. Nous choisissons alors de représenter le point $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ par le couple $[(x_0, x_1), (y_0, y_1)]$.
Concernant les points du type $[k\varepsilon : 1 : 0]$, une seule donnée dans $\mathbb{F}_p$ est à enregistrer : k. Afin de garder le même format de représentant que précédemment, nous allons représenter ces points sous la forme $[(\infty, *), (k, *)]$ où ∞ est un caractère spécial et où $*$ désigne n'importe quel caractère entre 0 et $p-1$. Ainsi les points à l'infini auront plusieurs représentants.

Voici les cinq algorithmes qui, à partir de ces choix, permettent de définir un groupe effectif repésentant $E_{a,b}(\mathbb{F}_p[\varepsilon])$:

1. l'algorithme $\mathsf{App}_{p,a,b}$, (numéroté algorithme 3), détermine les choix de représentant :
   – un point à l'infini $[k\varepsilon : 1 : 0]$ sera représenté par tout élément de la forme $[(\infty, x_1), (k, y_1)]$ avec x_1, y_1 dans $\mathbb{F}_p$,
   – un point du type $[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1]$ sera représenté par le couple $[(x_0, x_1), (y_0, y_1)]$;

Algorithme 3 : $\mathsf{App}_{p,a,b}$

Entrées : $[(x_0, x_1), (y_0, y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
Sortie : 0 *ou* 1

output := 0
If $x_0 = \infty$
  then *output* := 1
  else $r_0 := y_0^2 - x_0^3 - a_0x_0 - b_0 \mod p$
    $r_1 := 2y_0y_1 - (3x_0^2 + a_0)x_1 - a_1x_0 - b_1 \mod p$
    if $(r_0, r_1) = (0, 0)$
      then *output* := 1
      endif
  endif
Return *output*

2. l'algorithme $\mathsf{Egal}_{p,a,b}$, (numéroté algorithme 4), détermine si deux couples représentent le même élément de $E_{a,b}(\mathbb{F}_p[\varepsilon])$.
   Evidemment, si l'un des deux est du type $[(\infty, x_1), (k, y_1)]$ et si le second s'écrit $[(X_0, X_1), (Y_0, Y_1)]$, l'algorithme doit tester seulement si les égalités $X_0 = \infty$ et $Y_0 = k$ sont vérifiées ;

Algorithme 4 : $\mathsf{Egal}_{p,a,b}$

Entrées : $[(x_0, x_1), (y_0, y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
$[(X_0, X_1), (Y_0, Y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
Sortie : 0 *ou* 1

output := 0
If $x_0 = \infty$ *or* $X_0 = \infty$
  then if $(x_0, y_0) = (X_0, Y_0)$
    then *output* := 1
    endif
  else if $[(x_0, x_1), (y_0, y_1)] = [(X_0, X_1), (Y_0, Y_1)]$
    then *output* := 1
    endif
  endif
Return *output*

3. l'algorithme $\mathsf{Neut}_{p,a,b}$, (numéroté algorithme 5), renvoie un représentant de l'élément neutre.
   L'élément $[0 : 1 : 0]$ admet pour représentant tout couple $[(\infty, x_1), (0, y_1)]$ et en particulier $[(\infty, 0), (0, 0)]$;

Algorithme 5 : $\mathsf{Neut}_{p,a,b}$

Entrées :
Sortie : $[(\infty, 0), (0, 0)]$

Return $[(\infty, 0), (0, 0)]$

4. l'algorithme $\mathsf{Opp}_{p,a,b}$, (numéroté algorithme 6), prend en entrée un représentant d'un élement de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et renvoie un représentant de l'opposé de cet élément.
Les éléments de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ vérifient :

$$-[k\varepsilon : 1 : 0] = [-k\varepsilon : 1 : 0] \text{ et } -[x_0 + x_1\varepsilon : y_0 + y_1\varepsilon : 1] = [x_0 + x_1\varepsilon : -y_0 - y_1\varepsilon : 1].$$

Le mode de représentation choisi vérifie alors

$$-[(x_0, x_1), (y_0, y_1)] = [(x_0, x_1), (p - y_0, p - y_1)],$$
$$\text{pour tout } [(x_0, x_1), (y_0, y_1)] \text{ dans } (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3.$$

Ainsi, l'algorithme $\mathsf{Opp}_{p,a,b}$ n'a pas besoin de tester si l'élément est à l'infini ;

Algorithme 6 : $\mathsf{Opp}_{p,a,b}$

Entrées : $P = [(x_0, x_1), (y_0, y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
Sortie : $-P$

1. $X_0 := x_0$
2. $X_1 := x_1$
3. $Y_0 := p - y_0$
4. $Y_1 := p - y_1$
5. ***Return*** $[(X_0, X_1), (Y_0, Y_1)]$

5. l'algorithme $\mathsf{Som}_{p,a,b}$, (numéroté algorithme 7), renvoie la somme de deux éléments à partir des relations établies dans la section 2.3 (lemmes 2.3.1, 2.3.2, 2.3.3, 2.3.4, 2.3.5 et 2.3.6) et synthétisées dans la table 2.1.

Algorithme 7 : $\mathsf{Som}_{p,a,b}$

Entrées : $P = [(x_0, x_1), (y_0, y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
$Q = [(X_0, X_1), (Y_0, Y_1)] \in (\llbracket 0, p-1 \rrbracket \bigcup \{\infty\}) \times \llbracket 0, p-1 \rrbracket^3$
Sortie : $Som = P + Q$

1. **If** $x_0 = \infty$
2. | **then if** $X_0 = \infty$
3. | | **then** $Som := [(\infty, 0), (x_1 + X_1 \mod p, 0)]$
4. | | **else** $Som := [(X_0, X_1 - 2Y_0 x_1 \mod p), (Y_0, Y_1 - (3X_0^2 + a_0)x_1 \mod p)]$
5. | | **endif**
6. | **else if** $X_0 = \infty$
7. | | **then** $Som := \mathsf{Som}_{p,a,b}(Q, P)$
8. | | **else if** $X_0 = x_0$
9. | | | **then if** $Y_0 \neq y_0$
10. | | | | **then** $Som := \left[(\infty, 0), \left((x_1 - X_1)(2Y_0)^{-1} \mod p, 0\right)\right]$
11. | | | | **Goto** *31.*
12. | | | | **else if** $Y_0 = 0$
13. | | | | | **then** $Som := \left[(\infty, 0), (-(y_1 + Y_1)(3X_0^2 + a_0)^{-1} \mod p, 0)\right]$
14. | | | | | **Goto** *31.*
15. | | | | | **else** $Int_1 := (2Y_0)^{-1} \mod p$
16. | | | | | $\lambda_0 := Int_1(3X_0^2 + a_0) \mod p$
17. | | | | | $\lambda_1 := Int_1\left(a_1 + 3X_0(x_1 + X_1) - \lambda_0(y_1 + Y_1)\right) \mod p$
18. | | | | | **endif**
19. | | | | **endif**
20. | | | **else** $Int_1 := (X_0 - x_0)^{-1} \mod p$
21. | | | $\lambda_0 := Int_1(Y_0 - y_0) \mod p$
22. | | | $\lambda_1 := Int_1\left(Y_1 - y_1 + \lambda_0(x_1 - X_1)\right) \mod p$
23. | | | **endif**
24. | | | $X_0' := \lambda_0^2 - x_0 - X_0 \mod p$
25. | | | $X_1' := 2\lambda_0\lambda_1 - x_1 - X_1 \mod p$
26. | | | $Y_0' := \lambda_0(x_0 - X_0') - y_0 \mod p$
27. | | | $Y_1' := \lambda_1(x_0 - X_0') + \lambda_0(x_1 - X_1') - y_1 \mod p$
28. | | | $Som := [(X_0', X_1'), (Y_0', Y_1')]$
29. | | **endif**
30. | **endif**
31. **Return** *Som*

Nous venons de détailler les différents algorithmes qui nous serviront par la suite, étudions maintenant le temps d'exécution de chacun d'eux.

Les entiers considérés sont compris entre 0 et $p-1$ et nous notons $\mathbf{I}(p), \mathbf{M}(p)$ et $\mathbf{S}(p)$ les temps d'exécution de l'inversion, du produit et de l'élévation au carré dans $\mathbb{F}_p$. Ces temps d'exécution dépendent du groupe effectif choisi pour représenter $\mathbb{F}_p$ et de la puissance des machines utilisées. Par exemple, on peut raisonnablement estimer que $\mathbf{S}(p) = 0.8\mathbf{M}(p)$ (cf. [CJLM06, p.2-3]) et que $\mathbf{I} = \alpha\mathbf{M}$ avec α compris entre 3 et 10 (cf. [BSS99, p.72]).
On peut tout de même supposer qu'ils vérifient :

$$\text{pour } p \text{ nombre premier supérieur à } 5,\ \mathbf{S}(p) \leqslant \mathbf{M}(p) \leqslant \mathbf{I}(p).$$

Nous supposons aussi que le test d'égalité et que le calcul de la somme de deux entiers ont un temps d'exécution négligeable devant $\mathbf{S}(p)$.

Décompte des opérations	Total **M**	Total **S**
4. : **S S M M A A A**	2	2
5. : **M A** (**S** déjà effectué en *4*) **A A A M M A A A**	3	0

TAB. 3.1 – Calcul de complexité de $\mathsf{App}_{p,a,b}$.

	Décompte des opérations		Total **I**	Total **M**	Total **S**
Si $x_0 = X_0 = \infty$	*3.* : **A**		0	0	0
Si $x_0 = \infty$ et $X_0 \neq \infty$	*4.* : **M A A**, **S A A A M A**		0	2	1
Si $x_0 = X_0$ et $y_0 \neq Y_0$	*10.* : **A I A M**		1	1	0
Si $x_0 = X_0$ et $y_0 = Y_0 = 0$	*13.* : **S A A A I A M**		1	1	1
Si $x_0 = X_0$ et $y_0 = Y_0 \neq 0$	*15.* : **A I**				
	16. : **S A A A M**	*24.* : **S A A**			
	17. : **A M A M A A A A M**	*25.* : **M A A A**	1	8	2
Si $x_0 \neq X_0$	*20.* : **A I**	*26.* : **A M A**			
	21. : **A M**	*27.* : **A M A M A A**			
	22. : **A M A A M**		1	7	1

TAB. 3.2 – Calcul de complexité de $\mathsf{Som}_{p,a,b}$

Concernant chacun des algorithmes 3 à 7, on obtient comme temps d'exécution, pour P et Q représentants d'éléments de $E_{a,b}(\mathbb{F}_p[\varepsilon])$:

1. pour $\mathsf{App}_{p,a,b}$: $T_{\mathsf{App}_{p,a,b}}(P) = O(5\mathbf{M}(p) + 2\mathbf{S}(p))$.
   Le détail des opérations effectuées à chaque instruction est donné dans la table 3.1. Le décompte des opérations s'effectue de la gauche vers la droite avant d'additionner tous les termes. De plus, les multiplications par de petits nombres ont été effectuées à l'aide d'additions : par exemple calculer $3x$ s'opère en deux additions. Enfin, dans la table 3.1 la lettre **A** représente l'addition modulaire et la dernière colonne rend compte du nombre d'opérations **S** et **M** effectuées à chaque instruction.
   Remarque. On peut encore supprimer un calcul de carré, en enregistrant x_0^2 que l'on utilise deux fois (dans l'instruction 4. pour calculer x_0^3 puis dans l'instruction 5.) ;
2. pour $\mathsf{Egal}_{p,a,b}$: $T_{\mathsf{Egal}_{p,a,b}}(P, Q) = O(1)$;
3. pour $\mathsf{Neut}_{p,a,b}$: $T_{\mathsf{Neut}_{p,a,b}}() = O(1)$;
4. pour $\mathsf{Opp}_{p,a,b}$: $T_{\mathsf{Opp}_{p,a,b}}(P) = O(1)$;
5. pour $\mathsf{Som}_{p,a,b}$: $T_{\mathsf{Som}_{p,a,b}}(P, Q) = O(\mathbf{I}(p) + 8\mathbf{M}(p) + 2\mathbf{S}(p))$.
   La table 3.2 donne le détail des opérations effectuées dans chaque boucle if et sa dernière colonne rend compte du total des opérations **S**, **M**, **I** effectuées selon les valeurs des entrées.
   Remarque. Les temps de calcul les plus longs s'effectuent dans les deux derniers cas, à savoir, par ordre croissant dans le cas général et dans le cas de la tangente " non verticale".

Les groupes de type $\mathbb{F}_p$ sont représentés par une famille effective de groupes indexée par p de taille $t_2(p)$. Cela signifie entre autre, que les temps de calcul $\mathbf{S}(p)$, $\mathbf{M}(p)$ et $\mathbf{I}(p)$ sont majorés par des polynômes en $\log(p)$. Ainsi ces cinq algorithmes $\mathsf{App}_{p,a,b}, \mathsf{Egal}_{p,a,b}, \mathsf{Neut}_{p,a,b}$,

$\mathsf{Opp}_{p,a,b}$ et $\mathsf{Som}_{p,a,b}$ sont polynomiaux.

Il reste alors à montrer qu'ils définissent bien un groupe effectif. Nous n'effectuerons pas ce travail et nous supposerons vraie l'assertion suivante :

Assertion.
Le quintuplet $\mathsf{EC}^{\varepsilon}_{p,a,b} = \left(\mathsf{App}_{p,a,b}, \mathsf{Egal}_{p,a,b}, \mathsf{Neut}_{p,a,b}, \mathsf{Opp}_{p,a,b}\mathsf{Som}_{p,a,b}\right)$ est un groupe effectif représentant le groupe $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

3.2.2 Autres algorithmes liés à $E_{a,b}(\mathbb{F}_p[\varepsilon])$

Rappelons que $\mathbf{N}$ désigne le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$. Nous avons vu dans la section 2.2.2 que lorsque p ne divise pas $\mathbf{N}$ les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p$ sont isomorphes.

Dans cette section, nous étudions l'effectivité de cet isomorphisme.

Pour cela, rappelons son expression :

$$\begin{array}{rcl} \Lambda : E_{a,b}(\mathbb{F}_p[\varepsilon]) & \longrightarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p \\ P & \mapsto & (\overline{P}, \lceil \mathbf{N}\mathbf{N}' P \rfloor) \end{array}$$

et

$$\begin{array}{rcl} \Lambda^{-1} : E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p & \to & E_{a,b}(\mathbb{F}_p[\varepsilon]) \\ (P, \kappa) & \mapsto & (1 - \mathbf{N}\mathbf{N}')P' + \Theta(\kappa) \end{array}$$

où P' est un élément de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ vérifiant $\overline{P'} = P$.

Dans l'expression de $\Lambda(P)$, le calcul de la projection de P sur $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ notée $\overline{P}$ ne pose pas de difficulté.
De la même façon, dans l'expression de $\Lambda^{-1}(P, \kappa)$, le calcul d'un relèvement de P dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ nécessite peu de calculs. Le reste des opérations concerne des sommes et des multiplications qui restent effectives et dont nous pouvons évaluer le temps de calcul.

La difficulté majeure reste le calcul du cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ noté $\mathbf{N}$. Mais ce calcul ne s'avère pas si difficile. En effet, ce problème a trouvé une amorce de solution à partir de 1985 grâce à R. Schoof qui donne dans [Sch85] un algorithme permettant de calculer cette valeur en temps polynomial. Plus précisément, le temps d'exécution de cet algorithme est en $O(\log^8(p))$. Cet algorithme restait néanmoins difficile à implémenter. Depuis, il a été amélioré aussi bien pour de petites que pour de grandes valeurs de p. En ce qui concerne les grandes caractéristiques, les idées de A.O.L. Atkin et N.D. Elkies se développent à partir de 1988 au travers des manuscrits [Atk88] et [Elk92]. R. Schoof publie ces améliorations et donne un algorithme efficace dans [Sch95]. La même année, N.D. Elkies expose ces idées dans une conférence donnée en l'honneur de A.O.L. Atkin qu'il publiera en 1998 dans [Elk98].
Aujourd'hui, l'algorithme SEA (du nom des ses trois auteurs) permet de calculer $\mathbf{N}$ avec un temps d'exécution estimé à $O\left(\log^{4+\epsilon}(p)\right)$ où $\epsilon < 1$ (cf. [Gau04]).
Une synthèse de cet algorithme est exposé par exemple dans [Ler97] (de façon plus concise dans [Fou01]).

Ainsi les morphismes Λ et Λ^{-1} sont calculables en temps polynomial.

Voici en détail, les algorithmes polynomiaux $\mathsf{Lambda}_{p,a,b}$ et $\mathsf{Lambda}^{-1}_{p,a,b}$ calculant les images de Λ et Λ^{-1} dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p$ et $E_{a,b}(\mathbb{F}_p[\varepsilon])$. Ils sont numérotés respectivement 8 et 9.

Algorithme 8 : $\mathsf{Lambda}_{p,a,b}$

Entrées : $P = [(x_0, x_1), (y_0, y_1)] \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
Sortie : $\Lambda(P)$

1. $\mathbf{N} := Card\left(E_{\pi(a),\pi(b)}(\mathbb{F}_p)\right)$
2. $\mathbf{N}' := \mathbf{N}^{-1} \mod p$
3. $u := \mathbf{N}\mathbf{N}'$
4. $U := uP$
5. $\kappa := U[2][1]$ *(i.e. $\kappa = Y_0$ pour $U := [(X_0, X_1), (Y_0, Y_1)]$)*
6. ***Return*** $([x_0, y_0], \kappa)$

Algorithme 9 : $\mathsf{Lambda}^{-1}_{p,a,b}$

Entrées : $P = [x_0, y_0] \in \widehat{\mathsf{EC}_{p,a,b}}$
$\kappa \in [\![0, p-1]\!]$
Sortie : $\Lambda^{-1}(P, \kappa)$

1. $\mathbf{N} := Card\left(E_{\pi(a),\pi(b)}(\mathbb{F}_p)\right)$
2. $\mathbf{N}' := \mathbf{N}^{-1} \mod p$
3. $y_1 := (a_1 x_0 + b_1)(2y_0)^{-1} \mod p$
4. $P' := [(x_0, 0), (y_0, y_1)]$
5. $u := 1 - \mathbf{N}\mathbf{N}' \mod \mathbf{N}p$
6. $U := uP' + [(\infty, 0), (\kappa, 0)]$
7. ***Return*** U

Le lemme 3.2.1 précise leur temps d'exécution.

Lemme 3.2.1 *Les isomorphismes Λ et Λ^{-1} définis entre $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p$ et $E_{a,b}(\mathbb{F}_p[\varepsilon])$ sont calculables en temps polynomial ; plus précisément leur complexité est en $O\left(\log^{4+\epsilon}(p)\right)$.*

Preuve.

L'isomorphisme $\Lambda_{p,a,b}$ est défini dans le lemme 2.2.6 par :

$$\text{Pour } P \text{ dans } E_{a,b}(\mathbb{F}_p[\varepsilon]),\ \Lambda(P) = (\overline{P}, \lceil \mathbf{N}\mathbf{N}'P \rfloor)$$
$$\text{où } \mathbf{N} \text{ est le cardinal de } E_{\pi(a),\pi(b)}(\mathbb{F}_p) \text{ et } \mathbf{N}' = \mathbf{N}^{-1} \mod p.$$

En particulier, si P est représenté par $[(x_0, x_1), (y_0, y_1)]$ de $\mathsf{EC}^{\varepsilon}_{p,a,b}$ alors l'élément $[x_0, y_0]$ représente bien le point $\overline{P}$ de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$. Aussi, en supposant que $\mathsf{EC}^{\varepsilon}_{p,a,b}$ représente bien le groupe $E_{a,b}(\mathbb{F}_p[\varepsilon])$, l'élément $\mathbf{N}\mathbf{N}'P$ est un point à l'infini donc sa représentation est du type $[(\infty, x_1), (y_0, y_1)]$ (cf. Algorithme 1 : $\mathsf{App}_{p,a,b}$) et il représente l'élement $\Theta(y_0)$.

L'algorithme $\mathsf{Lambda}_{p,a,b}$ renvoie bien un représentant de $\Lambda(P)$ dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \times \mathbb{F}_p$.

L'isomorphisme $\Lambda^{-1}_{p,a,b}$ est défini dans le lemme 2.2.6, page 50, par :

$$\text{Pour } P \text{ dans } E_{\pi(a),\pi(b)}(\mathbb{F}_p) \text{ et } \kappa \text{ dans } \mathbb{F}_p,\ \Lambda^{-1}(P, \kappa) = (1 - \mathbf{N}\mathbf{N}')\, P' + \Theta(\kappa)$$
$$\text{où } P' \text{ est un élément quelconque de } E_{a,b}(\mathbb{F}_p[\varepsilon]) \text{ tel que } \overline{P'} = P.$$

En particulier, si P est représenté par (x_0, y_0) alors l'élément $\left[(x_0, 0), \left(y_0, \frac{a_1 x_0 + b_1}{2y_0}\right)\right]$ de $\mathsf{EC}^{\varepsilon}_{p,a,b}$ représente bien un point P' de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ tel que $\overline{P'} = P$:
– si $x_0 = \infty$ c'est immédiat ;
– sinon il suffit de vérifier les relations 2.2 (introduites page 47 avec $Z_0 = 1$ et $Z_1 = 0$). L'algorithme $\mathsf{Lambda}^{-1}_{p,a,b}$ renvoie alors un représentant de $(1 - \mathbf{N}\mathbf{N}')\,P' + \Theta(\kappa)$ et donc calcule bien $\Lambda^{-1}(P, \kappa)$ dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

Les tables 3.3 et 3.4 précisent la complexité de chacune des instructions des algorithmes $\mathsf{Lambda}_{p,a,b}$ et $\mathsf{Lambda}^{-1}_{p,a,b}$. La notation SEA y représente l'algorithme SEA introduit page 70.

Instruction	Algorithme(s)	Nombre d'opérations dans $\mathbb{F}_p$
2. 3. 5.	Opérations élémentaires dans $\mathbb{F}_p$	$O(1)$
1.	SEA	$O(\log^{4+\epsilon}(p))$
4.	$\mathsf{Mult}_{\mathsf{p,a,b}}$	$O\left(\log\left(p\mathbf{N}\right)\right)$

TAB. 3.3 – Calcul de complexité de $\mathsf{Lambda}_{p,a,b}$.

Instruction	Algorithme(s)	Nombre d'opérations dans $\mathbb{F}_p$
2. 3. 4. 5.	Opérations élémentaires dans $\mathbb{F}_p$	$O(1)$
1.	SEA	$O(\log^{4+\epsilon}(p))$
6.	$\mathsf{Mult}_{p,a,b}$	$O\left(\log\left(p\mathbf{N}\right)\right)$
6.	$\mathsf{Som}_{p,a,b}$	$O(1)$

TAB. 3.4 – Calcul de complexité de $\mathsf{Lambda}^{-1}_{p,a,b}$.

D'après le théorème de Hasse (cf. [Sil85] page 131 ou [Joy95] page 55), $\mathbf{N}$ appartient à l'intervalle $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$. Ainsi, l'expression $O\left(\log\left(p\mathbf{N}\right)\right)$ est majorée par $O\left(\log(p)\right)$ et on en déduit le résultat. □

3.3 Equivalence des problèmes paramétrés

Dans cette section, nous allons étudier les relations entre les *problèmes de logarithme discret* (DL), les *problèmes calculatoires de Diffie-Hellman* (CDH) et les *problèmes décisionnels de Diffie-Hellman* (DDH) sur les groupes $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ et $E_{a,b}(\mathbb{F}_q[\varepsilon])$.
Dans la section 1.3.5, nous avons défini ce type de problème uniquement pour des groupes cyliques. Ainsi, nous souhaitons resteindre cette étude aux cas où les groupes $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ et $E_{a,b}(\mathbb{F}_q[\varepsilon])$ sont cycliques.
Dans cet objectif, nous posons les conditions suivantes :

1. $\mathbf{N}$ le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ est un nombre premier différent de p : ainsi le groupe $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ est bien cyclique (puisqu'il est de cardinal premier). Cette condition nous assure également que $E_{a,b}(\mathbb{F}_q[\varepsilon])$ est isomorphe à $E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$ (d'après le corollaire 2.2.2, puisque p ne divise pas $\mathbf{N}$),
2. $q = p$: cette condition est nécessaire pour qu'un groupe $E_{a,b}(\mathbb{F}_q[\varepsilon])$ vérifiant la condition 1. soit cyclique . En effet :

- $E_{a,b}(\mathbb{F}_q[\varepsilon])$ est isomorphe à $E_{\pi(a),\pi(b)}(\mathbb{F}_q) \times \mathbb{F}_q$ d'après la condition 1. ;
- $\mathbb{F}_q$ n'est pas cyclique si q n'est pas un nombre premier ;

Donc, un groupe $E_{a,b}(\mathbb{F}_q[\varepsilon])$ vérifiant la condition 1. ne peut être cyclique si $q \neq p$.

Les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$ vérifiant les conditions 1. et 2. sont alors cycliques eux aussi puisqu'ils sont isomorphes à $E_{a,b}(\mathbb{F}_p) \times \mathbb{F}_p$ avec $E_{a,b}(\mathbb{F}_p)$ cyclique et de cardinal premier avec p.

Précisons également que si p divise $\mathbf{N}$, cette étude ne présente guère d'intérêt puisque, dans ce cas, le problème du logarithme discret dans $E_{a,b}(\mathbb{F}_p)$ est résolu en temps polynomial (cf. [AS98], [Sem98], [Sma99]). Nous y reviendrons aussi dans la section 4.1.

Ainsi, nous introduisons l'ensemble :

$$I_\varepsilon = \{(p,a,b,P) : p \text{ premier}, (a,b) \in \mathbb{F}_p^2[\varepsilon], \#E_{a,b}(\mathbb{F}_p) = \mathbf{N}, p \nmid \mathbf{N}, P \in E_{a,b}(\mathbb{F}_p[\varepsilon]), \langle P \rangle = E_{a,b}(\mathbb{F}_p[\varepsilon])\}$$

muni de la taille $t_{I_\varepsilon}(p,a,b,P) = t_2(p)$.
L'étude qui suit porte sur les courbes elliptiques paramétrées par (p,a,b,P) dans I_ε. Elles sont définies sur des anneaux de nombres duaux $\mathbb{F}_p[\varepsilon]$ de grande caractéristique et dont le cardinal de la projection sur $\mathbb{F}_p$ est un nombre premier différent de p.

La partie 3.3.1 traite des problèmes calculatoires DL et CDH (définis dans la section 1.3.5). Elle montre l'équivalence entre ces problèmes calculatoires sur les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et $E_{a,b}(\mathbb{F}_p)$ paramétrés par I_ε (c'est à dire vérifiant les conditions 1. et 2.).
La partie 3.3.2 traite du problème décisionnel DDH (défini dans la sous-section 1.3.5). Elle montre que le problème DDH sur les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$ vérifiant les conditions 1. et 2. sont faciles à résoudre.

3.3.1 Problèmes calculatoires : DL et CDH

Cette section aborde uniquement les problèmes calculatoires entre les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et $E_{a,b}(\mathbb{F}_p)$ vérifiant les conditions 1. et 2..

Nous allons montrer que les problèmes DL et CDH entre ces deux types de groupes sont équivalents.

Ce résultat paraît naturel dans la mesure où nous savons :
- qu'il existe un isomorphisme entre $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et $E_{a,b}(\mathbb{F}_p) \times \mathbb{F}_p$ calculable en temps polynomial (cf. section 3.2) ;
- que le problème du logarithme discret est trivial sur $(\mathbb{F}_p, +)$.
- que le caractère calculatoire de ces problème impose de résoudre les problèmes calculatoires correspondants sur $E_{a,b}(\mathbb{F}_p)$ et $\mathbb{F}_p$ pour résoudre le problème sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

Commençons par étudier le cas du problème du logarithme discret.

Nous allons montrer que les problèmes $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ et $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$ paramétrés par (p,a,b,P) dans I_ε sont équivalents.

Equivalence des problèmes DL

Dans ce paragraphe, nous allons montrer que chacun de ces deux problèmes de logarithme discret se réduit à l'autre. Pour chacune de ces réductions, nous allons suivre la méthode

suivante :
- exprimer un logarithme en fonction de l'autre ;
- montrer que la relation établie permet de construire une réduction du premier problème vers le second ;
- évaluer le temps de calcul arithmétique de cette réduction et montrer qu'il est polynomial en $t_2(p)$.

Ainsi, nous montrerons que le premier problème paramétré se réduit au second.

Commençons par la réduction la moins évidente, celle de $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$.
Dans ce cas, la difficulté consiste à calculer le logarithme discret de base P dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ à partir du logarithme discret de base $\overline{P}$ dans $E_{a,b}(\mathbb{F}_p)$. Pour cela, nous allons utiliser l'isomorphisme Λ du lemme 2.2.6 pour établir une relation où apparaît un logarithme discret dans $\mathbb{F}_p$. Ce dernier étant facile à calculer, nous montrerons ensuite que cette relation, nous permet de construire une réduction en temps polynomial en $t_2(p)$ de $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$.
Cette relation est établie dans le lemme 3.3.1, puis les considérations effectives sont étudiées dans le lemme 3.3.2.

Lemme 3.3.1 *Soit (p, a, b, P) dans I_ε, soit Q dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$, le logarithme discret de Q en base P est l'unique entier compris entre 0 et $\mathbf{N}p - 1$ représentant dans $\dfrac{\mathbb{Z}}{\mathbf{N}p\mathbb{Z}}$:*

$$\left(1 - \mathbf{N}\mathbf{N}'\right) \log_{\overline{P}}\left(\overline{Q}\right) + \mathbf{N}\mathbf{N}' \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right).$$

Preuve. Soit Λ l'isomorphisme défini dans le lemme 2.2.6 et c l'entier tel que $\mathbf{N}\mathbf{N}' = 1 + cp$.
Les groupes $E_{a,b}(\mathbb{F}_p[\varepsilon])$, $E_{a,b}(\mathbb{F}_p)$ et $\mathbb{F}_p$ sont, respectivement, d'ordre $\mathbf{N}p$, $\mathbf{N}$ et p.
Pour tout entier k, on a les relations :

$$\begin{aligned}
&\left(\left(1 - \mathbf{N}\mathbf{N}'\right) \log_{\overline{P}}\left(\overline{Q}\right) + \mathbf{N}\mathbf{N}' \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right) + k\mathbf{N}p\right) \Lambda(P) \\
&= \left(\left(\left(1 - \mathbf{N}\mathbf{N}'\right) \log_{\overline{P}}\left(\overline{Q}\right) + \mathbf{N}\mathbf{N}' \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right)\right) \overline{P}, \right. \\
&\qquad \left. \left((-cp) \log_{\overline{P}}\left(\overline{Q}\right) + (1 + cp) \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right)\right) \lceil \mathbf{N}\mathbf{N}'P \rfloor\right) \\
&= \left(\log_{\overline{P}}\left(\overline{Q}\right) \overline{P}, \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right) \lceil \mathbf{N}\mathbf{N}'P \rfloor\right) = \left(\overline{Q}, \mathbf{N}' \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right) \lceil \mathbf{N}P \rfloor\right) \\
&= \left(\overline{Q}, \mathbf{N}' \lceil \mathbf{N}Q \rfloor\right) = \Lambda(Q)
\end{aligned}$$

Ainsi : $\left(\left(1 - \mathbf{N}\mathbf{N}'\right) \log_{\overline{P}}\left(\overline{Q}\right) + \mathbf{N}\mathbf{N}' \log_{\lceil \mathbf{N}P \rfloor}\left(\lceil \mathbf{N}Q \rfloor\right) + k\mathbf{N}p\right) P = Q$. Le résultat suit. □

Le lemme 3.3.1 nous permet de calculer facilement le logarithme discret de Q en base P à partir du logarithme discret de $\overline{Q}$ en base $\overline{P}$. En effet, les variables $\lceil \mathbf{N}Q \rfloor$ et $\lceil \mathbf{N}P \rfloor$ sont dans le groupe cyclique $(\mathbb{F}_p, +)$. Ainsi, le logarithme discret de $\lceil \mathbf{N}Q \rfloor$ en base $\lceil \mathbf{N}P \rfloor$ n'est rien d'autre que le quotient dans $\mathbb{F}_p$ de $\lceil \mathbf{N}Q \rfloor$ par $\lceil \mathbf{N}P \rfloor$. L'application du lemme 3.3.1 nous donne alors, la réduction énoncée au lemme 3.3.2.

Lemme 3.3.2 *Soit un ensemble effectif infini $(I_\varepsilon, \phi_{I_\varepsilon})$ muni de la taille $t_2(p)$, le problème $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ paramétré par (p, a, b, P) dans I_ε se réduit au problème paramétré $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$.*

Preuve. Soit (p, a, b, P) dans I_ε, commençons par donner la réduction $(\mathsf{A}^1, \mathsf{B}^1)$ de $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$. Les algorithmes A^1 et B^1 sont numérotés algorithmes 10 et 11 respectivement.

Algorithme 10 : A^1

Entrées : (p, a, b, P) *dans* I_ε
$Q = [(x_0, x_1), (y_0, y_1)] \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$

Sortie : $\overline{Q} \in \widehat{\mathsf{EC}_{p,a,b}}$

1. $\overline{Q} := [x_0, y_0]$
2. *RETURN* $(\overline{Q})$

Pour une entrée $((p, a, b, P), Q)$, l'algorithme A^1 renvoie $\overline{Q}$ dans $E_{a,b}(\mathbb{F}_p)$.

Algorithme 11 : B^1

Entrées : (p, a, b, P) *dans* I_ε
$Q = [(x_0, x_1), (y_0, y_1)] \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$n \in \mathbb{Z}$

Sortie : $n' \in [\![0, \mathbf{N}p - 1]\!]$

1. $\mathbf{N} := \mathsf{Card}(\mathsf{E}_{a,b}(\mathbb{F}_p))$
2. $n := n \mod \mathbf{N}$
3. $c := \lceil \mathbf{N}P \rfloor$
4. $d := \lceil \mathbf{N}Q \rfloor$
5. $r := dc^{-1} \mod p$
6. $\mathbf{N}' := \mathbf{N}^{-1} \mod p$
7. $q := \mathbf{N}'(r - n) \mod p$
8. $n' := n + \mathbf{N}q$
9. *RETURN* (n')

Pour une entrée $((p, a, b, P), Q, n)$, l'algorithme 11 renvoie $n + \mathbf{N}\mathbf{N}'(r - n)$ où r est le logarithme discret de $\lceil \mathbf{N}Q \rfloor$ dans $\mathbb{F}_p$ de base $\lceil \mathbf{N}P \rfloor$.

Vérifions que ce couple d'algorithmes est bien une réduction des problèmes paramétrés $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$.
Pour cela, le lemme 3.3.1, nous assure que pour Q dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$, on a

$$\log_P(Q) = \mathsf{B}^1\left((p, a, b, P), Q, \log_{\overline{P}}\left(\mathsf{A}^1\left((p, a, b, P)(Q)\right)\right)\right).$$

Evaluons ensuite le temps de calcul arithmétique de cette réduction. L'algorithme A^1 ne nécessite aucune opération dans $\mathbb{F}_p$ et l'algorithme B^1 a un temps d'exécution polynomial en $\log(p)$ d'après le décompte des opérations présenté dans la table 3.5 ($\mathbf{I}$ et $\mathbf{M}$ y représentent les opérations d'inversion et de multiplication dans $\mathbb{F}_p$ i.e. modulo p).
La réduction $(\mathsf{A}^1, \mathsf{B}^1)$ est donc polynomiale en $t_2(p)$ et résout le problème du logarithme discret dans $(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ à partir d'une solution du problème du logarithme discret dans $(E_{a,b}(\mathbb{F}_p), \overline{P})$. Le résultat suit. □

Illustrons le fonctionnement de cette réduction par un exemple :

			Total
Décompte des opérations	**I**	**M**	Complexité arithmétique
1 : Card			$O(\log^{4+\epsilon}(p))$
2 : mod			$O(1)$ car $\mathbf{N} = O(p)$
3 : Mult			$O(\log(\mathbf{N})) = O(\log(p))$
4 : Mult			$O(\log(\mathbf{N})) = O(\log(p))$
5 : **I M**	1	0	$O(1)$
6 : **I**	1	0	$O(1)$
7 : **A M**	0	1	$O(1)$
8 : **M A**	0	1	$O(1)$

TAB. 3.5 – Calcul de complexité arithmétique de B^1.

Exemple 3.3.1 *Considérons la courbe d'équation* $y^2 = x^3 + (333 + 623\varepsilon)x + 394 + 34\varepsilon$ *dans* $\mathbb{F}_{701}$. *Le groupe* $E_{333,394}(\mathbb{F}_{701})$ *est d'ordre* 733, *qui est bien un nombre premier.*
L'élément $P = (42 + 137\varepsilon, 172 + 464\varepsilon)$ *est un générateur du groupe* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$.

Considérons donc $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *dans* I_ε.
Soit $Q = (161 + 11\varepsilon, 164 + 59\varepsilon)$. *Calculons le logarithme discret de* Q *en base* P *sachant que dans* $E_{333,394}(\mathbb{F}_{701})$, *les points* $\overline{Q}$ *et* $\overline{P}$ *vérifient* $\overline{Q} = 182\overline{P}$.

L'algorithme A^1 *renvoie :* $\overline{Q} = (161, 164)$.

Puis l'algorithme B^1 *prend en entrée* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$, $Q = (161 + 11\varepsilon, 164 + 59\varepsilon)$ *et* $n = 182$. *Son exécution donne :*

1. $\mathbf{N} := 733$
2. $n := 182 \mod 733 = 182$
3. $c := \lceil 733(42 + 137\varepsilon, 172 + 464\varepsilon) \rfloor = \lceil \mathcal{O}_{145} \rfloor = 145$
4. $d := \lceil 733(161 + 11\varepsilon, 164 + 59\varepsilon) \rfloor = \lceil \mathcal{O}_{159} \rfloor = 159$
5. $r := 159 \times 145^{-1} \mod 701 = 296$
6. $\mathbf{N}' := 733^{-1} \mod 701 = 241$
7. $q := 241(296 - 182) \mod 701 = 135$
8. $n' := 182 + 733 \times 135 = 99137$
9. *RETURN* (99137).

L'algorithme B^1 *renvoie* 99137, *dont nous pouvons vérifier qu'il est bien le logarithme discret de* Q *en base* P *par l'égalité :*

$$99137(42 + 137\varepsilon, 172 + 464\varepsilon) = (161 + 11\varepsilon, 164 + 59\varepsilon).$$

Etablissons maintenant la réduction inverse, à savoir de $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$.

Dans ce cas, la difficulté consiste à reconstituer une entrée du problème du logarithme discret dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ à partir d'une entrée du problème du logarithme discret dans $E_{a,b}(\mathbb{F}_p)$. Pour cela, le lemme 3.3.3 nous indique que, pour une entrée Q dans $E_{a,b}(\mathbb{F}_p)$, nous pouvons choisir comme entrée dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ un relevé quelconque Q' de Q.
Cette propriété est établie dans le lemme 3.3.3, puis les considérations effectives qui en découlent sont étudiées dans le lemme 3.3.4.

Lemme 3.3.3 *Soit (p, a, b, P) dans I_ε, soit Q dans $E_{a,b}(\mathbb{F}_p)$, on a :*

- *$\overline{P}$ est un générateur de $E_{a,b}(\mathbb{F}_p)$;*
- *le logarithme discret de Q en base $\overline{P}$ est l'unique entier entre 0 et $\mathbf{N}-1$ représentant :*

$$\log_P(Q') \text{ dans } \frac{\mathbb{Z}}{\mathbf{N}\mathbb{Z}}$$

où Q' est un relevé quelconque de Q dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

Preuve. Soit Q' un relevé de Q, on a $Q' = \log_P(Q')P$ et, par projection, $\overline{Q'} = \log_P(Q')\overline{P}$. De plus, le groupe $E_{a,b}(\mathbb{F}_p)$ est d'ordre $\mathbf{N}$, donc pour tout entier k, $(\log_P(Q') + k\mathbf{N})\,\overline{P} = Q$. □

L'application du lemme 3.3.3 nous donne la réduction énoncée au lemme 3.3.4.

Lemme 3.3.4 *Soit un ensemble effectif infini $(I_\varepsilon, \phi_{I_\varepsilon})$ muni de la taille $t_2(p)$, le problème $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$ paramétré par (p, a, b, P) dans I_ε se réduit au problème paramétré $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$.*

Preuve. Soit (p, a, b, P) dans I_ε, commençons par donner la réduction $(\mathsf{A}^2, \mathsf{B}^2)$ de $\mathcal{L}og\left(E_{a,b}(\mathbb{F}_p), \overline{P}\right)$ vers $\mathcal{L}og\left(E_{a,b}(\mathbb{F}_p)[\varepsilon], P\right)$. Les algorithmes A^2 et B^2 sont numérotés algorithmes 12 et 13 respectivement.

Algorithme 12 : A^2

Entrées : (p, a, b, P) *dans* I_ε
$Q = [x_0, y_0] \in \widehat{\mathsf{EC}_{p,a,b}}$

Sortie : $Q' \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$ *tel que* $\overline{Q'} = Q$.

1. ***If*** $x_0 = \infty$
2. ***then*** $Q' := [\infty, 0]$
3. ***else if*** $y_0 = 0$
4. ***then*** $Q' := \left[\left(x_0, -\dfrac{a_1 x_0 + b_1}{3x_0^2 + a_0}\right), (y_0, 0)\right]$
5. ***else*** $Q' := \left[(x_0, 0), \left(y_0, \dfrac{a_1 x_0 + b_1}{2y_0}\right)\right]$
6. ***endif***
7. ***endif***
8. *RETURN* (Q')

Pour une entrée $((p, a, b, P), Q)$, l'algorithme A^2 renvoie un relevé de Q dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$. En effet, y_0 et $3x_0^2 + a_0$ ne peuvent s'annuler en même temps, donc Q' est bien défini. De plus, les différentes sorties possibles Q' vérifient toutes l'équation définissant $E_{a,b}$.

Algorithme 13 : B^2

Entrées : $(p, a, b, P) \in I_\varepsilon$
$n \in \mathbb{Z}$

Sortie : $n' \in \mathbb{Z}$

1. $\mathbf{N} := \mathsf{Card}(\mathsf{E}_{a,b}(\mathbb{F}_p))$
2. $n' := n \mod \mathbf{N}$
3. *RETURN* (n')

Pour une entrée $((p, a, b, P), n)$, l'algorithme B^2 renvoie le reste de la division Euclidienne de n par $Card\,(E_{a,b}(\mathbb{F}_p))$.
Le couple d'algorithmes $(\mathsf{A}^2, \mathsf{B}^2)$ est bien une réduction du problème paramétré $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$.
En effet, le lemme 3.3.3, nous assure que pour Q dans $E_{a,b}(\mathbb{F}_p)$, on a

$$\log_{\overline{P}}(Q) = \mathsf{B}^2\left((p, a, b, P), (\log_P(\mathsf{A}^2\left((p, a, b, P), Q\right)))\right).$$

Evaluons, enfin, la complexité arithmétique de cette réduction, en négligeant le temps de calcul de l'addition (cf. page 68). L'algorithme A^2 nécessite dans le pire des cas une inversion, une multiplication et un calcul de carré dans $\mathbb{F}_p$. L'algorithme B^2 a un temps d'exécution polynomial en $\log(p)$ car l'algorithme de calcul de cardinal a une complexité en $O(\log^{4+\epsilon}(p))$ avec $\epsilon < 1$ (cf. table 3.5).
La réduction $\left(\mathsf{A}^2, \mathsf{B}^2\right)$ est donc polynomiale en $t_2(p)$ et résout le problème du logarithme discret dans $(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ à partir d'une solution au problème du logarithme discret dans $(E_{a,b}(\mathbb{F}_p), \overline{P})$. Le résultat suit. □

Illustrons le fonctionnement de cette seconde réduction par un exemple :

Exemple 3.3.2 *Considérons encore une fois*

$$(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$$

dans I_ε.
Soit $Q = (340, 315)$ *dans* $E_{333,394}(\mathbb{F}_{701})$. *Calculons le logarithme discret de* Q *en base* $(42, 172)$ *avec l'aide d'un algorithme résolvant le problème du logarithme discret dans* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$.

L'algorithme A^2 *calcule :*
5. $Q' := \left(340, 315 + (623 \times 340 + 34)(2 \times 315)^{-1}\varepsilon\right) \mod 701 = (340, 315 + 610\varepsilon)$.
L'algorithme A^2 *renvoie* $(340, 315 + 610\varepsilon)$, *dont le logarithme discret en base* P *dans* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$ *est* 393916.

Puis l'algorithme B^2 *prend en entrée* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *et* $n = 393916$. *Son exécution donne :*
1. $\mathbf{N} := 733$
2. $n := 393916 \mod 733$
3. *RETURN* (295).

L'algorithme B^2 *renvoie* 295, *dont nous pouvons vérifier qu'il est bien le logarithme discret de* Q *en base* $\overline{P}$ *par l'égalité :*

$$295(42, 172) = (340, 315).$$

Les réductions des lemmes 3.3.2 et 3.3.4 nous donnent, ainsi, l'équivalence des problèmes de logarithmes discrets recherchée :

Corollaire 3.3.1 *Soit* (p, a, b, P) *tels que :*
– p *est un nombre premier,*
– a *et* b *sont dans* $\mathbb{F}_p[\varepsilon]$,
– $E_{a,b}(\mathbb{F}_p)$ *est de cardinal premier disctinct de* p.
les problèmes $\mathcal{L}og(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ *et* $\mathcal{L}og(E_{a,b}(\mathbb{F}_p), \overline{P})$ *paramétrés par* (p, a, b, P) *sont équivalents.*

Preuve. Ce résultat se déduit des lemmes 3.3.2 et 3.3.4. □

Cette équivalence a pour conséquence que :
– le problème du logarithme discret restera aussi difficile sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ que sur $E_{a,b}(\mathbb{F}_p)$. Ainsi, tant qu'on ne trouve pas d'attaque générique de ce problème sur $E_{a,b}(\mathbb{F}_p)$, le problème reste difficile à résoudre sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$;
– le problème du logarithme discret est aussi facile sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ que sur $E_{a,b}(\mathbb{F}_p)$. Ainsi, une méthode pour trouver une attaque de ce problème sur $E_{a,b}(\mathbb{F}_p)$ peut être d'en chercher une sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

Comparons maintenant les problèmes Calculatoires de Diffie Hellman (introduits dans la section 1.3.5) sur $E_{a,b}(\mathbb{F}_p)$ et $E_{a,b}(\mathbb{F}_p[\varepsilon])$. S'agissant là encore de problèmes calculatoires, comme dans le cas du problème du logarithme discret, on obtient une équivalence. Nous en précisons les réductions dans le paragraphe suivant :

Equivalence des problèmes CDH

Commençons par la réduction la moins évidente, celle de $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$.

Dans ce cas, la difficulté consiste à calculer le composé de Diffie-Hellman de base P dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ à partir du composé de Diffie-Hellman de base $\overline{P}$ dans $E_{a,b}(\mathbb{F}_p)$. Pour cela, nous allons utiliser l'isomorphisme Λ du lemme 2.2.6 pour établir une relation entre les composés de Diffie-Hellman dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$, $E_{a,b}(\mathbb{F}_p)$ et $\mathbb{F}_p$. Le composé de Diffie-Hellman dans $\mathbb{F}_p$ étant facile à calculer, nous montrerons ensuite que cette relation, nous permet de construire une réduction en temps polynomial en $t_2(p)$ de $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$.
Cette relation est analogue au théorème chinois des restes. Elle est établie dans le lemme 3.3.5, puis les considérations effectives sont étudiées dans le lemme 3.3.6.

Lemme 3.3.5 *Soit* (p, a, b, P) *dans* I_ε, *soit* A *et* B *dans* $E_{a,b}(\mathbb{F}_p[\varepsilon])$, *le composé de Diffie-Hellman de* A *et* B *en base* P *est :*

$$\lambda\left(CDH_{\overline{P}}(\overline{A}, \overline{B})\right) + \Theta\left(\mathbf{N}' \frac{\lceil \mathbf{N}A \rfloor \lceil \mathbf{N}B \rfloor}{\lceil \mathbf{N}P \rfloor}\right).$$

Preuve. Notons C le composé de Diffie-Hellman de A et B en base P.
D'après l'isomorphisme défini dans le lemme 2.2.6, on a :

$$\begin{aligned} C &= \Lambda^{-1}\left(\Lambda(C)\right) \\ &= \Lambda^{-1}\left(\overline{C}, \lceil \mathbf{N}\mathbf{N}'C \rfloor\right) \\ &= \lambda\left(\overline{C}\right) + \Theta\left(\lceil \mathbf{N}\mathbf{N}'C \rfloor\right) \\ &= \lambda\left(\overline{C}\right) + \Theta\left(\mathbf{N}'\lceil \mathbf{N}C \rfloor\right) \end{aligned} \tag{3.1}$$

De plus, pour α et β tels que $A = \alpha P$ et $B = \beta P$, on a : $C = \alpha\beta P$. Donc on trouve les relations suivantes :

$$\overline{C} = \alpha\beta\overline{P} \text{ avec } \overline{A} = \alpha\overline{P} \text{ et } \overline{B} = \beta\overline{P}$$
$$\text{et}$$
$$\lceil \mathbf{N}C \rfloor = \alpha\beta\lceil \mathbf{N}P \rfloor \text{ avec } \lceil \mathbf{N}A \rfloor = \alpha\lceil \mathbf{N}P \rfloor \text{ et } \lceil \mathbf{N}B \rfloor = \beta\lceil \mathbf{N}P \rfloor.$$

Ainsi, on a :

$$\overline{C} = CDH_{\overline{P}}(\overline{A}, \overline{B}) \tag{3.2}$$
$$\text{et}$$
$$\begin{aligned} \lceil \mathbf{N}P \rfloor\lceil \mathbf{N}C \rfloor &= \alpha\lceil \mathbf{N}P \rfloor\beta\lceil \mathbf{N}P \rfloor \\ &= \lceil \mathbf{N}A \rfloor\lceil \mathbf{N}B \rfloor. \end{aligned} \tag{3.3}$$

En utilisant les relations 3.1, 3.2 et 3.3, on obtient l'égalité recherchée. □

Le lemme 3.3.5 nous permet de calculer facilement le composé de Diffie-Hellman de A et B en base P à partir du composé de Diffie-Hellman de $\overline{A}$ et $\overline{B}$ en base $\overline{P}$. En effet, toutes les expressions du type $\lceil \mathbf{N}\mathbf{N}'P \rfloor$ sont faciles à calculer.
L'application du lemme 3.3.5 nous donne alors la réduction énoncée au lemme 3.3.6

Lemme 3.3.6 *Soit un ensemble effectif infini $(I_\varepsilon, \phi_{I_\varepsilon})$ muni de la taille $t_2(p)$, le problème $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ paramétré par (p, a, b, P) dans I_ε se réduit au problème paramétré $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$.*

Preuve. Soit (p, a, b, P) dans I_ε, commençons par donner la réduction $(\mathsf{A}^3, \mathsf{B}^3)$ de $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$.
Cette réduction est du même type que la réduction $\left(\mathsf{A}^1, \mathsf{B}^1\right)$. En effet, le premier algorithme renvoie les projections des entrées A et B et le second calcule le composé de Diffie-Hellman de A et B à partir de la relation 3.3.5 enoncée dans le lemme 3.3.5.
Les algorithmes A^3 et B^3 sont numérotés algorithmes 14 et 15 respectivement.

Algorithme 14 : A^3

Entrées : (p, a, b, P) *dans* I_ε
$A \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$B \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$

Sortie : $\overline{Q} \in \widehat{\mathsf{EC}_{p,a,b}}$

1. $\overline{A} := \mathsf{A}^1(p, a, b, P, A)$
1. $\overline{B} := \mathsf{A}^1(p, a, b, P, B)$
2. *RETURN* $(\overline{A}, \overline{B})$

Décompte des opérations	I	M	Total Complexité arithmétique
1 : Card			$O(\log^{4+\epsilon}(p))$
2 : **I**	1	0	$O(1)$
3, *4*, *5* : Mult			$O(\log(\mathbf{N})) = O(\log(p))$
6 : **I M M**	1	2	$O(1)$
7 : $\mathsf{Lambda}^{-1}_{p,a,b}$			$O(\log(\mathbf{N})) = O(\log^{4+\epsilon}(p))$

TAB. 3.6 – Calcul de complexité arithmétique de B^3.

Pour une entrée $((p,a,b,P),A,B)$, l'algorithme A^3 renvoie $\overline{A}$ et $\overline{B}$ dans $E_{a,b}(\mathbb{F}_p)$.

Algorithme 15 : B^3

Entrées : (p,a,b,P) *dans* I_ε
$A \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$B \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$C \in \widehat{\mathsf{EC}_{p,a,b}}$

Sortie : $C' \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$

1. $\mathbf{N} := \mathsf{Card}(\mathsf{E}_{a,b}(\mathbb{F}_p))$
2. $\mathbf{N}' := \mathbf{N}^{-1} \mod p$
3. $k_A := \lceil \mathbf{N}A \rfloor$
4. $k_B := \lceil \mathbf{N}B \rfloor$
5. $k_P := \lceil \mathbf{N}P \rfloor$
6. $k := \mathbf{N}' k_A \times k_B / k_P \mod p$
7. $C' := \mathsf{Lambda}^{-1}_{p,a,b}(C,k)$
8. *RETURN* (C')

Pour une entrée $((p,a,b,P),A,B,C)$, l'algorithme B^3 renvoie $\lambda(C) + \Theta(k)$ où k vaut $\mathbf{N}' \dfrac{\lceil \mathbf{N}A \rfloor \lceil \mathbf{N}B \rfloor}{\lceil \mathbf{N}P \rfloor}$ dans $\mathbb{F}_p$.

Vérifions que ce couple d'algorithmes est bien une réduction des problèmes paramétrés $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]),P)$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p),\overline{P})$.
Pour cela, le lemme 3.3.5, nous assure que pour Q dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$, on a

$$CDH_P(A,B) = \mathsf{B}^3\left((a,b,p,P), CDH_{\overline{P}}\left(\mathsf{A}^3\left((a,b,p,P),A,B\right)\right)\right).$$

Evaluons ensuite le temps de calcul arithmétique de cette réduction. L'algorithme A^3 a la même complexité que l'algorithme A_1 et ne nécessite aucune opération dans $\mathbb{F}_p$ et l'algorithme B^3 a un temps d'exécution polynomial en $\log(p)$ d'après le décompte des opérations présenté dans la table 3.6 (**I** et **M** y représentent les opérations d'inversion et de multiplication dans $\mathbb{F}_p$ i.e. modulo p). Les complexités arithmétiques des algorithmes Card, Mult et $\mathsf{Lambda}^{-1}_{p,a,b}$ proviennent des résultats exposés pages 64 et 71.
La réduction $(\mathsf{A}^3,\mathsf{B}^3)$ est donc polynomiale en $t_2(p)$ et résout le problème calculatoire de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p[\varepsilon]),P)$ à partir d'une solution du problème calculatoire de Diffie-

Hellman dans $(E_{a,b}(\mathbb{F}_p), \overline{P})$. Le résultat suit. □

Illustrons le fonctionnement de cette réduction par un exemple :

Exemple 3.3.3 *Reprenons l'exemple de la courbe d'équation*

$$y^2 = x^3 + (333 + 623\varepsilon)x + 394 + 34\varepsilon$$

dans $\mathbb{F}_{701}$. *Le groupe* $E_{333,394}(\mathbb{F}_{701})$ *est toujours d'ordre* 733.
L'élément $P = (42 + 137\varepsilon, 172 + 464\varepsilon)$ *est un générateur du groupe* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$.

Considérons donc $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *dans* I_ε.
Soit $A = 385686P = (515 + 687\varepsilon, 276 + 88\varepsilon)$ *et* $B = 449730P = (247 + 269\varepsilon, 126 + 319\varepsilon)$.
Calculons le composé de Diffie-Hellman de A *et* B *en base* P *sachant que dans* $E_{333,394}(\mathbb{F}_{701})$, *le composé de Diffie-Hellman de* $\overline{A}$ *et* $\overline{B}$ *en base* $\overline{P}$ *est*

$$C = 385686 \times 449730(42, 172) = (398, 60).$$

L'algorithme A^3 *renvoie :* $\overline{A} = (515, 276)$ *et* $\overline{B} = (247, 126)$.

Puis l'algorithme B^3 *prend en entrée* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$, $A = (515 + 687\varepsilon, 276 + 88\varepsilon)$, $B = (247 + 269\varepsilon, 126 + 319\varepsilon)$ *et* $C = (398, 60)$.
Son exécution donne :

1. $\mathbf{N} := 733$
2. $\mathbf{N}' := 733^{-1} \mod 701 = 241$
3. $k_A := \lceil 733(515 + 687\varepsilon, 276 + 88\varepsilon) \rfloor = \lceil \mathcal{O}_{92} \rfloor = 92$
4. $k_B := \lceil 733(247 + 269\varepsilon, 126 + 319\varepsilon) \rfloor = \lceil \mathcal{O}_{325} \rfloor = 325$
5. $k_P := \lceil 733(42 + 137\varepsilon, 172 + 464\varepsilon) \rfloor = \lceil \mathcal{O}_{145} \rfloor = 145$
6. $k := 241 \times 92 \times 325/145 \mod 701 = 505$
7. $C' := \mathsf{Lambda}((398, 60), 505) = (398 + 661\varepsilon, 60 + 412\varepsilon)$
8. *RETURN* $(398 + 661\varepsilon, 60 + 412\varepsilon)$.

L'algorithme B^3 *renvoie* $(398 + 661\varepsilon, 60 + 412\varepsilon)$, *dont nous pouvons vérifier qu'il est bien le composé de Diffie-Hellman de* A *et* B *en base* P *par l'égalité :*

$$385686 \times 449730(42 + 137\varepsilon, 172 + 464\varepsilon) = (398 + 661\varepsilon, 60 + 412\varepsilon).$$

Etablissons maintenant la réduction inverse, à savoir de $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$.

Dans ce cas, la difficulté consiste à reconstituer une entrée du problème CDH dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ à partir d'une entrée du problème CDH dans $E_{a,b}(\mathbb{F}_p)$. Pour cela, le lemme 3.3.7 nous indique que pour une entrée A, B dans $E_{a,b}(\mathbb{F}_p)$ nous pouvons choisir comme entrée dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ deux relevés quelconques A' et B' de A et B.
Cette propriété est établie dans le lemme 3.3.7, puis les considérations effectives, qui en découlent, sont étudiées dans le lemme 3.3.8.

Lemme 3.3.7 *Soit* (p, a, b, P) *dans* I_ε, *soit* A *et* B *dans* $E_{a,b}(\mathbb{F}_p)$, *on a :*
- $\overline{P}$ *est un générateur de* $E_{a,b}(\mathbb{F}_p)$;
- *le composé de Diffie-Hellman de* A *et* B *en base* $\overline{P}$ *est la projection de* $CDH_P(A', B')$ *dans* $E_{a,b}(\mathbb{F}_p[\varepsilon])$; *où* A' *et* B' *sont des relevés quelconques de* A *et* B *respectivement dans* $E_{a,b}(\mathbb{F}_p[\varepsilon])$. *Soit :*

$$CDH_{\overline{P}}(A,B) = \overline{CDH_P(A',B')} \text{ avec } \overline{A'} = A \text{ et } \overline{B'} = B.$$

Preuve. Soit A' et B' deux relevés de A et B. Ils vérifient $\overline{A'} = A$ et $\overline{B'} = B$. Posons $C' = CDH_P(A',B')$. On a $C' = \alpha\beta P$ où $A' = \alpha P$ et $B' = \beta P$. Par projection, $\overline{C'} = \alpha\beta\overline{P}$ avec $\overline{A'} = \alpha\overline{P}$ et $\overline{B'} = \beta\overline{P}$.
Donc, on a bien :

$$\overline{C'} = \alpha\beta\overline{P} \text{ avec } A = \alpha\overline{P} \text{ et } B = \beta\overline{P}.$$

Ce qui prouve le lemme. □

L'application du lemme 3.3.7 nous donne la réduction énoncée au lemme 3.3.8.

Lemme 3.3.8 *Soit un ensemble effectif infini $(I_\varepsilon, \phi_{I_\varepsilon})$ muni de la taille $t_2(p)$, le problème $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$ paramétré par (p,a,b,P) dans I_ε se réduit au problème paramétré $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$.*

Preuve. Soit (p,a,b,P) dans I_ε, commençons par donner la réduction $(\mathsf{A}^4, \mathsf{B}^4)$ de $\mathcal{C}_{DH}\left(E_{a,b}(\mathbb{F}_p), \overline{P}\right)$ vers $\mathcal{C}_{DH}\left(E_{a,b}(\mathbb{F}_p)[\varepsilon], P\right)$. Cette réduction est du même type que la réduction $(\mathsf{A}^2, \mathsf{B}^2)$. En effet, la première consiste à calculer des relevés de A et B et la seconde à calculer le reste de la division euclidienne du résultat par $\mathbf{N}$.
On trouve donc pour les algorithmes A^4 et B^4, numérotés algorithmes 16 et 17 respectivement.

Algorithme 16 : A^4

Entrées : (a,b,p,P) *dans* I_ε
$A \in \widehat{\mathsf{EC}_{p,a,b}}$
$B \in \widehat{\mathsf{EC}_{p,a,b}}$

Sortie : $A' \in \widehat{\mathsf{EC}^\varepsilon_{p,a,b}}$ *tel que* $\overline{A'} = A$.
$B' \in \widehat{\mathsf{EC}^\varepsilon_{p,a,b}}$ *tel que* $\overline{B'} = B$.

1. $A' = \mathsf{A}^2((a,b,p,P), A)$
2. $B' = \mathsf{A}^2((a,b,p,P), B)$
3. *RETURN* (A', B')

Pour une entrée (A,B) l'algorithme A^4 renvoie deux relevés A' et B' de A et B respectivement.

Algorithme 17 : B^4

Entrées : $(p,a,b,P) \in I_\varepsilon$
$C' \in \widehat{\mathsf{EC}^\varepsilon_{p,a,b}}$

Sortie : $C \in \widehat{\mathsf{EC}_{p,a,b}}$

1. $C = \mathsf{A}^1((p,a,b,P), C')$
2. *RETURN* (C)

L'algorithme B^4 est identique à l'algorithme A^1 ; il renvoie la projection sur $E_{a,b}(\mathbb{F}_p)$ de l'élément C'.

Le couple d'algorithmes $(\mathsf{A}^4, \mathsf{B}^4)$ est bien une réduction des problèmes paramétrés $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ vers $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$.
En effet, le lemme 3.3.7, nous assure que pour (A, B) dans $E_{a,b}(\mathbb{F}_p)$, on a

$$CDH_{\overline{P}}(A, B) = \mathsf{B}^4\left((a, b, p, P), (CDH_P(\mathsf{A}^4\left((a, b, p, P), A, B\right)))\right).$$

Comme la réduction $\left(\mathsf{A}^2, \mathsf{B}^2\right)$, $\left(\mathsf{A}^4, \mathsf{B}^4\right)$ est polynomiale en $t_2(p)$ (cf. page 76). De plus, elle résout le problème calculatoire de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ à partir d'une solution au problème calculatoire de Diffie-Hellman dans $(E_{a,b}(\mathbb{F}_p), \overline{P})$. Le résultat suit. □

Illustrons le fonctionnement de cette seconde réduction par un exemple :

Exemple 3.3.4 *Soit* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *dans* I_ε.
Donc, le point $(42, 172)$ *est générateur de* $E_{333,394}(\mathbb{F}_{701})$. *Soit* $A = (453, 36)$ *et* $B = (294, 611)$ *dans* $E_{333,394}(\mathbb{F}_{701})$. *Calculons le composé de Diffie-Hellman de* A *et* B *en base* $(42, 172)$ *à l'aide de la réduction* $(\mathsf{A}^4, \mathsf{B}^4)$.

L'algorithme A^4 *calcule :*
1. $A' := \left(453 - (623 \times 453 + 34)(3 \times 453^2 + 333)^{-1}\varepsilon, 36\right) \mod 701 = (453 + 495\varepsilon, 36)$.
2. $B' := \left(294 - (623 \times 294 + 34)(3 \times 294^2 + 333)^{-1}\varepsilon, 611\right) \mod 701 = (294 + 25\varepsilon, 611)$.
L'algorithme A^4 *renvoie* $\left((453 + 495\varepsilon, 611), (294 + 25\varepsilon, 36)\right)$.
Le composé CDH de ce couple en base P *dans* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$ *est* $C' = (247 + 175\varepsilon, 126 + 287\varepsilon)$ *car :*
– $A' = 237108(42 + 137\varepsilon, 172 + 464\varepsilon)$,
– $B' = 63413(42 + 137\varepsilon, 172 + 464\varepsilon)$,
– $C' = 237108 \times 63413(42 + 137\varepsilon, 172 + 464\varepsilon)$.

Puis l'algorithme B^4 *prend en entrée* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *et* $C' = (247 + 175\varepsilon, 126 + 287\varepsilon)$.
Son exécution donne :
1. $C := (247, 126)$
L'algorithme B^4 *renvoie* $(247, 126)$, *dont nous pouvons vérifier qu'il est bien le composé de Diffie-Hellman de* A *et* B *en base* $\overline{P}$ *par les égalités :*
– $349(42, 172) = (453, 36) = A$,
– $375(42, 172) = (294, 611) = B$,
– $349 \times 375(42, 172) = 130875(42, 172) = (247, 126) = C$

Les réductions des lemmes 3.3.6 et 3.3.8 nous donnent, ainsi, l'équivalence des problèmes CDH recherchée :

Corollaire 3.3.2 *Soit* (p, a, b, P) *tels que :*
– p *est un nombre premier ;*
– a *et* b *sont dans* $\mathbb{F}_p[\varepsilon]$ *;*
– $E_{a,b}(\mathbb{F}_p)$ *est de cardinal premier disctinct de* p.
Les problèmes $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ *et* $\mathcal{C}_{DH}(E_{a,b}(\mathbb{F}_p), \overline{P})$ *paramétrés par* (p, a, b, P) *sont équivalentes.*

Preuve. Ce résultat se déduit des lemmes 3.3.6 et 3.3.8. □

Cette équivalence a pour conséquence que :

– le problème CDH restera aussi difficile sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ que sur $E_{a,b}(\mathbb{F}_p)$.
Ainsi, tant qu'on ne trouve pas d'attaque générique de ce problème sur $E_{a,b}(\mathbb{F}_p)$, le problème reste difficile à résoudre sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$;
– le problème CDH est aussi facile sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ que sur $E_{a,b}(\mathbb{F}_p)$.
Ainsi, une méthode pour trouver une attaque de ce problème sur $E_{a,b}(\mathbb{F}_p)$ peut être d'en chercher une sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

Dans la prochaine section, nous étudions le problème Décisionnel de Diffie-Hellman (introduit dans la section 1.3.5) sur $E_{a,b}(\mathbb{F}_p)$ et nous montrons que celui-ci est facile à résoudre.

3.3.2 Problèmes décisionnels : DDH

Cette section aborde le problème décisionnel de Diffie-Hellman sur les groupes $E_{a,b}(\mathbb{F}_q[\varepsilon])$ vérifiant les conditions 1. et 2. énoncées page 72 ; à savoir :

1. $\mathbf{N}$ le cardinal de $E_{\pi(a),\pi(b)}(\mathbb{F}_q)$ est un nombre premier différent de p ;
2. $q = p$, soit q est un nombre premier.

Nous allons montrer que ce problème est facile à résoudre.

Ce résultat paraît naturel dans la mesure où nous savons :
– qu'il existe un isomorphisme entre $E_{a,b}(\mathbb{F}_p[\varepsilon])$ et $E_{a,b}(\mathbb{F}_p) \times \mathbb{F}_p$ calculable en temps polynomial (cf. section 3.2) ;
– que le problème Décisionnel de Diffie-Hellman est trivial sur $(\mathbb{F}_p, +)$.
– que la nature décisonnelle de ce problème permet de conclure à une information sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$ à partir d'une information sur $\mathbb{F}_p$.

Nous allons construire un distingueur résolvant le problème $D_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ paramétré par (p, a, b, P) dans I_ε, puis nous allons montrer que son avantage est non négligeable.
Pour commencer, établissons la relation qui nous permettra de résoudre le problème DDH sur $E_{a,b}(\mathbb{F}_p[\varepsilon])$. Elle figure dans la preuve du lemme 3.3.5. Nous précisons cette relation dans le corollaire 3.3.3.

Corollaire 3.3.3 *Soit (p, a, b, P) dans I_ε, soit A, B et C dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$. Si $A = \alpha P$, $B = \beta P$ et $C = \alpha\beta P$, alors ces éléments vérifient :*

$$\lceil \mathbf{N}P \rfloor \lceil \mathbf{N}C \rfloor \quad = \quad \lceil \mathbf{N}A \rfloor \lceil \mathbf{N}B \rfloor \tag{3.4}$$

Preuve.
L'élément C est le composé de Diffie-Hellman de A et B en base P. Le résultat provient alors de la relation 3.3 établie dans la preuve du lemme 3.3.5. □

Nous pouvons maintenant considérer les aspects effectifs de ce problème.

Commençons par construire un distingueur du problème D_{DH} utilisant le corollaire 3.3.3.
L'idée est la suivante : si des éléments P, A, B et C choisis au hasard dans un groupe $E_{a,b}(\mathbb{F}_p[\varepsilon])$ vérifient la relation 3.4, alors il est plus probable que C soit le composé de Diffie-Hellman de A et B en base P que s'il ne la vérifie pas.
On obtient ainsi le distingueur $\mathcal{D}_{DDH}$, défini par l'algorithme 18, dont nous calculons l'avantage en fonction des paramètres (p, a, b, P) et dont nous montrons qu'il s'exécute en temps polynomial (en $t_2(p)$).

Algorithme 18 : $\mathcal{D}_{DDH}$

Entrées : $(p, a, b, P) \in I_\varepsilon$
$P \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$A \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$B \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$
$C \in \widehat{\mathsf{EC}^{\varepsilon}_{p,a,b}}$

Sortie : $\delta \in \{0, 1\}$

1. $\mathbf{N} := \mathsf{Card}(\mathsf{E}_{a,b}(\mathbb{F}_p))$
2. $\mathbf{N}' := \mathbf{N}^{-1} \mod p$
3. $k_P := \lceil \mathbf{N}P \rfloor$
4. $k_A := \lceil \mathbf{N}A \rfloor$
5. $k_B := \lceil \mathbf{N}B \rfloor$
6. $k_C := \lceil \mathbf{N}C \rfloor$
7. $k := k_A \times k_B \mod p$
8. $k' := k_C \times k_P \mod p$
9. ***If*** $k = k' \mod p$
10. ***then*** $\delta := 1$
11. ***else*** $\delta := 0$
12. ***endif***
13. *RETURN* (δ)

Proposition 3.3.1 *Soit (p_0, a_0, b_0, P_0) dans I_ε. Soit $\mathcal{D}_{DDH}$ le distingueur défini par l'algorithme 18. Notons :*

$$\mathsf{Adv}^{DDH(p_0,a_0,b_0,P_0)}_{\mathcal{D}_{DDH}} = \left| Pr\left(\mathcal{D}_{DDH}(x) = 1 : x \xleftarrow{u} Rand(p_0, a_0, b_0, P_0)\right) - Pr\left(\mathcal{D}_{DDH}(x) = 1 : x \xleftarrow{u} DH(p_0, a_0, b_0, P_0)\right)\right|$$

On a :

$$\mathsf{Adv}^{DDH(p_0,a_0,b_0,P_0)}_{\mathcal{D}_{DDH}} = 1 - \frac{1}{p_0}.$$

Preuve.
Tout d'abord, il est évident que :

$$\Pr\left(\mathcal{D}_{DDH}(x) = 1 : x \xleftarrow{u} \mathrm{DH}(p_0, a_0, b_0, P_0)\right) \quad = \quad 1. \tag{3.5}$$

En effet, si $x = (p_0, a_0, b_0, P_0, A, B, C)$ est dans $\mathrm{DH}(p_0, a_0, b_0, P_0)$, alors A, B et C vérifient la relation 3.4 et le distingueur $\mathcal{D}_{DDH}$ renvoie 1 (étape *10*).

Calculons, maintenant, la probabilité que le distingueur renvoie 1 lorsque $x = (p_0, a_0, b_0, P_0, A, B, C)$ est choisi uniformément dans $\mathrm{Rand}(p_0, a_0, b_0, P_0)$. On trouve :

$$\begin{aligned} & Pr\left(\mathcal{D}_{DDH}(x) = 1 : x \xleftarrow{u} \mathrm{Rand}(p_0, a_0, b_0, P_0)\right) \qquad (3.6) \\ = \quad & Pr\left(\lceil \mathbf{N}P_0 \rfloor \lceil \mathbf{N}C \rfloor = \lceil \mathbf{N}A \rfloor \lceil \mathbf{N}B \rfloor : (p_0, a_0, b_0, P_0, A, B, C) \xleftarrow{u} \mathrm{Rand}(p_0, a_0, b_0, P_0)\right) \end{aligned}$$

Les éléments A, B et C sont choisis uniformément dans $E_{a_0,b_0}(\mathbb{F}_{p_0})$. L'isomorphisme Λ défini dans le lemme 2.2.6 nous assure qu'alors les variables aléatoires $\lceil \mathbf{N}A \rfloor$, $\lceil \mathbf{N}B \rfloor$ et $\lceil \mathbf{N}C \rfloor$ sont distribuées uniformément dans $[\![0, p-1]\!]$.
De plus, la valeur non nulle $\lceil \mathbf{N}P_0 \rfloor$ dépend uniquement de (p_0, a_0, b_0, P_0) et la relation 3.6 devient :

$$\begin{aligned} & Pr\left(\mathcal{D}_{DDH}(x) = 1 : x \xleftarrow{u} \mathrm{Rand}(p_0, a_0, b_0, P_0)\right) \\ = \;& Pr\left(z = xy : (x, y, z) \xleftarrow{u} [\![0, p_0 - 1]\!]^3\right) \end{aligned} \tag{3.7}$$

Considérons donc l'expérience aléatoire consistant à choisir uniformément trois entiers x, y et z entre 0 et $p_0 - 1$, on trouve :

$$\begin{aligned} Pr\left(z = xy\right) &= \sum_{i=0}^{p_0-1} Pr\left(xy = z | z = i\right) \times Pr(z = i) \\ &= \sum_{i=0}^{p_0-1} \left(Pr\left(xy = i\right) \times \frac{1}{p_0}\right) = \frac{1}{p_0} \times \sum_{i=0}^{p_0-1} Pr\left(xy = i\right) \\ &= \frac{1}{p_0} \times 1 = \frac{1}{p_0}. \end{aligned} \tag{3.8}$$

Ainsi, d'après les relations 3.5, 3.7 et 3.8, on obtient la relation souhaitée :

$$\mathsf{Adv}^{DDH(p_0,a_0,b_0,P_0)}_{\mathcal{D}_{DDH}} = 1 - \frac{1}{p_0}. \qquad \square$$

Illustrons le fonctionnement et l'efficacité de ce distingueur sur un exemple :

Exemple 3.3.5 *Soit* $(701, 333 + 623\varepsilon, 394 + 34\varepsilon, (42 + 137\varepsilon, 172 + 464\varepsilon))$ *dans* I_ε.
Notons $P = (42 + 137\varepsilon, 172 + 464\varepsilon)$. *On choisit* $A = 177242P = (443 + 49\varepsilon, 17 + 279\varepsilon)$ *et* $B = 461254P = (691 + 576\varepsilon, 565 + 74\varepsilon)$.
Le distingueur $\mathcal{D}_{DH}$ *calcule :*
1. $\mathbf{N} = 733$
2. $\mathbf{N}' = 733^{-1} \mod 701 = 241$
3. $k_P = \lceil 733(42 + 137\varepsilon, 172 + 464\varepsilon) \rfloor = \lceil \mathcal{O}_{145} \rfloor = 145$
4. $k_A = \lceil 733(443 + 49\varepsilon, 17 + 279\varepsilon) \rfloor = \lceil \mathcal{O}_{28} \rfloor = 28$
5. $k_B = \lceil 733(691 + 576\varepsilon, 565 + 74\varepsilon) \rfloor = \lceil \mathcal{O}_{121} \rfloor = 121$
6. $k_C = \lceil \mathbf{N}C \rfloor$
7. $k = 28 \times 121 \mod 701 = 584$
8. $k' = 145 k_C \mod 701$

Or $584/145 \mod 701 = 589$. *Donc, on a :*
– si $k_C = 589$, *le distingueur renvoie* 1 *;*
– sinon , le distingueur renvoie 0.
De plus, le cardinal de $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701})$ *est premier distinct de* 701, *et d'après le corollaire 2.2.2 les groupes* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$ *et* $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}) \times \mathbb{F}_{701}$ *sont isomorphes. On remarque également, avec les notations du lemme 2.2.6 que* $\Lambda(C) = \left(\overline{C}, \mathbf{N}'k_C\right)$. *Or on calcule* $\mathbf{N}' = 733^{-1} \mod 701 = 241$ *et* $\mathbf{N}'k_C = 241 \times 589 \mod 701 = 347$.
L'isomorphisme Λ *nous assure qu'il existe exactement* $\mathbf{N}$ *éléments* C *parmi les* $\mathbf{N}p$ *éléments*

de $E_{333+623\varepsilon,394+34\varepsilon}(\mathbb{F}_{701}[\varepsilon])$ *tels que* $\lceil \mathbf{NN}'C \rfloor = 347$, *(soit* 733 *sur* 513833*).*

Donc le distingueur $\mathcal{D}_{DDH}$ *renvoie* 1 *dans environ* 0.14% *des cas et son avantage pour cette courbe est d'environ :*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{D}_{DDH}}^{DDH(701,333+623\varepsilon,394+34\varepsilon,(42+137\varepsilon,172+464\varepsilon))} &\simeq 1 - 0.0014 \\ &\simeq 0.9986.\end{aligned}$$

Nous en déduisons que le problème DDH est facile à résoudre sur les courbes elliptiques sur des anneaux de nombres duaux dans le corollaire 3.3.4.

Corollaire 3.3.4 *Soit* (p, a, b, P) *tels que :*
– p *est un nombre premier ;*
– a *et* b *sont dans* $\mathbb{F}_p[\varepsilon]$ *;*
– $E_{a,b}(\mathbb{F}_p)$ *est de cardinal premier disctinct de* p.
Le problème $D_{DH}(E_{a,b}(\mathbb{F}_p[\varepsilon]), P)$ *paramétré par* (p, a, b, P) *est facile à résoudre.*

Preuve. Nous devons montrer qu'il existe un distingueur de ce problème s'exécutant en temps polynomial (en $t_2(p)$) dont l'avantage est non négligeable.
La proposition 3.3.1 donne ce distingueur et calcule son avantage. Il reste alors à évaluer son temps de calcul.
Le décompte des opérations est effectué dans la table 3.7 et nous donne le résultat recherché.

			Total
Décompte des opérations	**I**	**M**	Complexité arithmétique
1 : Card			$O(\log^{4+\epsilon}(p))$
2 : **I**	1	0	$O(1)$
3, *4*, *5*, *6* : Mult			$O(\log(\mathbf{N})) = O(\log(p))$
7, *8* : **M**, **M**	0	2	$O(1)$

TAB. 3.7 – Calcul de complexité arithmétique de $\mathcal{D}_{DDH}$.

□

Ce résultat peut s'étendre à toutes les courbes elliptiques définies sur des anneaux $\mathbb{F}_p[\varepsilon]$, car :
– si le cardinal de $E_{a,b}(\mathbb{F}_p)$ n'est pas premier mais reste premier avec p, le même distingueur $\mathcal{D}_{DDH}$ s'exécute toujours et garde les mêmes propriétés ;
– si le cardinal de $E_{a,b}(\mathbb{F}_p)$ est divisible par le nombre premier p, nous allons voir dans la section suivante (partie 4.1) qu'alors le problème du logarithme discret est facile à résoudre. Donc, a fortiori le problème Décisionnel de Diffie Hellman l'est.

Chapitre 4

Applications cryptographiques de $E_{a,b}(\mathbb{F}_p[\varepsilon])$

Dans ce chapitre, nous utilisons les courbes elliptiques sur des nombres duaux à des fins cryptographiques.

L'étude de ces courbes nous a permis de développer une attaque du problème du logarithme discret sur des courbes elliptiques de cardinal p à coefficients non nuls. Celle-ci est exposée dans la section 4.1. De telles attaques existent déjà (cf. [AS98], [Sem98] et [Sma99]). Celle que nous exposons est particulièrement performante et facile à programmer.

Ensuite, nous avons utilisé ces courbes pour construire de nouveaux cryptosystèmes. Ceux-ci ressemblent aux cryptosystèmes de type El Gamal. Nous les avons exposés dans la section 4.2 puis nous en avons étudié certaines propriétés de sécurité dans la section 4.3. Nous obtenons ainsi un cryptosystème basé sur les courbes elliptiques sur des nombres duaux. Celui-ci est OW-CPA, sous l'hypothèse que le générateur de courbes elliptiques utilisé renvoie des courbes sur lesquelles le problème CDH est difficile ; mais il n'est malheureusement pas IND-CPA.

4.1 Résolution du problème du logarithme discret quand p divise N

Dans cette section, nous présentons une attaque du problème du logarithme discret sur les courbes elliptiques dans $\mathbb{F}_p$, à coefficients non nuls et de cardinal $\mathbf{N}$ égal à p.
Pour cela, nous utiliserons les outils développés dans la section 2.2.4. En particulier, la fonction μ y est définie quand p divise $\mathbf{N}$. Rappelons que cette fonction est un morphisme de $E_{a,b}(\mathbb{F}_p)$ dans $\mathbb{F}_p$. Cela va nous permettre de ramener le problème du logarithme discret de $(E_{a,b}(\mathbb{F}_p), +)$ dans $(\mathbb{F}_p, +)$ où ce dernier est trivial.

Comme dans la section 3.3, nous restreignons notre étude aux corps finis de cardinal premier différent de 2 et 3. La condition "p divise $\mathbf{N}$" se ramène alors, à une exception près, au cas où $\mathbf{N}$ égale p.
En effet, d'après le théorème de Hasse (cf. [Sil85, p.131] ou [Joy95, p.55]), $\mathbf{N}$ appartient à l'intervalle $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$. Il est alors facile de constater que, pour p strictement supérieur à 5, cet intervalle ne contient qu'un seul entier divisible par p : p. L'étude du cas $p = 5$ montre alors qu'il n'existe qu'une seule courbe elliptique dans $\mathbb{F}_5$ de cardinal divisible

par 5 mais différent de 5 : la courbe d'équation $y^2 = x^3 + 3x$ (cf Annexe B.2).

Ainsi, dans cette section uniquement, p est un nombre premier distinct de 2, 3 et 5 ; de plus a et b sont des éléments de $\mathbb{F}_p^*$ tels que le cardinal $\mathbf{N}$ de $E_{a,b}(\mathbb{F}_p)$ soit égal à p.
Nous supposerons vraie la conjecture suivante dont nous n'avons trouvé aucun contre-exemple parmi toutes les courbes concernées définies sur $\mathbb{F}_p$ avec $p < 211$ soit au total 8218 courbes.

Conjecture.

Soit p un nombre premier strictement supérieur à 5, soient $a' = a_0 + a_1\varepsilon$ et $b' = b_0 + b_1\varepsilon$ avec a_0, b_0 dans $\mathbb{F}_p^*$ et a_1, b_1 dans $\mathbb{F}_p$ tels que :
- $E_{a_0,b_0}(\mathbb{F}_p)$ soit de cardinal p,
- $3b_0a_1 - 2a_0b_1 \neq 0$ (soit j non constant)

le groupe $E_{a',b'}(\mathbb{F}_p[\varepsilon])$ est cyclique.

Elle est nécessaire à la preuve du corollaire 4.1.1.

Corollaire 4.1.1 *Considérons a et b non nuls dans $\mathbb{F}_p$. Soit P et Q deux éléments non neutres de $E_{a,b}(\mathbb{F}_p)$. Il existe a' et b' dans $\mathbb{F}_p[\varepsilon]$, tels que pour tout P' et Q' relevés de P et Q dans $E_{a',b'}(\mathbb{F}_p[\varepsilon])$, on ait :*

$$\lceil \mathbf{N}P' \rfloor \neq 0 \text{ et } \log_P(Q) \text{ est le plus petit entier compris entre } 1 \text{ et } p-1 \text{ représentant } \frac{\lceil \mathbf{N}Q' \rfloor}{\lceil \mathbf{N}P' \rfloor}.$$

Preuve. En supposant vraie la conjecture précédente, il existe a' et b' tels que $E_{a',b'}(\mathbb{F}_p[\varepsilon])$ soit cyclique.
Considérons alors P' et Q' relevés de P et Q dans $E_{a',b'}(\mathbb{F}_p[\varepsilon])$ et G un générateur de $E_{a',b'}(\mathbb{F}_p[\varepsilon])$. On a $P' = \alpha G$ avec α dans $[\![1, p^2 - 1]\!]$.
Supposons que l'on ait $\lceil pP' \rfloor = 0$, alors on a $p\alpha G = \Theta(0)$ et p^2 divise $p\alpha$, soit $\alpha = \alpha' p$. Ainsi on a $P = \overline{P'} = \alpha' p \overline{G} = O$ car $\mathbf{N} = p$. Cela est impossible car P est non neutre. Donc $\lceil \mathbf{N}P' \rfloor \neq 0$.
Tout élément non neutre d'un groupe d'ordre premier est générateur de ce groupe. Donc P engendre $E_{a,b}(\mathbb{F}_p)$ et $Q = \log_P(Q)P$ avec $\log_P(Q)$ dans $[\![1, p-1]\!]$. On obtient, avec les notations du lemme 2.2.10, $\mu(Q) = \log_P(Q)\mu(P)$. Ainsi, on a la relation dans $\mathbb{F}_p$: $\lceil \mathbf{N}Q' \rfloor = \overline{\log_P(Q)} \lceil \mathbf{N}P' \rfloor$. On en déduit le résultat. □

Nous utilisons le corollaire 4.1.1 pour élaborer une attaque du problème du logarithme discret sur les courbes elliptiques de cardinal premier p.
Nous procédons de la façon suivante. Soit $P = [x_0 : y_0 : 1]$ et $Q = [x_0' : y_0' : 1]$ deux points non neutres dans $E_{a,b}(\mathbb{F}_p)$:
- nous savons que la courbe $E_{a,b+\varepsilon}(\mathbb{F}_p[\varepsilon])$ est cyclique ;
- on commence par calculer un relevé du point P sur $E_{a,b+\varepsilon}(\mathbb{F}_p[\varepsilon])$: on choisit le point P' de coordonnées $[x_0 : y_0 + \dfrac{1}{2y_0}\varepsilon : 1]$.
- on calcule pP' ;
- si $pP' = 0$, alors la onjecture est fausse
- si $pP' \neq O$, alors la courbe $E_{a,b+\varepsilon}(\mathbb{F}_p[\varepsilon])$ est cyclique.
  - on choisit le relevé P' de P sur cette courbe ;

- on choisit un relevé de Q sur cette courbe en posant $Q' = [x'_0 : y'_0 + \dfrac{1}{2y'_0}\varepsilon : 1]$;

– les relevés choisis, on calcule pP' et pQ' ;
– on obtient deux entiers κ et κ' vérifiant $pP' = \Theta(\kappa)$ et $pQ' = \Theta(\kappa')$;
– d'après le corollaire 4.1.1, le logarithme discret recherché est $\dfrac{\kappa'}{\kappa}$.

L'algorithme $\mathsf{Attaque}_{p,a,b}$ (numéroté 19) accepte en entrée deux points P et Q de $E_{a,b}(\mathbb{F}_p)$ et résout le problème du logarithme discret sur les courbes elliptiques de cardinal premier p. Il procède de la façon décrite précédemment.

Algorithme 19 : $\mathsf{Attaque}_{p,a,b}$

Entrées : $P = [x_0, y_0] \in E_{a,b}(\mathbb{F}_p)$
$Q = [x'_0, y'_0] \in E_{a,b}(\mathbb{F}_p)$

Sortie : n *tel que* $Q = nP$.

1. $r = (2y_0)^{-1} \mod p$
2. $r' = (2y'_0)^{-1} \mod p$
3. $P' = [x_0, y_0 + r\varepsilon]$
4. $\kappa = \lceil pP' \rfloor$
5. ***If*** $\kappa = 0$
6. ***then*** *print ("La conjecture est fausse. Vérifier que a et b vérifient toutes les conditions.")*
7. ***else*** $Q' = [x'_0, y'_0 + r'\varepsilon]$
8. ***endif***
9. $\kappa' = \lceil pQ' \rfloor$
10. $n = \kappa'\kappa^{-1} \mod p$
11. *RETURN* n

Exemple.

Considérons la courbe $E_{340,93}(\mathbb{F}_{701})$, et les points $P = (68, 638)$ et $Q = (357, 101)$.

1. $r = (2 \times 638)^{-1} \mod 701 = 395$
2. $r = (2 \times 101)^{-1} \mod 701 = 59$
3. $P' = (68, 638 + 395\varepsilon)$ dans $E_{340,93+\varepsilon}(\mathbb{F}_{701})$
4. $\kappa = \lceil 701(68, 638 + 395\varepsilon) \rfloor = \lceil \Theta(332) \rfloor = 332$
5. On a $\kappa \neq 0$
7. donc $Q' = (357, 101 + 59\varepsilon)$ dans $E_{340,93+\varepsilon}(\mathbb{F}_{701})$
9. $\kappa' = \lceil 701(357, 101 + 59\varepsilon) \rfloor = \lceil \Theta(627) \rfloor = 627$
10. $n = 627 \times 332^{-1} \mod 701 = 4$

Nous pouvons vérifier l'égalité $Q = 4P$ dans $E_{340,93}(\mathbb{F}_{701})$. •

Nous avons testé cette attaque sur des courbes elliptiques de cardinal beaucoup plus élevé. Nous avons pu constater que cette attaque est d'une grande efficacité : à titre d'exemple, sur une machine DELL 1950, 2×QuadCore, Intel, Intel(R), Xeon(R), $CPUX5365$, de 3.00 GHz et 32 Gbytes de RAM, la programmation en langage maple de cette attaque sur la courbe elliptique $E_{a,b}(\mathbb{F}_p)$ avec :

$$
\begin{aligned}
a = \;& 70707039812766414181560797856636965403898166889753060003649679\\
& 41639949400884267937687235100621595108808434905850229200709399\\
& 52337451879850992275608452368868008053091512469257595093748569\\
& 95275443872094466424131065245485448636989865403033446188149954\\
& 09429645048255326401399009465463110510246710406342609099042 4;\\
b = \;& 72708182448976784394246480814843671971933020669651731513186934\\
& 49422212119777218917055741754412772328869050988091273423370986\\
& 30233794857582624132465295360439744130065800558010168539798057\\
& 78160597943946196605946095393942583975772597442741939948191933\\
& 92715389719432363941061245582410179675631051266899475394298 8;\\
p = \;& 89884656743115795386465259539451236680898848947115328636715040\\
& 57886633790275048156635423866120376801056005693993569667882939\\
& 48844072083112464237153197704763812124095106653136423951722828\\
& 31258170394006880666525104168199662300324380783573178055218927\\
& 49251364813701936156495438919539190011846643653345911095245 3;
\end{aligned}
$$

de taille 1024 (soit de l'ordre de $2^{1024} \simeq 10^{308}$), de générateur :

$$
\begin{aligned}
P = \;& (94352908529302867117733538774937838298447294218274281013117 10\\
& 89445835941858990781349636103293766939850765337818306056923311\\
& 64748655241254938837157952951013692716737355712224187471605123\\
& 16076155527684409654570400238048201592216079014878388953289377\\
& 49719295179961005821453423851363883271950672098693260508872,\\
& 39762954454433767605185387100644982494645229455231103764604600\\
& 75680920434284467551983954159980261583596181563104869550061319\\
& 24795839145590024418749747738443955462526011221887359649184882\\
& 24721521937104522216932761684872947757965230219413687501533557\\
& 52747026166081121340758270943364014725351790831289188293935 0)
\end{aligned}
$$

sur l'instance :

$$
\begin{aligned}
Q = \;& (25120261118872905028454882032622098920504395855693747697410 16\\
& 67949699097794293131930276978266381248183180862987975626608854\\
& 99308715393061920855375403800406809315052218800326285654854289\\
& 42593457685442089590085426572029313987521416061477395311994780\\
& 04878279997022891451449123236438190121058396233575425226938 07,\\
& 85697654724924021715797265582700182877285739551963095610652558\\
& 88740211433479874066846721519731627161041892655381393742722529\\
& 99231251463619173252816201113331298799042984261519162584406187\\
& 53671899325676451150457223673527455207351036801294832478317009\\
& 70472878047617001792206814759695777426317457416524484864740 8)
\end{aligned}
$$

nous a donné le logarithme discret recherché en $1,21$ secondes.
Il fallait trouver :

$$
\begin{aligned}
k = \;& 44753980858354423535861646297540885362553562295198446713743391\\
& 94484178039089182579593972912638360985934176835145223233377333\\
& 02677346882406891304375147087752841459792695179933179726902092\\
& 52907654634574710447578676949707952759162934795108001824731806\\
& 37253828158890526244960203288755742212400871751334899600321 2.
\end{aligned}
$$

Décompte des opérations	I	M	Total Complexité arithmétique
1, *2*, *10* : **I**, **I**, **I M**	3	1	$O(1)$
4, *9* : Mult			$O(\log(p))$

TAB. 4.1 – Calcul de complexité arithmétique de $\mathsf{Attaque}_{p,a,b}$.

Nous avons effectué dans la table 4.1 le décompte des opérations nécessaires à l'exécution de l'algorithme $\mathsf{Attaque}_{p,a,b}$.
Nous constatons alors que cette attaque requiert un nombre de multiplications dans $\mathbb{F}_p$ majoré par $O(\log(p))$ ainsi que trois inversions dans $\mathbb{F}_p$.

4.2 Nouveaux cryptosystèmes

Les courbes elliptiques sont prisées en cryptographie pour leur structure de groupe et pour leur résistance aux attaques. En effet, en l'an 2000, le problème du logarithme discret sur des courbes elliptiques définies sur des corps de taille 160 bits était réputé aussi difficile que le problème RSA sur des clés de taille 1024 bits (cf. [LV00]).
Ceci dit, des attaques du logarithme discret sur certaines courbes elliptiques ont été menées à bien : par exemple sur celles de cardinal **N** divisible par p, la caractéristique du corps de base (cf. [Sem98], [AS98], [Sma99]), ou bien sur les courbes dites supersingulières (cf. [MOV91]).
D'autres attaques, applicables à un groupe quelconque, cassent la résistance de certaines courbes : par exemple celles dont le cardinal **N** se décompose en facteurs premiers trop petits sont vulnérables face à la méthode de Pohlig-Hellman (cf. [PH78]).

Engendrer des courbes elliptiques résistantes reste un problème d'actualité qui consiste, tant qu'il n'a pas été prouvé quelles courbes sont sûres, à éviter les courbes dont on sait qu'elles ne sont pas sûres. Dans le cadre de protocoles utilisant des courbes elliptiques (comme SSL), on peut dire des courbes utilisées qu'elles sont bonnes, ce qui signifie qu'on les suppose robustes au problème DDH.
Notre propos ici n'est pas de donner une liste des courbes à éviter, ni de donner une définition d'une bonne courbe.
Nous nous contenterons d'utiliser des courbes elliptiques dont le cardinal est un nombre premier distinct de p. Donnons la définition 4.2.1 d'un générateur de courbes elliptiques.

Définition 4.2.1 *On appelle* générateur de courbes elliptiques *tout algorithme probabiliste polynomial qui prend en entrée* 1^k *dans* $\mathbb{1}^*$ *et qui renvoie un groupe effectif représentant un groupe du type* $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ *et un élément* P *de* $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ *tel que :*
- p *est un nombre premier vérifiant* $t_2(k) = \lfloor \log_2(p) \rfloor + 1 = k$ *;*
- a *et* b *sont dans* $\mathbb{F}_p$ *;*
- $\#(E_{\pi(a),\pi(b)}(\mathbb{F}_p))$ *est un nombre premier distinct de* p *;*
- P *est un générateur de* $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$.

Remarque. Cette notion ne détermine pas si un générateur est un bon générateur, dans le sens où l'on ne sait pas si les courbes qu'il fournit sont robustes ou non face aux problèmes DL, CDH ou DDH.

Nous allons nous servir d'un générateur de courbes elliptiques pour construire un générateur de courbes elliptiques sur un anneau de type $\mathbb{F}_p[\varepsilon]$. Nous l'appelons générateur de courbes elliptiques sur un anneau de nombres duaux. Nous en donnons une définition en 4.2.2 qui ne diffère de la définition 4.2.1 que par le type de groupe représenté et par la nature des éléments a et b.

Définition 4.2.2 *On appelle* générateur de courbes elliptiques sur un anneau de nombres duaux *tout algorithme probabiliste polynomial qui prend en entrée* 1^k *dans* $\mathbb{1}^*$ *et qui renvoie un groupe effectif représentant un groupe du type* $E_{a,b}(\mathbb{F}_p[\varepsilon])$ *tel que :*

- p *est un nombre premier vérifiant* $t_2(k) = \lfloor \log_2(p) \rfloor + 1 = k$ *;*
- a *et* b *sont dans* $\mathbb{F}_p[\varepsilon]$ *;*
- $\#(E_{\pi(a),\pi(b)}(\mathbb{F}_p))$ *est un nombre premier distinct de* p *;*
- P *est un générateur de* $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

A partir d'un générateur de courbes elliptiques, on peut construire un générateur de courbes elliptiques sur un anneau de nombres duaux de différentes façons : les parties constantes des coefficients a et b peuvent être fixées par le générateur de courbes elliptiques et les parties infinitésimales peuvent être choisies aléatoirement ou prendre une valeur particulière comme 0, a ou b. Il en est de même pour les coordonnées du point P engendrant le groupe.

Nous dirons alors d'un générateur de courbes elliptiques sur un anneau de nombres duaux qu'il est issu d'un générateur de courbes elliptiques G si la courbe et le générateur de groupe qu'il renvoie se projettent sur ceux renvoyés par le générateur G. C'est l'objet de la définition 4.2.3.

Définition 4.2.3 *Soit* G *un générateur de courbes elliptiques, soit* G' *un générateur de courbes elliptiques sur un anneau de nombres duaux, on dit que* G' *est issu de* G *s'il vérifie :*

$$\forall 1^k \in \mathbb{1}^*, \forall \rho \in \{0,1\}^*,\ \mathsf{G}'(1^k;\rho) = (a,b,p,\mathbf{N},P) \Rightarrow \mathsf{G}(1^k;\rho) = (\pi(a),\pi(b),p,\mathbf{N},\overline{P}).$$

Soit G un générateur de courbes elliptiques, nous retiendrons les constructions décrites par les algorithmes 20 et 21.

Dans le cas de l'algorithme 20, les parties infinitésimales des coefficients a et b et de l'abscisse du point P sont choisies aléatoirement (étapes 2. 4. et 6.). Nous noterons le cryptosystème associé $\mathsf{W}\varepsilon$.
Dans le cas de l'algorithme 21, seule la partie infinitésimale de l'abscisse du point P est choisie aléatoirement (étape 2.) ; les coefficients de la courbe ont une partie infinitésimale nulle. Nous noterons le cryptosystème associé $\mathsf{W}\varepsilon_0$.

Algorithme 20 : G^{ε}

Entrée : $1^k \in \mathbb{1}^*$

Sortie : $(p, a', b', \mathbf{N}, P)$

1. $(p, a, b, \mathbf{N}, (X_0, Y_0)) \longleftarrow \mathsf{G}(1^k)$
2. $a_1 \longleftarrow [\![0, p-1]\!]$
3. $a' := a + a_1\varepsilon$
4. $b_1 \longleftarrow [\![0, p-1]\!]$
5. $b' := b + b_1\varepsilon$
6. $X_1 \longleftarrow [\![1, p-1]\!]$
7. $Y_1 := \left((3X_0^2 + a)X_1 + a_1X_0 + b_1\right)(2Y_0)^{-1} \mod p$
8. $P := (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon)$
9. *RETURN* $(p, a', b', \mathbf{N}, P)$

Algorithme 21 : $\mathsf{G}^{\varepsilon_0}$

Entrée : $1^k \in \mathbb{1}^*$

Sortie : $(p, a, b, \mathbf{N}, P)$

1. $(p, a, b, \mathbf{N}, (X_0, Y_0)) \longleftarrow \mathsf{G}(1^k)$
2. $X_1 \longleftarrow [\![1, p-1]\!]$
3. $Y_1 := (3X_0^2 + a)X_1(2Y_0)^{-1} \mod p$
4. $P := (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon)$
5. *RETURN* $(p, a, b, \mathbf{N}, P)$

Dans les deux cas, les parties constantes sont fixées par le générateur de courbes elliptiques G et les générateurs G^{ε} et $\mathsf{G}^{\varepsilon_0}$ sont issus de G.

Nous définissons alors les cryptosystèmes $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$ sur un procédé de type El Gamal prenant comme paramètres initiaux les courbes engendrées par G^{ε} et $\mathsf{G}^{\varepsilon_0}$. Le texte clair est introduit dans la partie infinitésimale d'un point dont la projection dans $\mathbb{F}_p$ est choisie aléatoirement (étape 2. de l'algorithme 23 pour $\mathsf{E}^{\mathsf{W}\varepsilon}$ et de l'algorithme 26 pour $\mathsf{E}^{\mathsf{W}\varepsilon_0}$).
Nous détaillons les trois algorithmes définissant le cryptosystème $\mathsf{W}\varepsilon$ dans la définition 4.2.4.

Définition 4.2.4 *Soit* G *un générateur de courbes elliptiques, le cryptosystème* $\mathsf{W}\varepsilon = \left(\mathsf{K}^{\mathsf{W}\varepsilon}, \mathsf{E}^{\mathsf{W}\varepsilon}, \mathsf{D}^{\mathsf{W}\varepsilon}\right)$ *défini à partir de* G *est alors :*

Algorithme 22 : $\mathsf{K}^{\mathsf{W}\varepsilon}$

Entrée : $1^k \in \mathbb{1}^*$

Sortie : $((p, a', b', \mathbf{N}, P), \mathsf{Pk}, \mathsf{sk})$

1. $(p, a', b', \mathbf{N}, P) \longleftarrow \mathsf{G}^{\varepsilon}(1^k)$
2. $\mathsf{sk} \longleftarrow [\![1, \mathbf{N} - 1]\!]$
3. $\mathsf{Pk} := \mathsf{sk}pP$
4. *RETURN* $((p, a', b', \mathbf{N}, P), \mathsf{Pk}, \mathsf{sk})$

où G^{ε} *est défini par l'algorithme 20.*

Algorithme 23 : $\mathsf{E}^{\mathsf{W}\varepsilon}$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a', b', \mathbf{N}, P), \mathsf{Pk}) \in \mathsf{P}_{\mathsf{K}^{\mathsf{W}\varepsilon}}(1^k)$
$m \in [\![1, p-1]\!]$

Sortie : (C_1, C_2)

1. $r \longleftarrow [\![1, \mathbf{N}-1]\!]$
2. $(x_0, y_0) \longleftarrow E_{a',b'}(\mathbb{F}_p) \setminus \{O\}$
3. $y_1 := \left(m(3x_0^2 + a'_0) + a'_1 x_0 + b'_1\right)(2y_0)^{-1} \mod p$
4. $M := (x_0 + m\varepsilon, y_0 + y_1\varepsilon)$
5. $C_2 := r\mathsf{Pk} + M$
6. $C_1 := rpP$
7. *RETURN* (C_1, C_2)

Algorithme 24 : $\mathsf{D}^{\mathsf{W}\varepsilon}$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a', b', \mathbf{N}, P), \mathsf{sk}) \in \mathsf{S}_{\mathsf{K}^{\mathsf{W}\varepsilon}}(1^k)$
$C = (C_1, C_2) \in E_{a',b'}(\mathbb{F}_p[\varepsilon])^2$

Sortie : $m \in \mathbb{F}_p$

1. $M := C_2 - \mathsf{sk}C_1 = (x_0 + m\varepsilon, y_0 + y_1\varepsilon)$
2. *RETURN* m

Concernant le cryptosystème $\mathsf{W}\varepsilon_0$, seul le générateur de courbes elliptiques sur des nombres duaux le différencie du cryptosystème $\mathsf{W}\varepsilon$.
Nous détaillons les trois algorithmes définisssant le cryptosystème $\mathsf{W}\varepsilon_0$ dans la définition 4.2.5, dont les seules modifications avec le cryptosystème $\mathsf{W}\varepsilon$ se situent à l'étape 1. de l'algorithme 25 de génération de clés et à l'étape 3. de l'algorithme 26 de chiffrement ; les algorithmes 24 et 27 de déchiffrement restent identiques.

Définition 4.2.5 *Soit* G *un générateur de courbes elliptiques, le cryptosystème* $\mathsf{W}\varepsilon_0 = \left(\mathsf{K}^{\mathsf{W}\varepsilon_0}, \mathsf{E}^{\mathsf{W}\varepsilon_0}, \mathsf{D}^{\mathsf{W}\varepsilon_0}\right)$ *défini à partir de* G *est alors :*

Algorithme 25 : $\mathsf{K}^{\mathsf{W}\varepsilon_0}$

Entrée : $1^k \in \mathbb{1}^*$

Sortie : $((p, a, b, \mathbf{N}, P), \mathsf{Pk}, \mathsf{sk})$

1. $(p, a, b, \mathbf{N}, P) \longleftarrow \mathsf{G}^{\varepsilon_0}(1^k)$
2. $\mathsf{sk} \longleftarrow [\![1, \mathbf{N}-1]\!]$
3. $\mathsf{Pk} := \mathsf{sk}pP$
4. *RETURN* $((p, a, b, \mathbf{N}, P), \mathsf{Pk}, \mathsf{sk})$

où $\mathsf{G}^{\varepsilon_0}$ *est défini par l'algorithme 21.*

Algorithme 26 : $\mathsf{E}^{\mathsf{W}\varepsilon_0}$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a, b, \mathbf{N}, P), \mathsf{Pk}) \in \mathsf{P}_{\mathsf{K}^{\mathsf{W}\varepsilon_0}}(1^k)$
$m \in [\![1, p-1]\!]$

Sortie : (C_1, C_2)

1. $r \longleftarrow [\![1, \mathbf{N}-1]\!]$
2. $(x_0, y_0) \longleftarrow E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$
3. $y_1 := m(3x_0^2 + a)(2y_0)^{-1} \mod p$
4. $M := (x_0 + m\varepsilon, y_0 + y_1\varepsilon)$
5. $C_2 := r\mathsf{Pk} + M$
6. $C_1 := rpP$
7. *RETURN* (C_1, C_2)

Algorithme 27 : $\mathsf{D}^{\mathsf{W}\varepsilon_0}$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a, b, \mathbf{N}, P), \mathsf{sk}) \in \mathsf{S}_{\mathsf{K}^{\mathsf{W}\varepsilon_0}}(1^k)$
$C = (C_1, C_2) \in E_{a',b'}(\mathbb{F}_p[\varepsilon])^2$

Sortie : $m \in \mathbb{F}_p$

1. $M := C_2 - \mathsf{sk}C_1 = (x_0 + m\varepsilon, y_0 + y_1\varepsilon)$
2. *RETURN* m

Les cryptosystèmes $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$ ressemblent au système de chiffrement El Gamal défini sur un groupe de type $E_{a,b}(\mathbb{F}_p[\varepsilon])$. Ce qui les en distingue est le procédé de codage du texte clair : s'il s'agissait du chiffrement El Gamal, le texte clair serait constitué de la variable M introduite dans les algorithmes de chiffrement 23 et 26. Or dans les cas des cryptosystèmes $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$, le texte clair ne constitue que la partie infinitésimale de l'ordonnée de cette variable M.
Ainsi, nous ne pouvons utiliser les résultats généraux obtenus sur les propriétés de sécurité des systèmes de chiffrement de type El Gamal (exposés dans [Poi02b, p.40-41]).

Dans la section 4.3, nous allons étudier minutieusement certaines propriétés de sécurité du cryptosystème $\mathsf{W}\varepsilon_0$. Quant au cryptosystème $\mathsf{W}\varepsilon$, il généralise le cryptosystème $\mathsf{W}\varepsilon_0$ dans la mesure où les coefficients de la courbe $E_{a,b}(\mathbb{F}_p[\varepsilon])$ utilisée, sont dans $\mathbb{F}_p[\varepsilon]$ dans le premier cas et restent dans $\mathbb{F}_p$ dans le second cas. Ainsi, l'étude du cryptosystème $\mathsf{W}\varepsilon_0$ nous donne une étude du cryptosystème $\mathsf{W}\varepsilon$ sur une partie de ces instances.

4.3 Sécurité du cryptosystème $\mathsf{W}\varepsilon_0$: étude et preuve

Dans cette section, nous étudions certaines propriétés de sécurité du cryptosystème $\mathsf{W}\varepsilon_0$. La partie 4.3.1 traite de son unidirectionnalité puis la partie 4.3.2 traite de son indistinguabilité.

Dans la sous-section 4.3.1, nous montrons que le cryptosystème $\mathsf{W}\varepsilon_0$ défini à partir d'un générateur de courbes elliptiques G est OW-CPA si l'on suppose que le problème CDH sur les courbes qu'il renvoie est difficile à résoudre. Dans un premier temps, nous expliquons brièvement pourquoi $\mathsf{W}\varepsilon_0$ vérifie cette propriété. Puis, nous montrons ce résultat dans la proposition 4.3.1.

Dans la sous-section 4.3.2, nous montrons que le cryptosystème $\mathsf{W}\varepsilon_0$ n'est pas IND-CPA quel que soit le générateur de courbes elliptiques utilisé comme primitive.
Dans un premier temps, nous expliquons brièvement pourquoi le cryptosystème $\mathsf{W}\varepsilon_0$ ne peut être IND-CPA. Puis, nous construisons un IND-CPA adversaire noté $\mathcal{A}^0$ contre $\mathsf{W}\varepsilon_0$, dont nous évaluons dans la proposition 4.3.2 l'avantage sur une courbe elliptique fixée. Enfin, nous montrons dans le corollaire 4.3.1 qu'un générateur de courbes elliptiques ne peut renvoyer de courbe sur laquelle cet avantage serait nul.
Autrement dit, nous exhibons un adversaire IND-CPA dont l'avantage sur le cryptosystème $\mathsf{W}\varepsilon_0$ est non nul.

4.3.1 Unidirectionnalité : OW CPA

La sécurité des cryptosystèmes de type $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$ repose sur le codage choisi. Pour expliquer ce propos, utilisons les notations de la section 4.2 et rappelons succintement que m représente le clair et M le codage de m en un point de la courbe.

En effet, les résultats obtenus sur l'unidirectionnalité (OneWayness abrégé par OW) des cryptosystèmes de type El Gamal (exposés dans [Poi02b, p.40]) montrent, dans le cas d'un groupe $E_{a,b}(\mathbb{F}_p[\varepsilon])$, qu'il est aussi difficile de retrouver M à partir de Pk, C_1 et C_2 que de calculer le CDH en base P de deux éléments de $E_{a,b}(\mathbb{F}_p[\varepsilon])$.
Dans le cas des cryptosystèmes $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$, le clair m constitue seulement la partie infinitésimale de l'abscisse de l'élément M écrit sous la forme $M = [x : y : 1]$ (soit $x = x_0 + m\varepsilon$). La question est donc : peut-on retrouver m à partir de Pk, C_1 et C_2 sans nécessairement calculer M ? Autrement dit, est-il aussi difficile de calculer M que m ? On sait que connaître M c'est en particulier connaître m ; mais est-il possible de retrouver M à partir de m ?
Dans le cas du cryptosystème $\mathsf{W}\varepsilon_0$, l'étude effectuée en section 2.2 nous permet de répondre positivement à cette question : à partir du lemme 2.2.9, y_0 est déterminé par $\Lambda(C_2)$ et m. Il n'existe alors au plus que trois éléments M susceptibles de coder le clair m à C_2 fixé. Ainsi il est possible de retrouver M à partir de m avec une probabilité supérieure à $1/3$.

Dans le cadre de la preuve de sécurité de la proposition 4.3.1, on utilise cette remarque pour calculer le CDH de deux points de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, à partir de Pk, C_1, C_2 et m. Ce résultat est énoncé dans le lemme 4.3.1 dont les hypothèses correspondent au contexte rencontré plus tard dans la preuve de sécurité de la proposition 4.3.1.

Lemme 4.3.1 *Soit a et b deux éléments de $\mathbb{F}_p$ de taille k, P générateur de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, P' et C_2 dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ avec $\overline{P'} = P$, soit α et β dans $[\![1, \mathbf{N} - 1]\!]$.*
Posons $C_1 = \Lambda^{-1}(p\beta P, 0)$ et $m = \mathsf{D}^{\mathsf{W}\varepsilon_0}(1^k, ((p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2))$, alors il existe au triplet (x_0, y_0, y_1) tel que :

$$\alpha\beta P = \kappa\overline{C_2 - (x_0 + m\varepsilon, y_0 + y_1\varepsilon)} \text{ où } \kappa = p^{-1} \mod \mathbf{N}. \tag{4.1}$$

Ce triplet est solution du système

$$\begin{cases} y_0 = \dfrac{-m}{2\lceil \mathbf{N}\mathbf{N}'C_2 \rfloor} \\ x_0^3 + ax_0 + b = y_0^2 \\ y_1 = \dfrac{m(3{x_0}^2 + a)}{2y_0} \end{cases} \tag{4.2}$$

qui admet au plus trois solutions.

Preuve. Commençons par remarquer que le système 4.2 d'équations d'inconnues (x_0, y_0, y_1) admet au plus trois solutions.

▽ En effet, la première équation d'inconnue y_0 admet une unique solution. En remplaçant y_0 par cette solution dans la deuxième équation, on obtient un polynôme univarié de degré 3. La deuxième équation d'inconnue x_0 admet alors au plus 3 solutions. Enfin, en remplaçant dans la dernière équation, y_0 par l'unique solution de la première équation et x_0 par chacune des solutions de la deuxième équation, on obtient trois équations d'inconnues y_1 admettant chacune une solution. △

Posons maintenant $M = C_2 - \alpha C_1$ et montrons qu'il existe (x_0, y_0, y_1) vérifiant la relation 4.1.

▽ On a $m = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k, (p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2))$, donc il existe (x_0, y_0, y_1) tels que :

$$M = (x_0 + m\varepsilon, y_0 + y_1\varepsilon) \text{ avec } \begin{cases} x_0^3 + ax_0 + b - y_0^2 = 0 \\ 2y_0y_1 = (3x_0^2 + a)m. \end{cases}$$

De plus, on a $C_2 - M = \alpha C_1 = \alpha\Lambda^{-1}(\beta pP, 0) = \Lambda^{-1}(\alpha\beta pP, 0)$.
Donc, on a $\overline{C_2 - M} = \alpha\beta pP$ dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, d'où $\alpha\beta P = \kappa\overline{C_2 - M}$ avec $\kappa = p^{-1} \mod \mathbf{N}$. △

Enfin, montrons que $y_0 = -\dfrac{m}{2\lceil \mathbf{N}\mathbf{N}'C_2 \rfloor}$.

▽ Pour cela, remarquons que :

$$\lceil \mathbf{N}\mathbf{N}'C_2 \rfloor = \frac{-m}{2y_0}.$$

En effet, on a les relations :

$$\begin{aligned} \lceil \mathbf{N}\mathbf{N}'C_2 \rfloor &= \lceil \mathbf{N}\mathbf{N}'\alpha C_1 \rfloor + \lceil \mathbf{N}\mathbf{N}'M \rfloor = \lceil \mathbf{N}\mathbf{N}'\alpha\Lambda^{-1}(\beta pP, 0) \rfloor + \lceil \mathbf{N}\mathbf{N}'M \rfloor \\ &= p\lceil \mathbf{N}\mathbf{N}'\alpha\Lambda^{-1}(\beta P, 0) \rfloor + \lceil \mathbf{N}\mathbf{N}'M \rfloor = \lceil \mathbf{N}\mathbf{N}'M \rfloor. \end{aligned}$$

Or d'après le lemme 2.2.9, on a $\lceil \mathbf{N}\mathbf{N}'M \rfloor = \dfrac{-m}{2y_0}$. On montre ainsi la relation

$$y_0 = -\frac{m}{2\lceil \mathbf{N}\mathbf{N}'C_2 \rfloor}. \text{ △}$$

On obtient alors le résultat recherché. □

Exemple. Le point $P = (649, 573)$ est un générateur du groupe $E_{244,38}(\mathbb{F}_{701})$ d'ordre premier $\mathbf{N} = 739$.
On pose alors $\mathbf{N}' = 739^{-1} \mod 701 = 535$ et $\kappa = 701^{-1} \mod 739 = 175$.
Les éléments $P' = (649 + 621\varepsilon, 573 + 333\varepsilon)$ et $C_2 = (280 + 234\varepsilon, 346 + 289\varepsilon)$ sont dans $E_{244,38}(\mathbb{F}_{701}[\varepsilon])$.

Pour $\alpha = 512$ et $\beta = 351$, on a d'une part :

– $\beta pP = 351 \times 701(649, 573) = (569, 237)$ et $C_1 = \Lambda^{-1}((569, 237), 0) = (569, 237)$;
– $C_2 - \alpha C_1 = (280 + 234\varepsilon, 346 + 289\varepsilon) - 512(569, 237) = (579 + 45\varepsilon, 444 + 570\varepsilon)$ et $m = 45$;
– $\alpha\beta P = 512 \times 351(649, 573) = (656, 41)$.

D'autre part, calculons les solutions du système

$$\begin{cases} y_0 = \dfrac{-45}{2\lceil 739 \times 535(280 + 234\varepsilon, 346 + 289\varepsilon)\rfloor} = \dfrac{-45}{2\lceil \Theta(26)\rfloor} = \dfrac{-45}{2 \times 26} = 444 \\ x_0^3 + 244x_0 + 38 - 444^2 = 0 \\ y_1 = \dfrac{45({3x_0}^2 + 244)}{2 \times 444}. \end{cases} \tag{4.3}$$

Dans $\mathbb{F}_{701}$, le polynôme $x_0^3 + 244x_0 + 38 - 444^2$ admet trois racines 166, 657 et 579.
Le système 4.3 d'inconnue (x_0, y_0, y_1) admet donc trois solutions en (x_0, y_0, y_1) :

$$(166, 444, 564),\ (657, 444, 373) \text{ et } (579, 444, 570).$$

Les points $\overline{\kappa C_2 - (x_0 + m\varepsilon, y_0 + y_1\varepsilon)} = 175\overline{(280 + 234\varepsilon, 346 + 289\varepsilon) - (x_0 + 45\varepsilon, y_0 + y_1\varepsilon)}$ correspondants sont :

$$(444, 518),\ (21, 395) \text{ et } (656, 41).$$

Ce dernier point est bien $\alpha\beta P$. •

A partir du lemme 4.3.1, nous pouvons alors montrer que l'unidirectionnalité du cryptosystème $\mathsf{W}\varepsilon_0$ est équivalente au problème CDH sur les courbes elliptiques du générateur de clés.
Nous dirons, pour cela, que le problème CDH est dur pour un générateur de courbes elliptiques s'il est difficile pour le problème CDH sur la famille de courbes issues du générateur paramétrées par ses entrées, comme le précise la définition 4.3.1.

Définition 4.3.1 *Soit* G *un générateur de courbes elliptiques, soit* $\{\mathsf{G}(k; r) : k \in \mathbb{N}, r \in R_{\mathsf{G}}\}$ *l'ensemble de ces sorties.*
Le problème CDH est dur pour G *si et seulement si le problème CDH sur* $\mathsf{G}(k; r)$ *paramétré par* $(k, r) \in \mathbb{N} \times R_{\mathsf{G}}$ *est difficile à résoudre.*

Nous montrons alors la proposition 4.3.1 assurant la sécurité OW-CPA du cryptosystème $\mathsf{W}\varepsilon_0$ si le problème CDH est dur pour le générateur de groupe utilisé par le cryptosystème.

Proposition 4.3.1 *Soit* G *un générateur de courbes elliptiques, le cryptosystème* $\mathsf{W}\varepsilon_0$ *défini à partir de* G *est OW-CPA si le problème CDH est dur pour* G.

Preuve. Soit $\mathcal{A}$ un adversaire OW-CPA contre le cryptosystème $\mathsf{W}\varepsilon_0$, et 1^k un entier dans $\mathbb{1}^*$.

Dans un premier temps, nous allons considérer la succession d'expériences aléatoires $(\mathsf{Exp}_i)_{i=0..7}$. La première, Exp_0 formalise l'attaque du cryptosystème $\mathsf{W}\varepsilon_0$ par l'adversaire $\mathcal{A}$; la dernière Exp_7 formalise la résolution du problème CDH par un adversaire $\mathcal{A}'$, explicité le moment venu.
Les transitions effectuées d'une expérience à l'autre sont essentiellement de type 3. (cf. section 1.5) ; cela signifie que les changements sont mineurs et que l'évènement considéré garde la même probabilité. Seul le passage des expériences aléatoires Exp_5 à Exp_6 n'est pas une transition de

ce type. Il s'agit, dans ce cas, de montrer en quoi l'évènement considéré suite à Exp_6 est lié de façon "quasi"-déterministe à celui considéré suite à Exp_5. C'est cette étape qui précise en quoi l'attaque du cryptosystème $\mathsf{W}\varepsilon_0$ permet de résoudre le problème CDH.

Dans un second temps, nous étudions le temps de calcul de l'adversaire $\mathcal{A}'$, afin de s'assurer qu'il reste polynomial si $\mathcal{A}$ l'est.

1. Commençons par étudier la succession d'expériences aléatoires $(\mathsf{Exp}_i)_{i=0..7}$:

$$\begin{array}{lrcl}
\mathsf{Exp}_0 : OW^{\mathcal{A}}_{\mathsf{PKC}}(k) : & ((p,a,b,\mathbf{N},P'),\mathsf{Pk},\mathsf{sk}) & \longleftarrow & \mathsf{K}^{\mathsf{W}\varepsilon_0}(1^k) \\
& m_\star & \longleftarrow & [\![1,p-1]\!] \\
& (C_1,C_2) & \longleftarrow & \mathsf{E}^{\mathsf{W}\varepsilon_0}(1^k,((p,a,b,\mathbf{N},P'),\mathsf{Pk}),m_\star) \\
& m^\star & \longleftarrow & \mathcal{A}\left(1^k,((p,a,b,\mathbf{N},P'),\mathsf{Pk}),(C_1,C_2)\right)
\end{array}$$

Cette expérience aléatoire Exp_0 correspond à l'attaque de l'adversaire OW-CPA $\mathcal{A}$ contre le cryptosystème $\mathsf{W}\varepsilon_0$: un couple de clés de paramètre de sécurité k est généré par $\mathsf{W}\varepsilon_0$, puis un clair $m_\star$ est choisi aléatoirement dans le domaine des clairs sous-jacent à ce couple ; $m_\star$ est chiffré par $\mathsf{W}\varepsilon_0$; enfin l'adversaire $\mathcal{A}$ prend en entrée la clé publique et le cryptogramme issus de $\mathsf{W}\varepsilon_0$ et génère un clair $m^\star$.
On note δ la probabilité de succès de l'adversaire $\mathcal{A}$, c'est à dire la probabilité que les variables $m_\star$ et $m^\star$ vérifient $m_\star = m^\star$ suite à l'expérience Exp_0. On considère $P_0 = Pr\,(m^\star = m_\star : \mathsf{Exp}_0)$. On a :

$$P_0 = \mathsf{Succ}^{\mathsf{OW}}_{\mathcal{A}/\mathsf{PKC}}(k) = \delta.$$

N.B. : à chaque nouvelle expérience, les modifications effectuées par rapport à l'expérience précédente sont encadrées puis expliquées.

Exp_1 :

$$\begin{array}{|rcl|}
\hline
(p,a,b,\mathbf{N},P') & \longleftarrow & \mathsf{G}^{\varepsilon_0}(1^k) \\
\mathsf{sk} & \longleftarrow & [\![1,\mathbf{N}-1]\!] \\
\mathsf{Pk} & = & \mathsf{sk}pP' \\
\hline
m_\star & \longleftarrow & [\![1,p-1]\!] \\
\hline
r & \longleftarrow & [\![1,\mathbf{N}-1]\!] \\
(x_0,y_0) & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p)\setminus\{O\} \\
y_1 & = & m_\star(3x_0^2+a)(2y_0)^{-1} \mod p \\
M & = & (x_0+m_\star\varepsilon, y_0+y_1\varepsilon) \\
C_2 & = & r\mathsf{Pk}+M \\
C_1 & = & rpP' \\
\hline
m^\star & \longleftarrow & \mathcal{A}\left(1^k,((p,a,b,\mathbf{N},P'),\mathsf{Pk}),(C_1,C_2)\right)
\end{array}$$

La transition effectuée entre les expériences Exp_0 et Exp_1 consiste à détailler le fonctionnement des algorithmes publics $\mathsf{K}^{\mathsf{W}\varepsilon_0}$ et $\mathsf{E}^{\mathsf{W}\varepsilon_0}$ (c'est une transition de type 3).
Considérons $P_1 = Pr\,(m^\star = m_\star : \mathsf{Exp}_1)$. Les expériences Exp_0 et Exp_1 sont identiques donc on a :

$$P_1 = P_0.$$

Exp_2 :

$$
\begin{array}{|rcl|}
\hline
(p, a, b, \mathbf{N}, (X_0, Y_0)) & \longleftarrow & \mathsf{G}(1^k) \\
\hline
X_1 & \longleftarrow & [\![1, p-1]\!] \\
Y_1 & = & X_1(3X_0^2 + a)(2Y_0)^{-1} \mod p \\
P' & = & (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon) \\
\hline
\alpha & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
\hline
\mathsf{Pk} & = & \alpha p P' \\
m_\star & \longleftarrow & [\![1, p-1]\!] \\
\hline
\beta & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
\hline
(x_0, y_0) & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\} \\
y_1 & = & m_\star(3x_0^2 + a)(2y_0)^{-1} \mod p \\
M & = & (x_0 + m_\star\varepsilon, y_0 + y_1\varepsilon) \\
C_2 & = & \beta\mathsf{Pk} + M \\
C_1 & = & \beta p P' \\
m^\star & \longleftarrow & \mathcal{A}\left(1^k, ((p, a, b, \mathbf{N}, P'), \mathsf{Pk}), (C_1, C_2)\right) \\
\hline
\end{array}
$$

La transition effectuée entre les expériences Exp_1 et Exp_2 consiste à détailler le fonctionnement de l'algorithme $\mathsf{G}^{\varepsilon_0}$ et à renommer les variables sk et r par α et β respectivement (c'est une transition de type 3).
Considérons $P_2 = Pr\left(m^\star = m_\star : \mathsf{Exp}_2\right)$. Les expériences Exp_1 et Exp_2 sont donc identiques et on a :

$$P_2 = P_1.$$

Exp_3 :

$$
\begin{array}{|rcl|}
\hline
(p, a, b, \mathbf{N}, (X_0, Y_0)) & \longleftarrow & \mathsf{G}(1^k) \\
\hline
\alpha & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
\hline
\beta & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
\hline
X_1 & \longleftarrow & [\![1, p-1]\!] \\
Y_1 & = & X_1(3X_0^2 + a)(2Y_0)^{-1} \mod p \\
P' & = & (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon) \\
\mathsf{Pk} & = & \alpha p P' \\
m_\star & \longleftarrow & [\![1, p-1]\!] \\
(x_0, y_0) & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\} \\
y_1 & = & m_\star(3x_0^2 + a)(2y_0)^{-1} \mod p \\
M & = & (x_0 + m_\star\varepsilon, y_0 + y_1\varepsilon) \\
C_2 & = & \beta\mathsf{Pk} + M \\
C_1 & = & \beta p P' \\
m^\star & \longleftarrow & \mathcal{A}\left(1^k, ((p, a, b, \mathbf{N}, P'), \mathsf{Pk}), (C_1, C_2)\right) \\
\hline
\end{array}
$$

La transition effectuée entre les expériences aléatoires Exp_2 et Exp_3 consiste à déplacer les expériences élémentaires $\alpha \longleftarrow [\![1, \mathbf{N}-1]\!]$ et $\beta \longleftarrow [\![1, \mathbf{N}-1]\!]$ (c'est une transition de type 3). La variable aléatoire α est indépendante des variables X_1, Y_1, P' et la variable aléatoire β est indépendante des variables X_1, Y_1, P', Pk, et $m_\star$. Donc les variables $m^\star$ et $m_\star$ ont même loi lorsque elles font suite aux expériences aléatoires Exp_2 et Exp_3.
Considérons $P_3 = Pr\left(m^\star = m_\star : \mathsf{Exp}_3\right)$, on a :

$$P_3 = P_2.$$

$$
\begin{array}{lrcl}
\mathsf{Exp}_4 : & (p, a, b, \mathbf{N}, (X_0, Y_0)) & \longleftarrow & \mathsf{G}(1^k) \\
& \alpha & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
& \beta & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
\hline
& P & = & (X_0, Y_0) \\
& A & = & \alpha P \\
& B & = & \beta P \\
\hline
& X_1 & \longleftarrow & [\![1, p-1]\!] \\
& Y_1 & = & X_1(3X_0^2 + a)(2Y_0)^{-1} \mod p \\
& P' & = & (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon) \\
\hline
& \mathsf{Pk} & = & \Lambda^{-1}(pA, 0) \\
\hline
& m_\star & \longleftarrow & [\![1, p-1]\!] \\
& (x_0, y_0) & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\} \\
& y_1 & = & m_\star(3x_0^2 + a)(2y_0)^{-1} \mod p \\
& M & = & (x_0 + m_\star\varepsilon, y_0 + y_1\varepsilon) \\
& C_2 & = & \beta\mathsf{Pk} + M \\
\hline
& C_1 & = & \Lambda^{-1}(pB, 0) \\
\hline
& m^\star & \longleftarrow & \mathcal{A}\left(1^k, ((p, a, b, \mathbf{N}, P'), \mathsf{Pk}), (C_1, C_2)\right)
\end{array}
$$

La transition entre les expériences aléatoires Exp_3 et Exp_4 consiste à remplacer les variables Pk et C_1 par $\Lambda^{-1}(pA, 0)$ et $\Lambda^{-1}(pB, 0)$ respectivement, où $A = \alpha P$ et $B = \beta P$. Montrons que les modes d'obtention de ces variables sont équivalents d'une expérience à l'autre (c'est une transition de type 3).

D'après le lemme 2.2.7, pour tout P dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, on a

$$\nu(\alpha P) = \Lambda^{-1}(\alpha p P, 0) \text{ et } \nu(\beta P) = \Lambda^{-1}(\beta p P, 0).$$

De plus, on a bien $\mathsf{Pk} = \nu(\alpha P)$ et $C_1 = \nu(\beta P)$ car $\nu(P) = pP'$ pour tout P' dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ tel que $\overline{P'} = P$. Donc les variables Pk et C_1 gardent la même loi suite aux expériences aléatoires Exp_3 et Exp_4.

Considérons $P_4 = Pr\left(m^\star = m_\star : \mathsf{Exp}_4\right)$, on a :

$$P_4 = P_3.$$

$$
\begin{array}{lrcl}
\mathsf{Exp}_5 : & (p, a, b, \mathbf{N}, (X_0, Y_0)) & \longleftarrow & \mathsf{G}(1^k) \\
& \alpha & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
& \beta & \longleftarrow & [\![1, \mathbf{N}-1]\!] \\
& P & = & (X_0, Y_0) \\
& A & = & \alpha P \\
& B & = & \beta P \\
& X_1 & \longleftarrow & [\![1, p-1]\!] \\
& Y_1 & = & X_1(3X_0^2 + a)(2Y_0)^{-1} \mod p \\
& P' & = & (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon) \\
& \mathsf{Pk} & = & \Lambda^{-1}(pA, 0) \\
\hline
& R & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\} \\
& R' & \longleftarrow & E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\} \\
& r & \longleftarrow & [\![1, p-1]\!] \\
& C_2 & = & \Lambda^{-1}\left(R + R', r\right) \\
\hline
& C_1 & = & \Lambda^{-1}(pB, 0) \\
& m^\star & \longleftarrow & \mathcal{A}\left(1^k, ((p, a, b, \mathbf{N}, P'), \mathsf{Pk}), (C_1, C_2)\right)
\end{array}
$$

Cette transition consiste à modifier le mode d'obtention de la variable C_2 par un mode équivalent : C_2 est choisie aléatoirement avec la même distribution que lors de l'expérience précédente (c'est une transition de type 3).
Considérons les variables aléatoires $\beta\mathsf{Pk}$ et M définies au cours de l'expérience Exp_4. On a :

$$\Lambda(\beta\mathsf{Pk}) = \Lambda\left(\beta\Lambda^{-1}(pA,0)\right) = (\beta pA,0) = (\beta p\alpha P,0)\,;$$
$$\text{et } \Lambda(M) = \Lambda((x_0+m_\star\varepsilon, y_0+y_1\varepsilon)) = \left((x_0,y_0),\frac{-m_\star}{2y_0}\right) \text{ d'après le lemme 2.2.9.}$$

De plus, P et (x_0,y_0) suivent une loi uniforme dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p)\setminus\{O\}$ et $m_\star$ suit une loi uniforme dans $[\![1,p-1]\!]$. Donc on a $\Lambda(C_2) = \left(\beta\alpha pP + (x_0,y_0),\frac{-m_\star}{2y_0}\right)$ où $\beta\alpha pP$ et (x_0,y_0) suivent tous deux une loi uniforme dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p)\setminus\{O\}$ et $\frac{-m_\star}{2y_0}$ suit une loi uniforme dans $[\![1,p-1]\!]$. Donc C_2 suit la même loi que la variable aléatoire $\Lambda^{-1}(R+R',r)$ définie dans l'expérience Exp_5.
Ainsi, les variables C_2, suite aux expériences aléatoires Exp_4 et Exp_5, suivent la même loi. De plus, la variable $m_\star$ définie dans Exp_4 est égale à $\mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k,((p,a,b,\mathbf{N},P'),\alpha),(C_1,C_2))$. On considère alors

$$P_5 = Pr\left(m^\star = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k,((p,a,b,\mathbf{N},P'),\alpha),(C_1,C_2)) : \mathsf{Exp}_5\right),$$

et on a :

$$P_5 = P_4.$$

L'expérience Exp_6 précise comment on peut calculer (avec une probabilité de succès supérieure à $1/3$) $\alpha\beta P$ à l'issue de l'expérience Exp_5. Elle est donc composée de l'expérience Exp_5 à la suite de laquelle d'autres expériences élémentaires ont été rajoutées.

Exp_6 : $(p,a,b,\mathbf{N},(X_0,Y_0)) \longleftarrow \mathsf{G}(1^k)$
$\alpha \longleftarrow [\![1,\mathbf{N}-1]\!]$
$\beta \longleftarrow [\![1,\mathbf{N}-1]\!]$
$P = (X_0,Y_0)$
$A = \alpha P$
$B = \beta P$
$X_1 \longleftarrow [\![1,p-1]\!]$
$Y_1 = X_1(3X_0^2+a)(2Y_0)^{-1} \mod p$
$P' = (X_0+X_1\varepsilon, Y_0+Y_1\varepsilon)$
$\mathsf{Pk} = \Lambda^{-1}(pA,0)$
$R \longleftarrow E_{\pi(a),\pi(b)}(\mathbb{F}_p)\setminus\{O\}$
$R' \longleftarrow E_{\pi(a),\pi(b)}(\mathbb{F}_p)\setminus\{O\}$
$r \longleftarrow [\![1,p-1]\!]$
$C_2 = \Lambda^{-1}(R+R',r)$
$C_1 = \Lambda^{-1}(pB,0)$
$m^\star \longleftarrow \mathcal{A}\left(1^k,((p,a,b,\mathbf{N},P'),\mathsf{Pk}),(C_1,C_2)\right)$

$y_0 = -m^\star(2r)^{-1} \mod p$
$x_0 \longleftarrow \mathsf{RootOf}(X^3+aX+b-{y_0}^2)$
$y_1 = m^\star(3{x_0}^2+a)(2y_0)^{-1} \mod p$
$C = C_2 - (x_0+m^*\varepsilon, y_0+y_1\varepsilon)$
$\kappa = p^{-1} \mod \mathbf{N}$
$C^\star = \kappa\overline{C}$

Considérons maintenant l'expérience aléatoire Exp_6 où $\mathsf{RootOf}(Q(X))$ est un algorithme probabiliste qui renvoie une racine choisie uniformément dans l'ensemble des racines d'un polynôme univarié Q. Soit $P_6 = Pr\left(C^\star = \alpha\beta P : \mathsf{Exp}_6\right)$. On a :

$$P_6 \geqslant Pr\left(C^\star = \alpha\beta P \bigcap m^\star = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k, ((p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2)) : \mathsf{Exp}_6\right).$$

Or d'après le lemme 4.3.1 :

$$Pr\left(C^\star = \alpha\beta P \,\middle|\, m^\star = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k, ((p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2)) : \mathsf{Exp}_6\right) = \frac{1}{3}.$$

Ainsi on trouve :

$$P_6 \geqslant \frac{1}{3} Pr\left(m^\star = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k, ((p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2)) : \mathsf{Exp}_6\right).$$

Or les évènements $m^\star = \mathsf{D}^{\mathsf{W}_{\varepsilon_0}}(1^k, ((p, a, b, \mathbf{N}, P'), \alpha), (C_1, C_2))$, suite aux expériences aléatoires Exp_5 et Exp_6, sont identiques. D'où la relation :

$$P_6 \geqslant \frac{P_5}{3}.$$

$$\begin{array}{lrcl}
\mathsf{Exp}_7 : & (p, a, b, \mathbf{N}, (X_0, Y_0)) & \longleftarrow & \mathsf{G}(1^k) \\
& \alpha & \longleftarrow & [\![1, \mathbf{N} - 1]\!] \\
& \beta & \longleftarrow & [\![1, \mathbf{N} - 1]\!] \\
& P & = & (X_0, Y_0) \\
& A & = & \alpha P \\
& B & = & \beta P \\
\boxed{} & C^\star & \longleftarrow & \mathcal{A}'\left(1^k, (p, a, b, (X_0, Y_0)), A, B\right)
\end{array}$$

Cette dernière étape consiste à déduire de l'expérience Exp_6 un algorithme probabiliste $\mathcal{A}'$ qui, à l'aide de l'adversaire OW-CPA $\mathcal{A}$ contre le cryptosystème $\mathsf{W}\varepsilon_0$, traite le problème calculatoire de Diffie Hellman sur les courbes elliptiques $E_{a,b}$ générées par G (c'est une transition de type 3).

Algorithme 28 : $\mathcal{A}'$

Entrées : $1^k \in \mathbb{1}^*$
$(p, a, b, \mathbf{N}, (X_0, Y_0)) \in \mathsf{G}(1^k)$
$A = \alpha P \in E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ *avec* $P = (X_0, Y_0)$
$B = \beta P \in E_{\pi(a),\pi(b)}(\mathbb{F}_p)$

Sortie : $C^\star$ *tel que* $C^* = \alpha\beta P$.

1. $A' = pA$
2. $B' = pB$
3. $X_1 \longleftarrow [\![1, p-1]\!]$
4. $Y_1 = X_1(3X_0^2 + a)(2Y_0)^{-1} \mod p$
5. $P' = (X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon)$
6. $\mathsf{Pk} = \Lambda^{-1}(A', 0)$
7. $R \longleftarrow E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$
8. $R' \longleftarrow E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$
9. $r \longleftarrow [\![1, p-1]\!]$
10. $C_2 = \Lambda^{-1}(R + R', r)$
11. $C_1 = \Lambda^{-1}(B', 0)$
12. $m^\star \longleftarrow \mathcal{A}(((p, a, b, \mathbf{N}, P'), \mathsf{Pk}), (C_1, C_2))$
13. $y_0 = -m^\star(2r)^{-1} \mod p$
14. $x_0 \longleftarrow \mathsf{RootOf}(X^3 + aX + b - {y_0}^2)$
15. $y_1 = m^\star(3{x_0}^2 + a)(2y_0)^{-1} \mod p$
16. $C = C_2 - (x_0 + m^\star\varepsilon, y_0 + y_1\varepsilon)$
17. $\kappa = p^{-1} \mod \mathbf{N}$
18. $C^\star = \kappa\overline{C}$
19. *RETURN* $C^\star$

Considérons $P_7 = Pr(C^\star = \alpha\beta P : \mathsf{Exp}_7)$, c'est la probabilité de succès de l'attaquant $\mathcal{A}'$ sur le problème calculatoire de Diffie Hellman. Notons-la δ'.

Les expériences aléatoires Exp_6 et Exp_7 sont identiques, donc $P_6 = P_7 = \delta'$.

Cette succession d'expériences aléatoires nous donne l'inégalité suivante :

$$\delta' \geqslant \frac{\delta}{3}.$$

Ainsi, s'il existe un adversaire OW-CPA $\mathcal{A}$ contre le cryptosystème $\mathsf{W}\varepsilon_0$ de probabilité de succès δ, alors il existe un attaquant $\mathcal{A}'$ du problème calculatoire de Diffie Hellman de probabilité de succès supérieure à $\delta/3$.

2. Calculons maintenant le temps d'exécution de l'adversaire $\mathcal{A}'$.

Voici, d'abord, quelques précisions sur l'évaluation du nombre d'opérations effectuées par les principaux algorithmes :

– d'après le résultat 5 obtenu page 69, l'algorithme $\mathsf{Som}_{p,a,b}$ permettant d'additionner deux éléments de $E_{a,b}$ a un temps de calcul égal à $\mathbf{I}(p) + 7\mathbf{M}(p) + 3\mathbf{S}(p)$, soit une complexité en nombre d'opérations dans $\mathbb{F}_p$ égale à $O(1)$;

Instruction	Algorithme(s)	Nombre d'opérations dans $\mathbb{F}_p$
1. 2. 18.	Mult	$O(\log(p))$
3. 4. 5. 7. 8. 9. 13. 15. 17.	Algorithmes de faible complexité	$O(1)$
6. 10. 11.	Lambda^{-1}	$O(\log(\mathbf{N})) = O(\log(p))$
12.	$\mathcal{A}$	$\tau(p)$
14.	RootOf	$O(\log(p))$
16.	$\mathsf{Som}_{p,a,b}$	$O(1)$

TAB. 4.2 – Calcul de complexité de $\mathcal{A}'$

- d'après la proposition 3.1.2, l'algorithme Mult permettant de calculer kP dans $E_{a,b}(\mathbb{F}_p[\varepsilon])$ a une complexité en $O\left(\log(k) T_{\mathsf{Som}}(4\log(p))\right)$ où T_{Som} représente le temps de calcul de l'algorithme $\mathsf{Som}_{p,a,b}$;
- l'algorithme Lambda^{-1} s'effectue en $O(\log(\mathbf{N}p))$, soit $O(\log(p))$;
- on peut choisir comme algorithme RootOf d'extraction de racines dans $\mathbb{F}_p$ l'algorithme 14.15, exposé dans l'ouvrage [vzGG99, p.368], dont la complexité pour un polynôme de degré trois est en $O(\log(p))$.

Notons $\tau(p)$ la complexité de l'algorithme $\mathcal{A}$. On obtient alors le tableau 4.2 qui donne l'algorithme utilisé pour chaque instruction de $\mathcal{A}'$ ainsi que sa complexité.

En notant τ' la complexité de $\mathcal{A}'$, on obtient :

$$\tau'(p) = \tau(p) + O\left(\log(p)\right).$$

Donc $\mathcal{A}'$ est en temps polynomial si $\mathcal{A}$ l'est. Ce qui achève la preuve. □

Illustrons la capacité de l'adversaire $\mathcal{A}'$ à résoudre le problème CDH sur une courbe elliptique sur un exemple de petite taille.

Exemple. Considérons l'expérience Exp_7 de la preuve de la proposition 4.3.1. Supposons que les algorithmes aléatoires aient engendré pour le paramètre de sécurité 10 les résultats suivants :

$$(p, a, b, \mathbf{N}, (X_0, Y_0)) = (701, 244, 38, 739, (649, 573)),\ \alpha = 512 \text{ et } \beta = 351$$

Ainsi, on a

$$P = (649, 573),\ A = (168, 245) \text{ et } B = (422, 1).$$

Alors l'algorithme $\mathcal{A}'$ accepte ces données en entrée et effectue les calculs suivants :

1. $A' = 701(168, 245) = (430, 237)$
2. $B' = 701(422, 1) = (569, 237)$
3. supposons que $X_1 = 621$
4. $Y_1 = 621(3 \times 649^2 + 244)(2 \times 573)^{-1} \mod 701 = 333$
5. $P' = (649 + 621\varepsilon, 573 + 333\varepsilon)$
6. $\mathsf{Pk} = \Lambda^{-1}((430, 237), 0) = (430, 237)$
7. supposons que $R = (619, 96)$
8. supposons que $R' = (617, 407)$
9. supposons que $r = 26$
10. $C_2 = \Lambda^{-1}((619, 96) + (617, 407), 26) = \Lambda^{-1}((280, 346), 26) = (280 + 234\varepsilon, 346 + 289\varepsilon)$
11. $C_1 = \Lambda^{-1}((569, 237), 0) = (569, 237)$
12. supposons que $\mathcal{A}$ renvoie 45 avec une probabilité δ
13. $y_0 = -45(2 \times 26)^{-1} \mod 701 = 444$
14. supposons $x_0 = 579$ avec une probabilité $1/3$ (cf. Exemple p. 99)
15. $y_1 = 45(3 \times 579^2 + 244)(2 \times 444)^{-1} \mod 701 = 570$
16. $C = (280 + 234\varepsilon, 346 + 289\varepsilon) - (579 + 45\varepsilon, 444 + 570\varepsilon) = (15, 391)$
17. $\kappa = 701^{-1} \mod 739 = 175$
18. $C^\star = 175(15, 391) = (656, 41)$
19. RETURN $(656, 41)$

L'algorithme $\mathcal{A}'$ renvoie le résultat recherché soit $\alpha\beta P = 512 \times 351(649, 573) = (656, 41)$ avec une probabilité égale à $\delta/3$ (cf étapes 12. et 14.). •

4.3.2 Indistinguabilité : IND CPA

Les résultats obtenus sur la sécurité sémantique des cryptosystèmes de type El Gamal (exposés dans [Poi02b, p.41]) montrent que si les messages sont codés par des éléments de $E_{a,b}(\mathbb{F}_p[\varepsilon])$, il est aussi difficile de tirer une information d'un cryptogramme que de déterminer si un triplet de $E_{a,b}(\mathbb{F}_p[\varepsilon])$ est de Diffie-Hellman.
Ce théorème ne s'applique pas aux cryptosystèmes $\mathsf{W}\varepsilon$ et $\mathsf{W}\varepsilon_0$, parce qu'un message clair ne constitue que la partie infinitésimale de l'abscisse d'un élément de $E_{a,b}(\mathbb{F}_p[\varepsilon])$.

L'étude de sécurité du cryptosystème $\mathsf{W}\varepsilon_0$ montre malheureusement que ce cryptosystème n'est pas sémantiquement sûr. Ce résultat provient du lemme 4.3.1 qui permet, à partir d'un clair et d'un de ses cryptogrammes associé, de calculer la variable aléatoire y_0 qui a servi au chiffrement. Or la distribution de cette variable n'est pas uniforme dans $\mathbb{F}_p^\star$.
Pour savoir si un cryptogramme est associé au clair m_0 plutôt qu'au clair m_1, il suffit alors de calculer cette variable pour chacun des deux candidats, et de choisir, parmi les deux valeurs y_0 trouvées, celle qui a la plus forte probabilité d'avoir été choisie lors du chiffrement.
Pour pouvoir réaliser ce type d'attaque il faut aussi être capable d'évaluer la distribution de y_0. Ce calcul ne pose pas de problème. Les détails de cette attaque sont explicités par les algorithmes 29 et 30 qui définissent l'adversaire $\mathcal{A}^0$.

Nous allons montrer l'efficacité de cet adversaire dans la proposition 4.3.2 et le corollaire 4.3.1, en calculant l'avantage de l'adversaire pour une courbe elliptique fixée puis en montrant qu'un générateur de courbes elliptiques ne peut renvoyer de courbe annulant cet avantage.
Pour évaluer l'avantage de l'adversaire $\mathcal{A}^0$ sur une courbe donnée $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, nous avons besoin de connaître la distribution des ordonnées des points de la courbe. Plus précisément nous considérons pour y dans $\mathbb{F}_p$, le nombre n_y de points de $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$ d'ordonnée

y c'est à dire le nombre de racines du polynôme $x^3 + ax + b - y^2$ de $\mathbb{F}_p[x]$. L'entier n_y varie alors entre 0 et 3. Enfin pour i entre 0 et 3, on note N_i le nombre de valeurs y dans $\mathbb{F}_p^\star$ telles que le polynôme $x^3 + ax + b - y^2$ admette exactement i racines.
Le lemme 4.3.2 précise ces notations et il nous permettra, dans notre contexte, d'évaluer l'avantage de l'adversaire $\mathcal{A}^0$.

Lemme 4.3.2 *Considérons une courbe elliptique $E_{a,b}(\mathbb{F}_p)$ de cardinal premier $\mathbf{N}$.*
Pour tout y dans $\mathbb{F}_p$, notons $n_y = Card\left(\{x \in \mathbb{F}_p : x^3 + ax + b - y^2 = 0\}\right)$.
Pour i entier compris entre 0 et 3, posons $N_i = Card\left(\{y \in \mathbb{F}_p^\star : n_y = i\}\right)$. On a :

$$\forall i \in [\![0,3]\!], Pr\left(n_y = i : (x,y) \leftarrow E_{a,b}(\mathbb{F}_p) \setminus \{O\}\right) = \frac{i \times N_i}{\mathbf{N} - 1}$$

Preuve Dans $\mathbb{F}_p$, un polynôme du troisième degré admet entre 0 et 3 racines distinctes. Ainsi les valeurs n_y varient entre 0 et 3 selon y dans $\mathbb{F}_p^\star$.
On suppose $\mathbf{N}$ premier, donc la courbe elliptique $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ n'admet pas de points d'ordre 2. Or ces points (x,y) sont caractérisés par la condition $y = 0$ (cf. [Sil85]). Donc pour tout (x,y) dans $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ on a $y \neq 0$.
Maintenant, pour y dans $\mathbb{F}_p^\star$, l'ensemble $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$ admet exactement n_y points (x_0, y_0) tels que $y_0 = y$. Ainsi $E_{\pi(a),\pi(b)}(\mathbb{F}_p) \setminus \{O\}$ contient exactement $N_1 + 2N_2 + 3N_3 = \mathbf{N} - 1$ et le nombre de points (x,y) vérifiant $n_y = i$ est exactement de $i \times N_i$ pour i entre 1 et 3. On en déduit alors la relation cherchée. □
Considérons l'adversaire IND-CPA $\mathcal{A}^0 = (\mathcal{A}_1^0, \mathcal{A}_2^0)$ contre le cryptosystème $\mathsf{W}^{\varepsilon_0}$ défini par les algorithmes 29 et 30.
A la première étape (algorithme 29), l'adversaire choisit aléatoirement deux clairs distincts dans l'ensemble des clairs possibles, à savoir $\mathbb{F}_p^\star$.

Algorithme 29 : $\mathcal{A}_1^0$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a, b, \mathbf{N}, P), \mathsf{Pk}) \in \mathsf{P}_{\mathsf{K}^{\mathsf{W}^{\varepsilon_0}}}(1^k)$.

Sortie : (m_0, m_1, s) .

1. $m_0 \longleftarrow [\![1, p-1]\!]$
2. $m_1 \longleftarrow [\![1, p-1]\!] \setminus \{m_0\}$
3. $s = (m_0, m_1)$
4. *RETURN* (m_0, m_1, s)

A la seconde étape (algorithme 30), l'adversaire calcule pour chacune des valeurs m_0 et m_1, l'ordonnée y_0 qui aurait servi à leur chiffrement (étapes 3. et 4.). Il calcule le nombre de points de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ ayant ces valeurs trouvées pour ordonnée (étapes 5. et 6.), en utilisant l'algorithme $\mathsf{RootsOf}$ qui renvoie toutes les racines d'un polynôme univarié. Puis, si les deux nombres obtenus sont différents, il choisit le candidat le plus fréquent. Dans le meillleur des cas (pour l'adversaire) il se peut qu'une des deux valeurs trouvées ne soit pas l'ordonnée d'un point de $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$, l'adversaire est alors sûr du résultat. Au contraire dans le pire des cas, ces deux nombres sont égaux et l'adversaire choisit aléatoirement un des deux candidats.

Algorithme 30 : $\mathcal{A}_2^0$

Entrées : $1^k \in \mathbb{1}^*$
$((p, a, b, \mathbf{N}, P), \mathsf{Pk}) \in \mathsf{P}_{\mathsf{K}^{\mathsf{W}\varepsilon_0}}(1^k)$
$(C_1, C_2) \in \mathsf{E}^{\mathsf{W}\varepsilon_0}(1^k)$
$s = (m_0, m_1) \in [\![0, p-1]\!]^2$ *avec* $m_0 \neq m_1$

Sortie : $\delta^\star$.

1. $\mathbf{N}' = \mathbf{N}^{-1} \mod p$
2. $y = -(2\lceil \mathbf{N}\mathbf{N}' C_2 \rfloor)^{-1} \mod p$
3. $y_0 = m_0 y \mod p$
4. $y_1 = m_1 y \mod p$
5. $n_0 = Card\left(\mathsf{RootsOf}\left(X^3 + aX + b - y_0^2\right)\right)$
6. $n_1 = Card\left(\mathsf{RootsOf}\left(X^3 + aX + b - y_1^2\right)\right)$
7. ***If*** $n_1 = n_0$
8. ***then*** $\delta^\star \longleftarrow \{0, 1\}$
9. ***else if*** $n_1 < n_0$
10. ***then*** $\delta^\star = 0$
11. ***else*** $\delta^\star = 1$
12. ***endif***
13. ***endif***
14. *RETURN* $(\delta^\star)$

L'avantage de l'adversaire $\mathcal{A}^0$ (défini page 38) dépend alors de la distribution de la variable y_0 lorsque l'on choisit uniformément un point dans la courbe $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ choisie par le générateur. Ce résultat est détaillé dans la proposition 4.3.2.

Proposition 4.3.2 *Soit* G *un générateur de courbe elliptique. Considérons un entier* k *et une courbe elliptique* $E_{a_0,b_0}(\mathbb{F}_{p_0})$ *issue de* $\mathsf{G}(1^k)$.
Conservons les notations du lemme 4.3.2 concernant $E_{a_0,b_0}(\mathbb{F}_{p_0})$.
Soit $\mathcal{A}^0 = \left(\mathcal{A}_1^0, \mathcal{A}_2^0\right)$ *l'adversaire IND-CPA défini par les algorithmes 29 et 30. Notons :*

$$\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0} = 2\left|Pr\left(\delta^\star = \delta_\star \mid (p,a,b) = (p_0,a_0,b_0) : IND^{\mathcal{A}^0}_{\mathsf{W}\varepsilon_0}(k)\right) - 1\right|$$

On a :

$$\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0} = \frac{N_0}{p_0 - 2} + \frac{1}{(\mathbf{N}-1)(p_0-2)}\left(N_1N_2 + 2N_1N_3 + N_2N_3\right)$$

Preuve.
Soit k un entier, soit $E_{a_0,b_0}(\mathbb{F}_{p_0})$ une courbe elliptique issue de $\mathsf{G}(1^k)$. Considérons l'expérience aléatoire $IND^{\mathcal{A}^0}_{\mathsf{W}\varepsilon_0}(k)$, au cours de laquelle nous supposons que les variables (p, a, b) prennent les valeurs (p_0, a_0, b_0). Nous la notons :

$$\begin{array}{rcl}
\mathsf{Exp}(k, p_0, a_0, b_0) : \ ((p_0, a_0, b_0, \mathbf{N}, P), \mathsf{Pk}, \mathsf{sk}) & \leftarrow & \mathsf{K}^{\mathsf{W}\varepsilon_0}(1^k) \\
(m_0, m_1, s) & \leftarrow & \mathcal{A}_1^0(1^k, ((p_0, a_0, b_0, \mathbf{N}, P), \mathsf{Pk})) \\
\delta_\star & \overset{u}{\leftarrow} & \{0, 1\} \\
(C_1, C_2) & \leftarrow & \mathsf{E}^{\mathsf{W}\varepsilon_0}(1^k, ((p_0, a_0, b_0, \mathbf{N}, P), \mathsf{Pk}), m_{\delta_\star}) \\
\delta^\star & \leftarrow & \mathcal{A}_2^0(1^k, (C_1, C_2), s)
\end{array}$$

En développant les expressions des algorithmes $\mathsf{K}^{\mathsf{W}\varepsilon_0}$ (page 96) (qui fait lui-même appel à $\mathsf{G}^{\varepsilon_0}$ (page 95)), $\mathcal{A}_1^0$ (page 109), $\mathsf{E}^{\mathsf{W}\varepsilon_0}$ (page 97) et $\mathcal{A}_2^0$ (page 110), l'expérience devient alors :

$\mathsf{Exp}(k, p_0, a_0, b_0)$:

$(p_0, a_0, b_0, \mathbf{N}, (X_0, Y_0))$	$\longleftarrow$	$\mathsf{G}(1^k)$
X_1	$\longleftarrow$	$[\![1, p_0 - 1]\!]$
Y_1	$=$	$\left((3X_0^2 + a_0)X_1\right)(2Y_0)^{-1} \mod p_0$
P	$=$	$(X_0 + X_1\varepsilon, Y_0 + Y_1\varepsilon)$
sk	$\longleftarrow$	$[\![1, \mathbf{N} - 1]\!]$
Pk	$=$	$\mathsf{sk}p_0P$
m_0	$\longleftarrow$	$[\![1, p_0 - 1]\!]$
m_1	$\longleftarrow$	$[\![1, p_0 - 1]\!] \setminus \{m_0\}$
s	$=$	(m_0, m_1)
$\delta_\star$	$\overset{u}{\longleftarrow}$	$\{0, 1\}$
r	$\longleftarrow$	$[\![1, \mathbf{N} - 1]\!]$
(x_0, y_0)	$\longleftarrow$	$E_{a_0,b_0}(\mathbb{F}_{p_0}) \setminus \{O\}$
y_1	$=$	$m_{\delta_\star}(3x_0^2 + a_0)(2y_0)^{-1} \mod p_0$
M	$=$	$(x_0 + m_{\delta_\star}\varepsilon, y_0 + y_1\varepsilon)$
C_2	$=$	$r\mathsf{Pk} + M$
C_1	$=$	rp_0P
$\mathbf{N}'$	$=$	$\mathbf{N}^{-1} \mod p_0$
y'	$=$	$-(2\lceil \mathbf{N}\mathbf{N}'C_2 \rfloor)^{-1} \mod p_0$
y'_0	$=$	$m_0y' \mod p_0$
y'_1	$=$	$m_1y' \mod p_0$
n'_0	$=$	$Card\left(\mathsf{RootsOf}\left(X^3 + a_0X + b_0 - {y'_0}^2\right)\right)$
n'_1	$=$	$Card\left(\mathsf{RootsOf}\left(X^3 + a_0X + b_0 - {y'_1}^2\right)\right)$
If $n'_1 = n'_0$	**then**	$\delta^\star \longleftarrow \{0, 1\}$
If $n'_1 < n'_0$	**then**	$\delta^\star = 0$
If $n'_1 > n'_0$	**then**	$\delta^\star = 1$

Tous les évènements de cette preuve sont considérés suite à cette expérience.

Pour évaluer l'avantage de $\mathcal{A}^0$, utilisons la relation obtenue dans le lemme 1.4.1 :

$$\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0} = \left|Pr\left(\delta^\star = 0 \mid \delta_\star = 0\right) - Pr\left(\delta^\star = 0 \mid \delta_\star = 1\right)\right|.$$

Calculons $Pr(\delta^\star = 0 \mid \delta_\star = 0)$ et $Pr(\delta^\star = 0 \mid \delta_\star = 1)$.

Calcul de $Pr(\delta^\star = 0 \mid \delta_\star = 0)$

La variable $\delta^\star$ peut prendre la valeur 0 seulement si $n'_1 = n'_0$ ou $n'_1 < n'_0$, donc on a

$$\begin{aligned} Pr(\delta^\star = 0 \mid \delta_\star = 0) &= Pr\left(\left(\delta^\star = 0 \cap n'_1 = n'_0\right) \mid \delta_\star = 0\right) \\ &\quad + Pr\left(\left(\delta^\star = 0 \cap n'_1 < n'_0\right) \mid \delta_\star = 0\right) \\ &= Pr\left(\delta^\star = 0 \mid \left(n'_1 = n'_0 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_1 = n'_0 \mid \delta_\star = 0\right) \quad (4.4) \\ &\quad + Pr\left(\delta^\star = 0 \mid \left(n'_1 < n'_0 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_1 < n'_0 \mid \delta_\star = 0\right) \end{aligned}$$

De plus, au vu de l'expérience aléatoire $\mathsf{Exp}(k, p_0, a_0, b_0)$, on obtient les résultats suivants :

$$Pr\left(\delta^\star = 0 \mid (n'_1 = n'_0 \cap \delta_\star = 0)\right) = \frac{1}{2} \text{ et } Pr\left(\delta^\star = 0 \mid (n'_1 < n'_0 \cap \delta_\star = 0)\right) = 1.$$

La relation 4.4 devient :

$$Pr(\delta^\star = 0 \mid \delta_\star = 0) = \frac{1}{2} Pr\left(n'_1 = n'_0 \mid \delta_\star = 0\right) + Pr\left(n'_1 < n'_0 \mid \delta_\star = 0\right) \tag{4.5}$$

Au vu de l'expérience aléatoire $\mathsf{Exp}(k, p_0, a_0, b_0)$, si $\delta_\star = 0$, alors, (C_1, C_2) est le chiffré de m_0 et d'après le lemme 2.2.9, on a $y_0 = -m_0 \left(\lceil \mathbf{NN}' C_2 \rfloor\right)^{-1}$ soit $y_0 = y'_0$. Avec les notations du lemme 4.3.2, on trouve $n'_0 = n_{y_0}$ où $(x_0, y_0) \longleftarrow E_{a_0,b_0}(\mathbb{F}_{p_0}) \setminus \{O\}$. Ainsi, la variable aléatoire n'_0 peut prendre seulement les valeurs 1, 2 et 3 ; elle est indépendante de $\delta_\star$ et d'après le lemme 4.3.2, on a :

$$Pr\left(n'_0 = 1 \mid \delta_\star = 0\right) = \frac{N_1}{\mathbf{N} - 1},\ Pr\left(n'_0 = 2 \mid \delta_\star = 0\right) = \frac{2N_2}{\mathbf{N} - 1} \tag{4.6}$$

$$\text{et } Pr\left(n'_0 = 3 \mid \delta_\star = 0\right) = \frac{3N_3}{\mathbf{N} - 1} \tag{4.7}$$

Nous allons décomposer le membre droit de l'égalité 4.5 comme la somme des termes

$$Pr\left(\left(n'_1 = k_1 \cap n'_0 = k_0\right) \mid \delta_\star = 0\right) = Pr\left(n'_1 = k_1 \mid \left(n'_0 = k_0 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = k_0 \mid \delta_\star = 0\right)$$

pour k_0 variant de 1 à 3 et k_1 variant de 0 à $k_0 - 1$.

Calculons alors les expressions $Pr\left(n'_1 = k_1 \mid (n'_0 = k_0 \cap \delta_\star = 0)\right)$ pour k_0 et k_1 entiers compris respectivement entre 1 et 3 et entre 0 et 3 et montrons que :

$$Pr\left(n'_1 = k_1 \mid \left(n'_0 = k_0 \cap \delta_\star = 0\right)\right) = \begin{cases} \dfrac{N_{k_1}}{p-2} & \text{si } k_1 \neq k_0 \\ \dfrac{N_{k_0} - 1}{p-2} & \text{si } k_1 = k_0 \end{cases} \tag{4.8}$$

$\bigtriangledown$ D'après l'expérience aléatoire $\mathsf{Exp}(k, p_0, a_0, b_0)$, on a la relation $m_1 y'_0 = m_0 y'_1$. De plus, si $\delta_\star = 0$, on a vu que $y'_0 = y_0$ et on en déduit que $y'_1 = \dfrac{m_1}{m_0} y_0$ avec $m_1 \longleftarrow [\![1, p_0 - 1]\!] \setminus \{m_0\}$. La variable $\dfrac{m_1}{m_0}$ suit alors une loi uniforme dans $[\![2, p_0 - 1]\!]$ et y'_1 suit une loi uniforme dans $[\![1, p_0 - 1]\!] \setminus \{y_0\}$. Donc on obtient :

$$\begin{aligned} & Pr\left(n'_1 = k_1 \mid \left(n'_0 = k_0 \cap \delta_\star = 0\right)\right) \\ = \ & Pr\left(n_{y'_1} = k_1 \mid n'_0 = k_0 : y'_1 \leftarrow [\![1, p_0 - 1]\!] \setminus \{y_0\}, (x_0, y_0) \leftarrow E_{a_0,b_0}(\mathbb{F}_{p_0})\right) \end{aligned}$$

Supposons maintenant que $n'_0 = n_{y_0} = k_0$, l'ensemble $[\![1, p_0 - 1]\!] \setminus \{y_0\}$ contient exactement :
- N_{k_1} éléments y tels que $n_y = k_1$ pour $k_1 \neq k_0$,
- $N_{k_0} - 1$ éléments y tels que $n_y = k_0$.

On obtient ainsi les équations 4.8. $\bigtriangleup$

En utilisant les égalités 4.6 et 4.8 pour k_0 allant de 1 à 3 et $k_1 = k_0$, on obtient la relation suivante :

$$\begin{aligned} Pr(n'_1 = n'_0 \mid \delta_\star = 0) \ = \ & Pr\left(n'_1 = 1 \mid \left(n'_0 = 1 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 1 \mid \delta_\star = 0\right) \\ & + Pr\left(n'_1 = 2 \mid \left(n'_0 = 2 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 2 \mid \delta_\star = 0\right) \\ & + Pr\left(n'_1 = 3 \mid \left(n'_0 = 3 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 3 \mid \delta_\star = 0\right) \\ = \ & \frac{1}{(\mathbf{N} - 1)(p_0 - 2)} \left[N_1(N_1 - 1) + 2N_2(N_2 - 1) + 3N_3(N_3 - 1)\right] \end{aligned} \tag{4.9}$$

De la même façon, en utilisant les égalités 4.6 et 4.8 pour k_0 allant de 1 à 3 et k_1 strictement inférieur à k_0, on obtient la relation suivante :

$$\begin{aligned} Pr(n'_1 < n'_0 \mid \delta_\star = 0) &= Pr\left(n'_1 = 0 \mid \left(n'_0 = 1 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 1 \mid \delta_\star = 0\right) \\ &\quad + Pr\left(n'_1 = 0 \mid \left(n'_0 = 2 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 2 \mid \delta_\star = 0\right) \\ &\quad + Pr\left(n'_1 = 1 \mid \left(n'_0 = 2 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 2 \mid \delta_\star = 0\right) \\ &\quad + Pr\left(n'_1 = 0 \mid \left(n'_0 = 3 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 3 \mid \delta_\star = 0\right) \\ &\quad + Pr\left(n'_1 = 1 \mid \left(n'_0 = 3 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 3 \mid \delta_\star = 0\right) \\ &\quad + Pr\left(n'_1 = 2 \mid \left(n'_0 = 3 \cap \delta_\star = 0\right)\right) \times Pr\left(n'_0 = 3 \mid \delta_\star = 0\right) \qquad (4.10) \\ &= \frac{1}{(\mathbf{N}-1)(p_0-2)}\left[N_0N_1 + 2N_0N_2 + 2N_1N_2 + 3N_0N_3 + 3N_1N_3 + 3N_2N_3\right] \end{aligned}$$

A partir des relations 4.9 et 4.10, l'égalité 4.5 devient :

$$\begin{aligned} Pr(\delta^\star = 0 \mid \delta_\star = 0) &= \frac{1}{2(\mathbf{N}-1)(p_0-2)}\left[N_1(N_1-1) + 2N_2(N_2-1) + 3N_3(N_3-1)\right] \qquad (4.11) \\ &\quad + \frac{1}{(\mathbf{N}-1)(p_0-2)}\left[N_0N_1 + 2N_0N_2 + 2N_1N_2 + 3N_0N_3 + 3N_1N_3 + 3N_2N_3\right] \end{aligned}$$

Calcul de $Pr(\delta^\star = 0 \mid \delta_\star = 1)$

Avec le même raisonnement que précédemment, on obtient l'analogue de l'equation 4.5, soit :

$$Pr(\delta^\star = 0 \mid \delta_\star = 1) = \frac{1}{2}Pr\left(n'_1 = n'_0 \mid \delta_\star = 1\right) + Pr\left(n'_1 < n'_0 \mid \delta_\star = 1\right) \qquad (4.12)$$

De le même façon que l'on a obtenu les relations 4.9 et 4.10, on trouve :

$$\begin{aligned} Pr(n'_1 = n'_0 \mid \delta_\star = 1) &= Pr\left(n'_0 = 1 \mid \left(n'_1 = 1 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 1 \mid \delta_\star = 1\right) \\ &\quad + Pr\left(n'_0 = 2 \mid \left(n'_1 = 2 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 2 \mid \delta_\star = 1\right) \\ &\quad + Pr\left(n'_0 = 3 \mid \left(n'_1 = 3 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 3 \mid \delta_\star = 1\right) \\ &= \frac{1}{(\mathbf{N}-1)(p_0-2)}\left[(N_1(N_1-1) + 2N_2(N_2-1) + 3N_3(N_3-1)\right] \end{aligned}$$

et

$$\begin{aligned} Pr(n'_1 < n'_0 \mid \delta_\star = 1) &= Pr\left(n'_0 = 2 \mid \left(n'_1 = 1 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 1 \mid \delta_\star = 1\right) \\ &\quad + Pr\left(n'_0 = 3 \mid \left(n'_1 = 1 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 1 \mid \delta_\star = 1\right) \\ &\quad + Pr\left(n'_0 = 3 \mid \left(n'_1 = 2 \cap \delta_\star = 1\right)\right) \times Pr\left(n'_1 = 2 \mid \delta_\star = 1\right) \\ &= \frac{1}{(\mathbf{N}-1)(p_0-2)}\left[N_2N_1 + N_3N_1 + 2N_3N_2\right] \end{aligned}$$

La relation 4.12 devient alors :

$$\begin{aligned} Pr(\delta^\star = 0 \mid \delta_\star = 1) &= \frac{1}{2(\mathbf{N}-1)(p_0-2)}\left[N_1(N_1-1) + 2N_2(N_2-1) + 3N_3(N_3-1)\right] \\ &\quad + \frac{1}{(\mathbf{N}-1)(p_0-2)}\left[N_2N_1 + N_3N_1 + 2N_3N_2\right] \qquad (4.13) \end{aligned}$$

Calcul de $\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}^{\varepsilon_0}}$

Les égalités 4.11 et 4.13 permettent de calculer

$$\begin{aligned}&\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}^{\varepsilon_0}}\\ &= |Pr(\delta^\star = 0 \mid \delta_\star = 0 : \mathsf{Exp}(k, p_0, a_0, b_0)) - Pr(\delta^\star = 0 \mid \delta_\star = 1 : \mathsf{Exp}(k, p_0, a_0, b_0))|\end{aligned}$$

On obtient :

$$\begin{aligned}&\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}^{\varepsilon_0}}\\ &= \frac{1}{(\mathbf{N}-1)(p_0-2)}\,|N_0N_1 + 2N_0N_2 + 2N_1N_2 + 3N_0N_3 + 3N_1N_3 + 3N_2N_3\\ &\quad -N_2N_1 - N_3N_1 - 2N_3N_2|\\ &= \frac{1}{(\mathbf{N}-1)(p_0-2)}\,|N_0(N_1 + 2N_2 + 3N_3) + N_1N_2 + 2N_1N_3 + N_2N_3|\end{aligned}$$

On remarque enfin que $N_1 + 2N_2 + 3N_3 = \mathbf{N} - 1$ (voir la preuve du lemme 4.3.2 page 109), et on obtient la relation cherchée. □

L'exemple suivant illustre le déroulement d'une attaque IND-CPA de l'adversaire $\mathcal{A}^0$ sur le cryptosystème $\mathsf{W}^{\varepsilon_0}$ pour un paramètre de taille 10 puis calcule l'avantage de $\mathcal{A}^0$ dans ce cas.

Exemple.
Pour $k = 10$ considérons la courbe elliptique $E_{a_0,b_0}(\mathbb{F}_{p_0})$ avec $a_0 = 244$, $b_0 = 38$ et $p_0 = 701$. L'étude de cette courbe montre que :
- le cardinal de $E_{244,38}(\mathbb{F}_{701})$ est le nombre premier 739 ;
- dans $\mathbb{F}^\star_{701}$, on trouve exactement :
  - 224 éléments y tels que le polynôme $x^3 + 244x + 38 - y^2$ n'ait pas de racines ;
  - 344 éléments y tels que le polynôme $x^3 + 244x + 38 - y^2$ ait exactement une racine ;
  - 2 éléments y tels que le polynôme $x^3 + 244x + 38 - y^2$ ait exactement deux racines ;
  - 130 éléments y tels que le polynôme $x^3 + 244x + 38 - y^2$ ait exactement trois racines ;

On a donc $\mathbf{N} = 739$, $N_0 = 224$, $N_1 = 344$, $N_2 = 2$ et $N_3 = 130$ qui vérifient bien

$$N_0 + N_1 + N_2 + N_3 = 224 + 344 + 2 + 130 = 700 = p_0 - 1$$
et
$$N_1 + 2N_2 + 3N_3 = 344 + 2 \times 2 + 3 \times 130 = 738 = \mathbf{N} - 1.$$

Considérons l'expérience $\mathsf{Exp}(10, 701, 244, 38)$ de la preuve de la proposition 4.3.2 présentée page 110 :

L'algorithme $\mathsf{K}^{\mathsf{W}^{\varepsilon_0}}$ prend en entrée 1^{10}
- supposons que $\mathsf{G}(1^{10})$ renvoie $(701, 244, 38, 739, (649, 573))$
- supposons que $X_1 = 621$
- $Y_1 = 621\dfrac{3 \times 649^2 + 244}{2 \times 573} \mod 701 = 333$
- $P = (649 + 621\varepsilon, 573 + 333\varepsilon)$
- supposons que $\mathsf{sk} = 512$
- $\mathsf{Pk} = (430, 237)$

Ainsi $\mathsf{K}^{\mathsf{W}^{\varepsilon_0}}(1^{10})$ renvoie $((701, 244, 38, 739, (649 + 621\varepsilon, 573 + 333\varepsilon)), (430, 237), 512)$

L'adversaire $\mathcal{A}^0_1$ prend en entrée $(1^{10}, (701, 244, 38, 739, (649, 573), (430, 237)))$:
- supposons que $m_0 = 246$

- supposons que $m_1 = 45$
- $s = (246, 45)$

Ainsi $\mathcal{A}_1^0(1^{10}, (701, 244, 38, 739, (649{+}621\varepsilon, 573{+}333\varepsilon), (430, 237)))$ renvoie $(246, 45, (246, 45))$

- supposons que $\delta_\star = 1$

L'algorithme $\mathsf{E}^{\mathsf{W}\varepsilon_0}$ prend en entrée $(1^{10}, (701, 244, 38, 739, (649{+}621\varepsilon, 573{+}333\varepsilon)), (430, 237), 45)$

- supposons que $r = 351$
- supposons que $(x_0, y_0) = (172, 142)$
- $y_1 = 45\dfrac{3 \times 172^2 + 244}{2 \times 142} \mod 701 = 664$
- $M = (172 + 45\varepsilon, 142 + 664\varepsilon)$
- $C_2 = 351(430, 237) + (172 + 45\varepsilon, 142 + 664\varepsilon) = (486 + 488\varepsilon, 356 + 461\varepsilon)$
- $C_1 = 351 \times 701 \times (649 + 621\varepsilon, 573 + 333\varepsilon) = (569, 237)$

Ainsi $\mathsf{E}^{\mathsf{W}\varepsilon_0}(1^{10}, (701, 244, 38, 739, (649 + 621\varepsilon, 573 + 333\varepsilon)), (430, 237), 45)$ renvoie $((569, 237), (486 + 488\varepsilon, 356 + 461\varepsilon))$

L'algorithme $\mathcal{A}_2^0$ prend en entrée $(1^{10}, (701, 244, 38, 739, (649{+}621\varepsilon, 573{+}333\varepsilon)), (430, 237), 45)$

- $\mathbf{N}' = 739^{-1} \mod 701 = 535$
- $y' = -(2\lceil 739 \times 535(486 + 488\varepsilon, 356 + 461\varepsilon)\rfloor)^{-1} \mod 701 = -(2 \times 338)^{-1} \mod 701$ soit $y' = 673$
- $y'_0 = 246 \times 673 \mod 701 = 122$
- $y'_1 = 45 \times 673 \mod 701 = 142$
- $n'_0 = Card\left(\mathsf{RootsOf}\left(X^3 + 244X + 38 - 122^2\right)\right) = 0$
- $n'_1 = Card\left(\mathsf{RootsOf}\left(X^3 + 244X + 38 - 142^2\right)\right) = 2$
- On a $n'_1 > n'_0$, donc $\delta^\star = 1$

Ainsi $\mathcal{A}_2^0$ renvoie $\delta^\star = 1$

De plus on évalue,

$$\begin{aligned} \mathsf{Adv}^{IND|(p,a,b)=(701,244,38)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0} &= \frac{224}{699} + \frac{1}{738 \times 699}(344 \times 2 + 2 \times 344 \times 130 + 2 \times 130) \\ &\simeq 0.495438 \end{aligned}$$

Dans cet exemple, les calculs ont donné $n'_0 = 0$, il n'existe donc aucun point de $E_{244,38}(\mathbb{F}_{701})$ d'ordonnée 122. L'adversaire est alors sûr que $\delta_\star = 1$. •

Malheureusement, il n'existe pas de groupe effectif E_{a_0,b_0,p_0} issu d'un générateur de courbes elliptiques qui annule $\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0}$ (défini dans la proposition 4.3.2). Cela nous permet de montrer le corollaire 4.3.1.

Corollaire 4.3.1 *Soit* G *un générateur de courbes elliptiques. L'avantage de* $\mathcal{A}^0$ *contre le cryptosystème* $\mathsf{W}\varepsilon_0$ *défini à partir de* G *est non nul.*

Preuve. Conservons les notations du lemme 4.3.2 et montrons qu'il n'existe aucune courbe elliptique $E_{\pi(a),\pi(b)}(\mathbb{F}_p)$ de cardinal premier $\mathbf{N}$ distinct de p telle que $N_0 = 0$ et pour laquelle deux des trois coefficients N_1, N_2, N_3 s'annulent.
Etudions les trois cas de figure possibles :

- si on pose $N_0 = N_1 = N_2 = 0$, alors on trouve $N_3 = p - 1$ et $\mathbf{N} = 3N_3 + 1 = 3p + 2$. Or cette égalité est impossible d'après le théorème de Hasse (cf page 89) ;
- si on pose $N_0 = N_1 = N_3 = 0$, alors on trouve $N_2 = p - 1$ et $\mathbf{N} = 2N_2 + 1 = 2p + 1$. Or cette égalité est impossible d'après le théorème de Hasse (cf page 89) ;

– si on pose $N_0 = N_2 = N_3 = 0$, alors on trouve $N_1 = p - 1$ et $\mathbf{N} = p$. Or c'est impossible car l'entier $\mathbf{N}$ est un nombre premier différent de p.

D'après la définition 4.2.1 de générateur de courbes elliptiques, les groupes effectifs renvoyés sont bien de cardinal premier distinct de p. Ainsi pour tout p_0, a_0, b_0 tels que E_{p_0,a_0,b_0} soit un groupe effectif issu de G, on a : $\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0} \neq 0$. Alors, on obtient le résultat recherché : l'avantage de l'adversaire $\mathcal{A}^0$ contre le cryptosystème $\mathsf{W}\varepsilon_0$ est non nul. □

Pour montrer que le système $\mathsf{W}\varepsilon_0$ n'est pas IND CPA, il reste à montrer que l'avantage $\mathsf{Adv}^{IND|(p,a,b)=(p_0,a_0,b_0)}_{\mathcal{A}^0/\mathsf{W}\varepsilon_0}$ n'est pas négligeable.

Conclusion

L'objet de notre étude était les applications cryptographiques des courbes elliptiques définies sur $\mathbb{F}_q[\varepsilon]$ où $\varepsilon^2 = 0$.
Nous avons montré que ces courbes présentaient une approche intéressante dans ce domaine.

1. Leurs implémentations nécessitent deux fois plus d'espace mais travaillent dans le même temps que celles des courbes elliptiques "classiques" (cf. section 3.2.1) : elles restent ainsi (presque) aussi faciles à implémenter.
2. Les problèmes calculatoires de Diffie Hellman et du logarithme discret sur ces courbes elliptiques sont équivalents à leurs homologues définis sur des courbes elliptiques "classiques" (cf. section 3.3.1) : cette propriété est aussi intéressante pour un cryptographe que pour un cryptanalyste. De ce point de vue, elles ont même forces et même faiblesses.
3. Elles ont permis d'attaquer le problème du logarithme discret sur les courbes elliptiques définies sur un corps fini de même cardinal que la courbe (cf.section 4.1) : même si ce type d'attaque existe depuis une dizaine d'années, celle-ci, supplémentaire, reste fulgurante.
4. Elles ont permis de construire le cryptosystème $\mathsf{W}_{\varepsilon_0}$ très proche du système ElGamal dont le niveau de sécurité est équivalent (cf. sections 4.2 et 4.3) : ce système présente l'avantage supplémentaire de ne poser aucun problème de codage.

Les courbes elliptiques définies sur $\mathbb{F}_q[\varepsilon]$ ont néanmoins montré quelques faiblesses.

1. Le problème décisionnel de Diffie Hellman sur ces courbes est facile à résoudre (cf. section 3.3.2) : on peut craindre que ces courbes "transportent trop d'informations redondantes" ; elles laissent peut être "transpirer" de l'information. Et on pourrait réussir à en retrouver (à leur insu).
2. Le cryptosystème $\mathsf{W}_{\varepsilon_0}$ n'est certainement pas IND CPA (cf. section 4.3.2) : ce système de chiffrement ne présente pas un niveau de sécurité élevé.

Il reste beaucoup de choses à étudier sur ce sujet et il ouvre d'autres voies d'étude.

Notamment, celle des courbes elliptiques définies sur l'anneau $\mathbb{F}_q[\varepsilon]$ où $\varepsilon^n = 0$ avec n un entier supérieur à 2. Des constructions analogues de cryptosystème avec ce type de courbes pourraient présenter l'avantage de réduire la bande passante d'un coefficient multiplicateur égal à $\dfrac{n-1}{n}$.

L'étude des morphisme de type λ, ν, Λ ... définis dans la section 2.2 pourrait révéler d'autres relations intéressantes du type de celle du lemme 2.2.9. Celle-ci nous a permis de construire l'adversaire IND CPA de la section 4.3.2.

Enfin, d'autres cryptosystèmes, plus particulièrement des systèmes de signature, peuvent être construits à partir de ces courbes et l'étude de celles-ci pourrait permettre d'en obtenir de plus solides.

Annexe A

Détails des calculs du chapitre 2

A.1 Relations énoncées dans la proposition 2.1.2

Relations I

```
> P1:=y^2*X*Z-z*x*Y^2-a*(z*X+x*Z)*(z*X-Z*x)+(2*y*Y-3*b*z*Z)*(z*X-Z*x);
```

$$P1 := y^2\,X\,Z - z\,x\,Y^2 - a\,(z\,X + x\,Z)\,(z\,X - x\,Z) + (2\,y\,Y - 3\,b\,z\,Z)\,(z\,X - x\,Z)$$

```
> Q1:=y*Y*(Z*y-z*Y)-a*(x*y*Z^2-z^2*X*Y)+(-2*a*z*Z-3*x*X)*(X*y-x*Y)-3*b*
> z*Z*(Z*y-z*Y);
```

$$Q1 := y\,Y\,(Z\,y - z\,Y) - a\,(x\,y\,Z^2 - z^2\,X\,Y) + (-2\,a\,z\,Z - 3\,x\,X)\,(X\,y - x\,Y) - 3\,b\,z\,Z\,(Z\,y - z\,Y)$$

```
> R1:=(z*Y+Z*y)*(Z*y-z*Y)+(3*x*X+a*z*Z)*(z*X-x*Z);
```

$$R1 := (z\,Y + Z\,y)\,(Z\,y - z\,Y) + (3\,x\,X + a\,z\,Z)\,(z\,X - x\,Z)$$

Relations II

```
> P2:=(y*Y-6*b*z*Z)*(x*Y+X*y)+(-2*a*x*X+a^2*z*Z)*(z*Y+Z*y)-3*b*(x*y*Z^2
> +z^2*X*Y)-a*(z*y*X^2+x^2*Z*Y);
```

$$P2 := (y\,Y - 6\,b\,z\,Z)\,(X\,y + x\,Y) + (a^2\,z\,Z - 2\,a\,x\,X)\,(z\,Y + Z\,y) - 3\,b\,(x\,y\,Z^2 + z^2\,X\,Y) - a\,(y\,z\,X^2 + x^2\,Y\,Z)$$

```
> Q2:=y^2*Y^2+3*a*x^2*X^2+(-a^3-9*b^2)*z^2*Z^2-a^2*(z*X+x*Z)^2-2*a^2*z*
> x*Z*X+(9*b*x*X-3*a*b*z*Z)*(z*X+x*Z);
```

$$Q2 := y^2\,Y^2 + 3\,a\,x^2\,X^2 + (-a^3 - 9\,b^2)\,z^2\,Z^2 - a^2\,(z\,X + x\,Z)^2 - 2\,a^2\,z\,x\,Z\,X + (9\,b\,x\,X - 3\,a\,b\,z\,Z)\,(z\,X + x\,Z)$$

```
> R2:=(y*Y+3*b*z*Z)*(Z*y+z*Y)+(3*x*X+2*a*z*Z)*(x*Y+X*y)+a*(x*y*Z^2+z^2*
> X*Y);
```

$$R2 := (y\,Y + 3\,b\,z\,Z)\,(z\,Y + Z\,y) + (3\,x\,X + 2\,a\,z\,Z)\,(X\,y + x\,Y) + a\,(x\,y\,Z^2 + z^2\,X\,Y)$$

A.2 Détails des calculs du lemme 2.3.1

```
> eval([P2,Q2,R2],x=k*epsilon):eval(%,y=1):eval(%,z=0):
> eval(%,X=K*epsilon):eval(%,Y=1):eval(%,Z=0):
> eval(%,a=a0+a1*epsilon):eval(%,b=b0+b1*epsilon);
```

$$[K\,\varepsilon + k\,\varepsilon,\ 1 + 3\,(a0 + a1\,\varepsilon)\,k^2\,\varepsilon^4\,K^2,\ 3\,k\,\varepsilon^2\,K\,(K\,\varepsilon + k\,\varepsilon)]$$

```
> eval(%,epsilon^2=0):eval(%,epsilon^4=0);
```

$$[K\,\varepsilon + k\,\varepsilon,\ 1,\ 0]$$

A.3 Détails des calculs du lemme 2.3.2

```
> eval([P1,Q1,R1],x=x0+x1*epsilon):eval(%,y=y0+y1*epsilon):
> eval(%,z=1):
> eval(%,X=k*epsilon):eval(%,Y=1):eval(%,Z=0):
> eval(%,a=a0+a1*epsilon):eval(%,b=b0+b1*epsilon);
```

$$[-x0 - x1\,\varepsilon - (a0 + a1\,\varepsilon)\,k^2\,\varepsilon^2 + (2\,y0 + 2\,y1\,\varepsilon)\,k\,\varepsilon,$$
$$-y0 - y1\,\varepsilon + (a0 + a1\,\varepsilon)\,k\,\varepsilon - 3\,(x0 + x1\,\varepsilon)\,k\,\varepsilon\,(k\,\varepsilon\,(y0 + y1\,\varepsilon) - x0 - x1\,\varepsilon),$$
$$-1 + 3\,(x0 + x1\,\varepsilon)\,k^2\,\varepsilon^2]$$

```
> expand(%):eval(%,epsilon^2=0):eval(%,epsilon^3=0):\
> eval(%,epsilon^4=0);
```

$$[-x0 - x1\,\varepsilon + 2\,k\,\varepsilon\,y0,\ -y0 - y1\,\varepsilon + k\,\varepsilon\,a0 + 3\,k\,\varepsilon\,x0^2,\ -1]$$

```
> -%:collect(%,epsilon);
```

$$[(x1 - 2\,k\,y0)\,\varepsilon + x0,\ (y1 - k\,a0 - 3\,k\,x0^2)\,\varepsilon + y0,\ 1]$$

A.4 Détails des calculs du lemme 2.3.3

```
> eval([P2,Q2,R2],x=x0+x1*epsilon):eval(%,y=y1*epsilon):eval(%,z=1):
> eval(%,X=x0+X1*epsilon):eval(%,Y=Y1*epsilon):eval(%,Z=1):
> eval(%,a=a0+a1*epsilon):Res:=eval(%,b=b0+b1*epsilon);
```

$$Res := [(y1\,\varepsilon^2\ Y1 - 6\,b0 - 6\,b1\,\varepsilon)\,((x0 + X1\,\varepsilon)\,y1\,\varepsilon + (x0 + x1\,\varepsilon)\ Y1\,\varepsilon)$$
$$+ ((a0 + a1\,\varepsilon)^2 - 2\,(a0 + a1\,\varepsilon)\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon))\,(Y1\,\varepsilon + y1\,\varepsilon)$$
$$- 3\,(b0 + b1\,\varepsilon)\,((x0 + x1\,\varepsilon)\,y1\,\varepsilon + (x0 + X1\,\varepsilon)\ Y1\,\varepsilon)$$
$$- (a0 + a1\,\varepsilon)\,(y1\,\varepsilon\,(x0 + X1\,\varepsilon)^2 + (x0 + x1\,\varepsilon)^2\ Y1\,\varepsilon),$$
$$y1^2\,\varepsilon^4\ Y1^2 + 3\,(a0 + a1\,\varepsilon)\,(x0 + x1\,\varepsilon)^2\,(x0 + X1\,\varepsilon)^2 - (a0 + a1\,\varepsilon)^3 - 9\,(b0 + b1\,\varepsilon)^2$$
$$- (a0 + a1\,\varepsilon)^2\,(2\,x0 + X1\,\varepsilon + x1\,\varepsilon)^2 - 2\,(a0 + a1\,\varepsilon)^2\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon)$$
$$+(9\,(b0 + b1\,\varepsilon)\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon) - 3\,(a0 + a1\,\varepsilon)\,(b0 + b1\,\varepsilon))$$
$$(2\,x0 + X1\,\varepsilon + x1\,\varepsilon),$$
$$(y1\,\varepsilon^2\ Y1 + 3\,b0 + 3\,b1\,\varepsilon)\,(Y1\,\varepsilon + y1\,\varepsilon)$$
$$+ (3\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon) + 2\,a0 + 2\,a1\,\varepsilon)\,((x0 + X1\,\varepsilon)\,y1\,\varepsilon + (x0 + x1\,\varepsilon)\ Y1\,\varepsilon)$$
$$+ (a0 + a1\,\varepsilon)\,((x0 + x1\,\varepsilon)\,y1\,\varepsilon + (x0 + X1\,\varepsilon)\ Y1\,\varepsilon)]$$

```
> expand(Res):
> eval(%,epsilon^2=0):eval(%,epsilon^3=0):eval(%,epsilon^4=0):
> eval(%,epsilon^5=0):
> Res:=collect(%,epsilon);
```

$$Res := [(-9\,b0\,x0\,y1 - 9\,b0\,x0\ Y1 - 3\,a0\,x0^2\ Y1 - 3\,a0\,x0^2\,y1 + a0^2\ Y1 + a0^2\,y1)\,\varepsilon,$$
$$(6\,a0\,x0^3\,X1 - 3\,a0^2\,a1 - 6\,a0^2\,x0\,X1 - 6\,a0^2\,x0\,x1 - 12\,a0\,a1\,x0^2 + 27\,b0\,x0^2\,X1$$
$$+ 27\,b0\,x0^2\,x1 - 3\,a0\,b0\,X1 - 3\,a0\,b0\,x1 - 6\,a0\,b1\,x0 - 6\,a1\,b0\,x0 + 3\,a1\,x0^4$$
$$+ 18\,b1\,x0^3 - 18\,b0\,b1 + 6\,a0\,x0^3\,x1)\varepsilon - 6\,a0^2\,x0^2 - a0^3 + 3\,a0\,x0^4 - 9\,b0^2$$
$$- 6\,a0\,b0\,x0 + 18\,b0\,x0^3,$$
$$(3\,x0^3\,y1 + 3\,x0^3\ Y1 + 3\,a0\,x0\,y1 + 3\,a0\,x0\ Y1 + 3\,b0\,y1 + 3\,b0\ Y1)\,\varepsilon]$$

```
> eval(Res,x0^3=-a0*x0-b0):eval(%,x0^4=-a0*x0^2-b0*x0):
> expand(%): Res:=collect(%,epsilon);
```

$$Res := [(-9\,b0\,x0\,y1 - 9\,b0\,x0\,Y1 - 3\,a0\,x0^2\,Y1 - 3\,a0\,x0^2\,y1 + a0^2\,Y1 + a0^2\,y1)\,\varepsilon,$$
$$(-12\,a0^2\,x0\,X1 - 9\,a0\,b0\,X1 - 3\,a0^2\,a1 - 12\,a0^2\,x0\,x1 - 15\,a0\,a1\,x0^2$$
$$+ 27\,b0\,x0^2\,X1 + 27\,b0\,x0^2\,x1 - 9\,a0\,b0\,x1 - 24\,a0\,b1\,x0 - 9\,a1\,b0\,x0$$
$$-36\,b0\,b1)\varepsilon - 27\,a0\,b0\,x0 - 27\,b0^2 - 9\,a0^2\,x0^2 - a0^3,\ 0]$$

```
> resultant(9*a0^2*x0^2+27*b0^2+27*a0*b0*x0+a0^3,x0^3+a0*x0+b0,x0);
```

$$(4\,a0^3 + 27\,b0^2)^3$$

```
> coef:=-1/(9*a0^2*x0^2+27*b0^2+27*a0*b0*x0+a0^3)+epsilon*(24*a0*b1*x0+
> 36*b0*b1+3*a0^2*a1+12*a0^2*x0*X1+9*a0*b0*X1+12*a0^2*x0*x1+9*a0*b0*x1+1
> 5*a0*a1*x0^2+9*a1*b0*x0-27*b0*x0^2*X1-27*b0*x0^2*x1)/(-9*a0^2*x0^2-27*
> b0^2-27*a0*b0*x0-a0^3)^2;
```

$$coef := -\frac{1}{9\,a0^2\,x0^2 + 27\,b0^2 + 27\,a0\,b0\,x0 + a0^3} + \varepsilon(24\,a0\,b1\,x0 + 36\,b0\,b1 + 3\,a0^2\,a1$$
$$+ 12\,a0^2\,x0\,X1 + 9\,a0\,b0\,X1 + 12\,a0^2\,x0\,x1 + 9\,a0\,b0\,x1 + 15\,a0\,a1\,x0^2$$
$$+ 9\,a1\,b0\,x0 - 27\,b0\,x0^2\,X1 - 27\,b0\,x0^2\,x1)/$$
$$(-27\,a0\,b0\,x0 - 27\,b0^2 - 9\,a0^2\,x0^2 - a0^3)^2$$

```
> coef*Res:
> expand(%):eval(%,epsilon^2=0):
> normal(%):Res:=collect(%,epsilon);
```

$$Res := [-\frac{(-9\,b0\,x0\,y1 - 9\,b0\,x0\,Y1 - 3\,a0\,x0^2\,Y1 - 3\,a0\,x0^2\,y1 + a0^2\,Y1 + a0^2\,y1)\,\varepsilon}{9\,a0^2\,x0^2 + 27\,b0^2 + 27\,a0\,b0\,x0 + a0^3},\ 1,\ 0]$$

```
> resultant(a0^2-9*b0*x0-3*a0*x0^2,x0^3+a0*x0+b0,x0);
```

$$(4\,a0^3 + 27\,b0^2)^2$$

```
> expand((a0^2-9*b0*x0-3*a0*x0^2)*(y1+Y1));
> expand((a0^2-9*b0*x0-3*a0*x0^2)*(3*x0^2+a0));
```

$$-9\,b0\,x0\,y1 - 9\,b0\,x0\,Y1 - 3\,a0\,x0^2\,Y1 - 3\,a0\,x0^2\,y1 + a0^2\,Y1 + a0^2\,y1$$
$$-27\,b0\,x0^3 - 9\,a0\,b0\,x0 - 9\,a0\,x0^4 + a0^3$$

```
> eval(Res,a0^3+27*a0*b0*x0+27*b0^2+9*a0^2*x0^2=(a0^2-9*b0*x0-3*a0*x0^2
> )*(3*x0^2+a0)):
> Res:=eval(
> %,a0^2*y1+a0^2*Y1-9*b0*x0*y1-9*b0*x0*Y1-3*a0*x0^2*Y1-3*a0*x0^2*y1=(a0
> ^2-9*b0*x0-3*a0*x0^2)*(y1+Y1));
```

$$Res := [-\frac{(Y1 + y1)\,\varepsilon}{3\,x0^2 + a0},\ 1,\ 0]$$

A.5 Détails des calculs du lemme 2.3.4

```
> eval([P1,Q1,R1],x=x0+x1*epsilon):eval(%,y=y0+y1*epsilon):
> eval(%,z=1):
> eval(%,X=x0+X1*epsilon):eval(%,Y=-y0+Y1*epsilon):eval(%,Z=1):
> eval(%,a=a0+a1*epsilon):Res:=eval(%,b=b0+b1*epsilon);
```

$$
\begin{aligned}
Res := & [(y0 + y1\,\varepsilon)^2\,(x0 + X1\,\varepsilon) - (x0 + x1\,\varepsilon)\,(-y0 + Y1\,\varepsilon)^2 \\
& - (a0 + a1\,\varepsilon)\,(2\,x0 + X1\,\varepsilon + x1\,\varepsilon)\,(X1\,\varepsilon - x1\,\varepsilon) \\
& + (2\,(y0 + y1\,\varepsilon)\,(-y0 + Y1\,\varepsilon) - 3\,b0 - 3\,b1\,\varepsilon)\,(X1\,\varepsilon - x1\,\varepsilon), \\
& (y0 + y1\,\varepsilon)\,(-y0 + Y1\,\varepsilon)\,(2\,y0 + y1\,\varepsilon - Y1\,\varepsilon) \\
& - (a0 + a1\,\varepsilon)\,((x0 + x1\,\varepsilon)\,(y0 + y1\,\varepsilon) - (x0 + X1\,\varepsilon)\,(-y0 + Y1\,\varepsilon)) \\
& + (-2\,a0 - 2\,a1\,\varepsilon - 3\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon)) \\
& ((x0 + X1\,\varepsilon)\,(y0 + y1\,\varepsilon) - (x0 + x1\,\varepsilon)\,(-y0 + Y1\,\varepsilon)) \\
& - 3\,(b0 + b1\,\varepsilon)\,(2\,y0 + y1\,\varepsilon - Y1\,\varepsilon), \\
& (Y1\,\varepsilon + y1\,\varepsilon)\,(2\,y0 + y1\,\varepsilon - Y1\,\varepsilon) \\
& + (3\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon) + a0 + a1\,\varepsilon)\,(X1\,\varepsilon - x1\,\varepsilon)]
\end{aligned}
$$

```
> expand(Res):
> eval(%,epsilon^2=0):eval(%,epsilon^3=0):eval(%,epsilon^4=0):
> Res:=collect(%,epsilon);
```

$$
\begin{aligned}
Res := & [(2\,y0\,y1\,x0 + 2\,x0\,y0\,Y1 - 2\,a0\,x0\,X1 + 2\,a0\,x0\,x1 - y0^2\,X1 + x1\,y0^2 \\
& - 3\,b0\,X1 + 3\,b0\,x1)\varepsilon, \\
& (3\,y0^2\,Y1 - 3\,x0^3\,y1 - 6\,a1\,x0\,y0 - 3\,a0\,x0\,y1 - 3\,a0\,x1\,y0 \\
& + 3\,x0^3\,Y1 - 3\,a0\,X1\,y0 - 9\,x0^2\,X1\,y0 - 9\,x0^2\,x1\,y0 + 3\,a0\,x0\,Y1 - 3\,y0^2\,y1 \\
& - 3\,b0\,y1 + 3\,b0\,Y1 - 6\,b1\,y0)\varepsilon - 2\,y0^3 - 6\,b0\,y0 - 6\,a0\,x0\,y0 - 6\,x0^3\,y0, \\
& (2\,y0\,Y1 + 2\,y0\,y1 + 3\,x0^2\,X1 - 3\,x0^2\,x1 + a0\,X1 - a0\,x1)\,\varepsilon]
\end{aligned}
$$

```
> eval(Res,2*y0*y1=(3*x0^2+a0)*x1+a1*x0+b1):
> eval(%,2*y0*Y1=-((3*x0^2+a0)*X1+a1*x0+b1)):
> expand(%):Res:=collect(%,epsilon);
```

$$
\begin{aligned}
Res := & [(2\,y0\,y1\,x0 + 2\,x0\,y0\,Y1 - 2\,a0\,x0\,X1 + 2\,a0\,x0\,x1 - y0^2\,X1 + x1\,y0^2 - 3\,b0\,X1 \\
& + 3\,b0\,x1)\varepsilon, (3\,y0^2\,Y1 - 3\,x0^3\,y1 - 6\,a1\,x0\,y0 - 3\,a0\,x0\,y1 - 3\,a0\,x1\,y0 \\
& + 3\,x0^3\,Y1 - 3\,a0\,X1\,y0 - 9\,x0^2\,X1\,y0 - 9\,x0^2\,x1\,y0 + 3\,a0\,x0\,Y1 - 3\,y0^2\,y1 \\
& - 3\,b0\,y1 + 3\,b0\,Y1 - 6\,b1\,y0)\varepsilon - 2\,y0^3 - 6\,b0\,y0 - 6\,a0\,x0\,y0 - 6\,x0^3\,y0,\ 0]
\end{aligned}
$$

```
> eval(Res,x0^3=y0^2-a0*x0-b0):
> expand(%):Res:=collect(%,epsilon);
```

$$
\begin{aligned}
Res := & [(2\,y0\,y1\,x0 + 2\,x0\,y0\,Y1 - 2\,a0\,x0\,X1 + 2\,a0\,x0\,x1 - y0^2\,X1 + x1\,y0^2 - 3\,b0\,X1 \\
& + 3\,b0\,x1)\varepsilon, (6\,y0^2\,Y1 - 6\,y0^2\,y1 - 6\,a1\,x0\,y0 - 3\,a0\,x1\,y0 - 3\,a0\,X1\,y0 \\
& - 9\,x0^2\,X1\,y0 - 9\,x0^2\,x1\,y0 - 6\,b1\,y0)\varepsilon - 8\,y0^3,\ 0]
\end{aligned}
$$

```
> eval(Res,y1=((3*x0^2+a0)*x1+a1*x0+b1)/(2*y0)):
> eval(%,Y1=-((3*x0^2+a0)*X1+a1*x0+b1)/(2*y0)):
> expand(%):Res:=collect(%,epsilon);
```

$$
\begin{aligned}
Res := & [(3\,x0^3\,x1 + 3\,a0\,x0\,x1 - 3\,x0^3\,X1 - 3\,a0\,x0\,X1 - y0^2\,X1 + x1\,y0^2 - 3\,b0\,X1 \\
& + 3\,b0\,x1)\varepsilon, \\
& (-18\,x0^2\,X1\,y0 - 6\,a0\,X1\,y0 - 12\,a1\,x0\,y0 - 12\,b1\,y0 - 18\,x0^2\,x1\,y0 - 6\,a0\,x1\,y0) \\
& \varepsilon - 8\,y0^3,\ 0]
\end{aligned}
$$

```
> eval(Res,x0^3= y0^2-a0*x0-b0):
> expand(%):Res:=collect(%,epsilon);
```

$$Res := [(4\,x1\,y0^2 - 4\,y0^2\,X1)\,\varepsilon,$$
$$(-18\,x0^2\,X1\,y0 - 6\,a0\,X1\,y0 - 12\,a1\,x0\,y0 - 12\,b1\,y0 - 18\,x0^2\,x1\,y0 - 6\,a0\,x1\,y0)$$
$$\varepsilon - 8\,y0^3,\ 0]$$

```
> coef:=(-1/(8*y0^3)-(-12*b1*y0-12*a1*y0*x0-6*a0*y0*x1-6*a0*X1*y0-18*x0
> ^2*y0*x1-18*x0^2*X1*y0)*epsilon/((8*y0^3)^2));
```

$$coef := -\frac{1}{8\,y0^3} - \frac{(-18\,x0^2\,X1\,y0 - 6\,a0\,X1\,y0 - 12\,a1\,x0\,y0 - 12\,b1\,y0 - 18\,x0^2\,x1\,y0 - 6\,a0\,x1\,y0)\,\varepsilon}{64\,y0^6}$$

```
> coef*Res:
> expand(%):eval(%,epsilon^2=0):
> normal(%):Res:=collect(%,epsilon);
```

$$Res := [-\frac{\varepsilon\,(x1 - X1)}{2\,y0},\ 1,\ 0]$$

A.6 Détails des calculs du lemme 2.3.5

```
> eval(2*y0*[P2,Q2,R2],x=x0+x1*epsilon):eval(%,y=y0+y1*epsilon):
> eval(%,z=1):
> eval(%,X=x0+X1*epsilon):eval(%,Y=y0+Y1*epsilon):eval(%,Z=1):
> eval(%,a=a0+a1*epsilon):Res1:=eval(%,b=b0+b1*epsilon);
```

$$Res1 := 2\,y0[((y0 + y1\,\varepsilon)\,(y0 + Y1\,\varepsilon) - 6\,b0 - 6\,b1\,\varepsilon)$$
$$((x0 + X1\,\varepsilon)\,(y0 + y1\,\varepsilon) + (x0 + x1\,\varepsilon)\,(y0 + Y1\,\varepsilon))$$
$$+ ((a0 + a1\,\varepsilon)^2 - 2\,(a0 + a1\,\varepsilon)\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon))\,(2\,y0 + Y1\,\varepsilon + y1\,\varepsilon)$$
$$- 3\,(b0 + b1\,\varepsilon)\,((x0 + x1\,\varepsilon)\,(y0 + y1\,\varepsilon) + (x0 + X1\,\varepsilon)\,(y0 + Y1\,\varepsilon))$$
$$- (a0 + a1\,\varepsilon)\,((y0 + y1\,\varepsilon)\,(x0 + X1\,\varepsilon)^2 + (x0 + x1\,\varepsilon)^2\,(y0 + Y1\,\varepsilon)),$$
$$(y0 + y1\,\varepsilon)^2\,(y0 + Y1\,\varepsilon)^2 + 3\,(a0 + a1\,\varepsilon)\,(x0 + x1\,\varepsilon)^2\,(x0 + X1\,\varepsilon)^2 - (a0 + a1\,\varepsilon)^3$$
$$- 9\,(b0 + b1\,\varepsilon)^2 - (a0 + a1\,\varepsilon)^2\,(2\,x0 + X1\,\varepsilon + x1\,\varepsilon)^2$$
$$- 2\,(a0 + a1\,\varepsilon)^2\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon)$$
$$+ (9\,(b0 + b1\,\varepsilon)\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon) - 3\,(a0 + a1\,\varepsilon)\,(b0 + b1\,\varepsilon))$$
$$(2\,x0 + X1\,\varepsilon + x1\,\varepsilon),$$
$$((y0 + y1\,\varepsilon)\,(y0 + Y1\,\varepsilon) + 3\,b0 + 3\,b1\,\varepsilon)\,(2\,y0 + Y1\,\varepsilon + y1\,\varepsilon)$$
$$+ (3\,(x0 + x1\,\varepsilon)\,(x0 + X1\,\varepsilon) + 2\,a0 + 2\,a1\,\varepsilon)$$
$$((x0 + X1\,\varepsilon)\,(y0 + y1\,\varepsilon) + (x0 + x1\,\varepsilon)\,(y0 + Y1\,\varepsilon))$$
$$+ (a0 + a1\,\varepsilon)\,((x0 + x1\,\varepsilon)\,(y0 + y1\,\varepsilon) + (x0 + X1\,\varepsilon)\,(y0 + Y1\,\varepsilon))]$$

```
> expand(Res1):
> eval(%,epsilon^2=0):eval(%,epsilon^3=0):eval(%,epsilon^4=0):
> eval(%,epsilon^5=0):
> Res1:=collect(%,epsilon);
```

$$Res1 := [(6\,y0^3\,x0\,y1 + 6\,y0^3\,x0\,Y1 - 18\,b0\,X1\,y0^2 - 18\,b0\,x1\,y0^2 + 8\,a0\,a1\,y0^2 - 36\,b1\,x0\,y0^2 + 2\,a0^2\,Y1\,y0 + 2\,a0^2\,y1\,y0 + 2\,y0^4\,x1 - 12\,a1\,x0^2\,y0^2 + 2\,y0^4\,X1 - 12\,a0\,x0\,X1\,y0^2 - 12\,a0\,x0\,x1\,y0^2 - 18\,b0\,x0\,y1\,y0 - 18\,b0\,x0\,Y1\,y0 - 6\,a0\,x0^2\,Y1\,y0 - 6\,a0\,x0^2\,y1\,y0)\varepsilon - 36\,b0\,x0\,y0^2 + 4\,a0^2\,y0^2 - 12\,a0\,x0^2\,y0^2 + 4\,y0^4\,x0, (4\,y0^4\,Y1 + 36\,b1\,x0^3\,y0 - 6\,a0^2\,a1\,y0 - 36\,b0\,b1\,y0 + 12\,a0\,x0^3\,X1\,y0 + 4\,y0^4\,y1 + 12\,a0\,x0^3\,x1\,y0 - 12\,a0^2\,x0\,X1\,y0 - 12\,a0^2\,x0\,x1\,y0 + 6\,a1\,x0^4\,y0 - 24\,a0\,a1\,x0^2\,y0 + 54\,b0\,x0^2\,X1\,y0 + 54\,b0\,x0^2\,x1\,y0 - 6\,a0\,b0\,X1\,y0 - 6\,a0\,b0\,x1\,y0 - 12\,a0\,b1\,x0\,y0 - 12\,a1\,b0\,x0\,y0)\varepsilon - 12\,a0^2\,x0^2\,y0 + 6\,a0\,x0^4\,y0 + 2\,y0^5 - 12\,a0\,b0\,x0\,y0 - 2\,a0^3\,y0 - 18\,b0^2\,y0 + 36\,b0\,x0^3\,y0, (12\,a1\,x0\,y0^2 + 6\,y0^3\,Y1 + 6\,a0\,X1\,y0^2 + 18\,x0^2\,x1\,y0^2 + 6\,a0\,x1\,y0^2 + 6\,x0^3\,Y1\,y0 + 18\,x0^2\,X1\,y0^2 + 6\,b0\,Y1\,y0 + 6\,x0^3\,y1\,y0 + 12\,b1\,y0^2 + 6\,b0\,y1\,y0 + 6\,y0^3\,y1 + 6\,a0\,x0\,y1\,y0 + 6\,a0\,x0\,Y1\,y0)\varepsilon + 12\,a0\,x0\,y0^2 + 4\,y0^4 + 12\,b0\,y0^2 + 12\,x0^3\,y0^2]$$

```
> eval(Res1,x0^3=y0^2-a0*x0-b0):
> eval(%,y1=((3*x0^2+a0)*x1+a1*x0+b1)/(2*y0)):
> eval(%,Y1=((3*x0^2+a0)*X1+a1*x0+b1)/(2*y0)):
> expand(%):Res1:=collect(%,epsilon);
```

$$Res1 := [(9\,y0^2\,x0^3\,X1 - 6\,a0\,x0^2\,b1 - 30\,b1\,x0\,y0^2 - 18\,b0\,X1\,y0^2 - 18\,b0\,x1\,y0^2 + 8\,a0\,a1\,y0^2 - 6\,a1\,x0^2\,y0^2 + 2\,y0^4\,X1 + 2\,y0^4\,x1 - 9\,a0\,x0\,X1\,y0^2 - 9\,a0\,x0\,x1\,y0^2 - 9\,b0\,x0\,a0\,x1 + a0^3\,x1 + 2\,a0^2\,b1 + 9\,y0^2\,x0^3\,x1 + 2\,a0^2\,a1\,x0 - 27\,b0\,x0^3\,x1 - 18\,b0\,x0^2\,a1 - 18\,b0\,x0\,b1 - 9\,a0\,x0^4\,x1 - 6\,a0\,x0^3\,a1 - 9\,b0\,X1\,a0\,x0 - 27\,b0\,X1\,x0^3 - 9\,X1\,a0\,x0^4 + a0^3\,X1)\varepsilon - 36\,b0\,x0\,y0^2 + 4\,a0^2\,y0^2 - 12\,a0\,x0^2\,y0^2 + 4\,y0^4\,x0, (-24\,a0^2\,x0\,X1\,y0 - 24\,a0^2\,x0\,x1\,y0 - 24\,a0\,a1\,x0^2\,y0 + 54\,b0\,x0^2\,X1\,y0 + 54\,b0\,x0^2\,x1\,y0 - 18\,a0\,b0\,X1\,y0 - 18\,a0\,b0\,x1\,y0 - 48\,a0\,b1\,x0\,y0 - 12\,a1\,b0\,x0\,y0 - 6\,a0^2\,a1\,y0 - 72\,b0\,b1\,y0 + 6\,a1\,x0^4\,y0 + 40\,b1\,y0^3 + 14\,a0\,X1\,y0^3 + 6\,y0^3\,x0^2\,x1 + 14\,y0^3\,a0\,x1 + 4\,y0^3\,a1\,x0 + 6\,y0^3\,x0^2\,X1)\varepsilon - 12\,a0^2\,x0^2\,y0 + 6\,a0\,x0^4\,y0 + 2\,y0^5 - 2\,a0^3\,y0 - 54\,b0^2\,y0 - 48\,a0\,b0\,x0\,y0 + 36\,b0\,y0^3, (24\,b1\,y0^2 + 36\,x0^2\,x1\,y0^2 + 36\,x0^2\,X1\,y0^2 + 12\,a0\,x1\,y0^2 + 12\,a0\,X1\,y0^2 + 24\,a1\,x0\,y0^2)\varepsilon + 16\,y0^4]$$

```
> coef:=16*y0^4+12*y0^2*(2*b1+(3*x0^2+a0)*(x1+X1)+2*a1*x0)*epsilon;
```

$$coef := 16\,y0^4 + 12\,y0^2\,(2\,b1 + (3\,x0^2 + a0)\,(x1 + X1) + 2\,a1\,x0)\,\varepsilon$$

```
> alpha:=(3*x0^2+a0)/(2*y0)+epsilon*((a1+3*x0*(x1+X1)-(y1+Y1)*
> (3*x0^2+a0)/(2*y0))*1/(2*y0));
```

$$\alpha := \frac{3\,x0^2 + a0}{2\,y0} + \frac{\varepsilon\,(a1 + 3\,x0\,(x1 + X1) - \frac{(Y1 + y1)\,(3\,x0^2 + a0)}{2\,y0})}{2\,y0}$$

```
> XX:=alpha^2-2*x0-(x1+X1)*epsilon;
```

$$XX := \left(\frac{3\,x0^2 + a0}{2\,y0} + \frac{\varepsilon\,(a1 + 3\,x0\,(x1 + X1) - \frac{(Y1 + y1)\,(3\,x0^2 + a0)}{2\,y0})}{2\,y0}\right)^2 - 2\,x0 - (x1 + X1)\,\varepsilon$$

```
> YY:=alpha*(x0+x1*epsilon-XX)-y0-y1*epsilon;
```

$$YY := \left(\frac{3\,x0^2+a0}{2\,y0}+\frac{\varepsilon\,(a1+3\,x0\,(x1+X1)-\frac{(Y1+y1)\,(3\,x0^2+a0)}{2\,y0})}{2\,y0}\right)\left(3\,x0+x1\,\varepsilon\right.$$
$$\left.-\left(\frac{3\,x0^2+a0}{2\,y0}+\frac{\varepsilon\,(a1+3\,x0\,(x1+X1)-\frac{(Y1+y1)\,(3\,x0^2+a0)}{2\,y0})}{2\,y0}\right)^2+(x1+X1)\,\varepsilon\right)$$
$$-y0-y1\,\varepsilon$$

```
> coef*[XX,YY,1]:
> expand(%):
> eval(%,epsilon^2=0):eval(%,epsilon^3=0):eval(%,epsilon^4=0):
> eval(%,epsilon^5=0):eval(%,epsilon^6=0):
> eval(%,y1=((3*x0^2+a0)*x1+a1*x0+b1)/(2*y0)):
> eval(%,Y1=((3*x0^2+a0)*X1+a1*x0+b1)/(2*y0)):
> expand(%):
> Res2:=collect(%,epsilon);
```

$$\begin{aligned}Res2 := [&(-16\,y0^4\,X1-24\,a1\,x0^2\,y0^2-16\,y0^4\,x1+8\,a0\,a1\,y0^2+2\,a0^2\,a1\,x0\\&+27\,a0\,x0^4\,x1-48\,b1\,x0\,y0^2+a0^3\,x1+2\,a0^2\,b1+12\,a0\,x0^2\,b1+12\,a0\,x0^3\,a1\\&+18\,x0^4\,b1+27\,x0^6\,x1+18\,x0^5\,a1+9\,a0^2\,x0^2\,x1+27\,X1\,a0\,x0^4+9\,X1\,a0^2\,x0^2\\&+27\,X1\,x0^6+a0^3\,X1)\varepsilon+24\,a0\,x0^2\,y0^2-32\,y0^4\,x0+4\,a0^2\,y0^2+36\,y0^2\,x0^4,\\&(-6\,a0^2\,a1\,y0+72\,b1\,x0^3\,y0-36\,a0\,x0^3\,x1\,y0-6\,a0^2\,x0\,X1\,y0-6\,a0^2\,x0\,x1\,y0\\&-12\,a0\,a1\,x0^2\,y0+24\,a0\,b1\,x0\,y0-4\,a0\,X1\,y0^3+18\,a1\,x0^4\,y0-4\,y0^3\,a0\,x1\\&-8\,y0^3\,a1\,x0-32\,b1\,y0^3+60\,y0^3\,x0^2\,x1-36\,a0\,x0^3\,X1\,y0-54\,y0\,x0^5\,x1\\&+60\,y0^3\,x0^2\,X1-54\,x0^5\,X1\,y0)\varepsilon-18\,a0^2\,x0^2\,y0+24\,y0^3\,a0\,x0-54\,a0\,x0^4\,y0\\&-16\,y0^5-54\,y0\,x0^6-2\,a0^3\,y0+72\,y0^3\,x0^3,\\&(24\,b1\,y0^2+36\,x0^2\,x1\,y0^2\\&+36\,x0^2\,X1\,y0^2+12\,a0\,x1\,y0^2+12\,a0\,X1\,y0^2\\&+24\,a1\,x0\,y0^2)\varepsilon+16\,y0^4]\end{aligned}$$

```
> Res1-Res2:
> eval(%,y0^5=y0*(x0^3+a0*x0+b0)^2):eval(%,y0^4=y0^2*(x0^3+a0*x0+b0)):
> eval(%,y0^3=y0*(x0^3+a0*x0+b0)):eval(%,y0^2=x0^3+a0*x0+b0):
> expand(%);
```

$$[0,\ 0,\ 0]$$

A.7 Détails des calculs du lemme 2.3.6

```
> alpha:=(Y-y)/(X-x);
```

$$\alpha := \frac{Y-y}{X-x}$$

```
> XX:=alpha^2-x-X; YY:=alpha*(x-XX)-y;
```

$$XX := \frac{(Y-y)^2}{(X-x)^2}-x-X$$

$$YY := \frac{(Y-y)\,(2\,x - \dfrac{(Y-y)^2}{(X-x)^2} + X)}{X-x} - y$$

```
> Res1:=[-(X-x)^3*normal(XX),-(X-x)^3*normal(YY),-(X-x)^3];
```

$$Res1 := [-(X-x)\,(Y^2 - 2\,y\,Y + y^2 + x\,X^2 + x^2\,X - x^3 - X^3),$$
$$3\,Y\,x^2\,X - 2\,Y\,x^3 + Y^3 - 3\,y\,Y^2 + 3\,Y\,y^2 - Y\,X^3 + y\,x^3 - y^3 + 2\,y\,X^3 - 3\,y\,x\,X^2,$$
$$-(X-x)^3]$$

```
> expand(Res1):
> eval(%,Y^2=X^3+a*X+b):eval(%,y^2=x^3+a*x+b):
> eval(%,y^3=y*(x^3+a*x+b)):eval(%,Y^3=Y*(X^3+a*X+b)):
> Res1:=expand(%);
```

$$Res1 := [-a\,X^2 - 2\,X\,b + 2\,X\,y\,Y + X\,x^3 - x\,X^3 + 2\,x\,b - 2\,x\,y\,Y + a\,x^2,$$
$$3\,Y\,x^2\,X + Y\,x^3 + Y\,a\,X + 4\,Y\,b - y\,X^3 - 3\,y\,a\,X - 4\,y\,b + 3\,Y\,a\,x - y\,a\,x - 3\,y\,x\,X^2,$$
$$-X^3 + 3\,x\,X^2 - 3\,x^2\,X + x^3]$$

```
> eval([P1,Q1,R1],z=1):eval(%,Z=1):
> expand(%):
> eval(%,y^2=x^3+a*x+b):eval(%,Y^2=X^3+a*X+b):
> Res2:=expand(%);
```

$$Res2 := [-a\,X^2 - 2\,X\,b + 2\,X\,y\,Y + X\,x^3 - x\,X^3 + 2\,x\,b - 2\,x\,y\,Y + a\,x^2,$$
$$3\,Y\,x^2\,X + Y\,x^3 + Y\,a\,X + 4\,Y\,b - y\,X^3 - 3\,y\,a\,X - 4\,y\,b + 3\,Y\,a\,x - y\,a\,x - 3\,y\,x\,X^2,$$
$$-X^3 + 3\,x\,X^2 - 3\,x^2\,X + x^3]$$

```
> Res2-Res1;
```

$$[0,\ 0,\ 0]$$

Annexe B

Exemples

B.1 Deux procédures écrites en maple

La procédure `GroupeSurFp` est écrite en language maple comme suit :

```
> GroupeSurFp:=proc(a,b,p)
> local X,Y0,G,i,k;
> if (4*a^3+27*b^2 mod p)=0  then  RETURN({}); fi;
> G:={o};
> for i from 0 to p-1 do
> X:= i^3+a*i+b mod p;
> Y0:=Roots(x^2-X) mod p;
> if nops(Y0)<>0 then Y0:=Roots(x^2-X) mod p;
> Y0:=Y0[1][1];
> G:=G union {[i,Y0], [i,-Y0 mod p]}; fi;
> od;
> RETURN(G);
> end:
```

Cette procédure :

- prend en entrée un nombre premier `p`, ainsi que deux éléments `a` et `b` entiers compris entre 0 et `p`-1 ;
- renvoie :
  - l'ensemble vide si la courbe projective d'équation $y^2z = x^3 + \mathtt{a}xz^2 + \mathtt{b}z^3$ est singulière ;
  - l'ensemble des éléments du groupe $E_{\mathtt{a},\mathtt{b}}(\mathbb{F}_{\mathtt{p}}[\varepsilon])$, sinon.

Autrement dit, elle renvoie un ensemble non vide seulement si l'équation de Weierstrass $y^2 = x^3 + \mathtt{a}x + \mathtt{b}$ définit bien une courbe elliptique.

La procédure `OrdreDansF` :

- prend en entrée un nombre premier `p`,
- renvoie un tableau à double entrée dont les lignes et les colonnes peuvent être numérotées de 0 à $\mathtt{p} - 1$ et dont le coefficient $c_{i,j}$ contient le cardinal de $E_{i,j}(\mathbb{F}_{\mathtt{p}})$.

Nous l'avons écrite en language maple comme suit :

```
> OrdreDansF:=proc(p)
> local G,i,j;
> G:=array(1..p,1..p):
> for i from 0 to p-1 do
> for j from 0 to p-1 do
> G[i+1,j+1]:=nops(GroupeSurFp(i,j,p));;
> print(G);
> end:
```

B.2 Ordre de toutes les courbes elliptiques de $\mathbb{F}_5$

Les procédures écrites en B.1 permet de connaître l'ordre de toutes les courbes elliptiques définies sur $\mathbb{F}_p$ par une équation du type $y^2 = x^3 + ax + b$.
Le tableau suivant donne le cardinal de $E_{i,j}(\mathbb{F}_5)$ où i et j sont respectivement les numéros de ligne et de colonne numérotées de 0 à 4. Par exemple, on lit que la courbe d'équation $y^2 = x^3 + x + 3$ est singulière. Aussi, on lit : $Card(E_{3,0}(\mathbb{F}_5)) = 10$.

```
> OrdreDansF(5);
```

$$\begin{bmatrix} 0 & 6 & 6 & 6 & 6 \\ 4 & 9 & 4 & 4 & 9 \\ 2 & 7 & 0 & 0 & 7 \\ 10 & 0 & 5 & 5 & 0 \\ 8 & 8 & 3 & 3 & 8 \end{bmatrix}$$

On peut alors donner la liste exhaustive des couples (a, b) de $\mathbb{F}_5^2$ tels que la courbe d'équation $y^2 = x^3 + ax + b$ soit singulière :

$$(0,0), (1,3), (4,3), (2,2), (3,2).$$

Ce sont les solutions de l'équation à deux inconnues définie sur $\mathbb{F}_5$ par : $4a^3 + 27b^2 = 0$.

Index

Bibliographie

[AS98] K. Araki and T. Satoh. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Math. Univ. St. Pauli*, 47 :81–92, 1998.

[Atk88] A.O.L. Atkin. The number of points an an elliptic curve modulo a prime. manuscript, 1988. Chicago.

[BBFK05] F. Bahr, M. Boehm, J. Franke, and T. Kleinjung. Annonce de résolution du problème RSA-640, novembre 2005. Annonce disponible á http ://www.crypto-world.com/announcements/rsa640.txt.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98*, volume 1462 of *Advances in Cryptology*, pages 26–45. Springer Verlag, 1998.

[Bel98] M. Bellare. Practice-oriented provable-security. In *First International Workshop on Information Security (ISW 97)*, volume 1396 of *LNCS*. Springer Verlag, 1998.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures : How to sign with rsa and rabin. In *Eurocrypt '96*, Advances in Cryptology, 1996.

[Bra79] G. Brassard. A note on the complexity of cryptography. In *IEEE Transactions on Information Theory*, volume IT-25, pages 232–233. Mars 1979.

[BSS99] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Number 265 in London Mathematic Society Lecture Notes Series. Cambridge University Press, 1999.

[BSS05] I. Blake, G. Seroussi, and N. Smart. *Advances in Elliptic Curves in Cryptography*. Number 317 in London Mathematic Society Lecture Notes Series. Cambridge University Press, 2005.

[CC87] D.V. Chudnovsky and G.V. Chudnovsky. Sequences of numbers generated by addition in formal group and new primality and factorization tests. *Adv. in Appl. Math.*, (7) :385–434, 1987.

[CJLM06] M. Ciet, M. Joye, K. Lauter, and P.L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. *Designs, Codes and Cryptography*, 39(2) :189–206, mai 2006.

[CLR94] T. Cormen, C. Leiserson, and R. Rivest. *Introduction à l'algorithmique*. 2e Cycle Universitaire / Ecoles d'ingénieurs. DUNOD, 1994.

[CMO98] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology, ASIACRYPT 98*, volume LNCS 1514, pages 51–65. Springer-Verlag, 1998.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume IT-22, pages 644–654. Nov. 1976. Disponible à http ://www-ee.stanford.edu/ hellman/publications.html.

[Dub00] G. Dubertret. *Initiation à la cryptographie*. vuibert, 2 edition, 2000.

[Elg85] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, (31) :469–472, 1985.

[Elk92] N. D. Elkies. Explicit isogenies. manuscript, 1992. Boston MA.

[Elk98] N. D. Elkies. Elliptic and modular curves over finite fields and related computational issues. In *Studies in Advenced Mathematics*, pages 21–76. American mathematical society and international press edition, 1998. Based on a talk given at the conference in honor of A.O.L. Atkin.

[Fou01] M. Fouquet. *Anneau d'endomorphismes et cardinalité des courbes elliptiques : aspects algorithmiques*. PhD thesis, Ecole Polytechnique, Dec. 2001.

[Gal04] D. Galindo. *Design and Analysis of Semantically Secure Public Key Encryption Schemes*. PhD thesis, Universitat Politècnica de Catalunya, may 2004.

[Gau04] P. Gaudry. Algorithmes de comptage de points d'une courbe définie sur un corps fini. Article de survol, suite à un cours donné à l'Institut Henri Poincarré, pour le trimestre "Méthodes explicites en théorie des nombres", 2004.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractibility, A guide to the Theory of NP-completeness*. W. H. Freeman and Company, 1979.

[Gjø04] K Gjøsteen. *Subgroup membership problems and public key cryptosystems*. PhD thesis, Department of Mathematical Sciences, Norwegian University of Sciences and Technology, 2004.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2) :270–299, April 1984.

[Gol99] O. Goldreich. Pseudorandomness. *Notices of the American Mathematic Society*, 46(10) :1209–1216, November 1999.

[Joh90] D. S. Johnson. *A catalog of complexity classes*, volume A of *Handbook of Theorical Computer Science*, chapter 2, pages 67–161. Elsevier, 1990.

[Joy95] M. Joye. Introduction élémentaire à la théorie des courbes elliptiques. UCL Crypto Group Technical Report Series CG-1995/1, Université Catholique de Louvain, http ://www.dice.ucl.ac.be/crypto/, 1995. disponible à http ://sciences.ows.ch/mathematiques/CourbesElliptiques.pdf.

[Kat04] J. Katz. Notes des cours "advanced topics in cryptography" transcrits par ses étudiants puis relu, 2004. disponible à http ://www.cs.umd.edu/ jkatz/.

[Ker83] A. Kerckhoffs. La cryptologie militaire. *Journal des sciences militaires*, IX :5–83, janvier et 161–191, février, 1883.

[KL07] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC Press, 2007.

[KM85] N. M. Katz and B. Mazur. *Arithmetic Moduli of Elliptic Curves*. Number 108 in Annals of Mathematics Studies. Princeton University Press, 1985.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, (48) :203–209, 1987.

[Lag05] F. Laguillaumie. *Signatures à vérification contrôlée basées sur des applications bilinéaires : conception et analyse de sécurité.* PhD thesis, Université de Caen Basse-Normandie, U.F.R. Sciences, Ecole doctorale SIMEM, juin 2005.

[Len86] H. W. Lenstra Jr. Elliptic curves and number theoretic algorithms. In *Proceedings of the INTERNATIONAL CONGRESS OF MATHEMATICIANS*, volume 1, pages 99–120, Berkeley,California,USA, Août 1986. AMS.

[Ler97] R. Lercier. *Algorithmique de courbes elliptiques dans les corps finis.* PhD thesis, Ecole polytechnique, juin 1997.

[LR85] H. Lange and W. Ruppert. Complete systems of addition laws on abelian varieties. In Springer-Verlag, editor, *Inventiones mathematicae*, volume 79, pages 603–610. 1985.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton Univ Press, 1996. A definitive source of techniques for provably random sequences.

[LV00] A. Lenstra and E. Verheul. Selecting cryptographic key size. *Public Key Cryptography*, pages 446–465, 2000.

[Mar98] Sebastià Martin. *Corbes El.liptiques modul N i Applicacions Criptografiques.* PhD thesis, Departament de Matemàtica Aplicada i Telemàtica,Universitat Politècnica de Catalunya, 1998.

[MFK94] D. Mumford, J. Fogarty, and F. Kirwan. *Geometric Invariant Theory*, volume 34 of *A Series of Modern Surveys in Mathematics.* Springer-Verlag, 3e edition, 1994.

[Mil86] V. Miller. Use of elliptic curves in cryptography. In H.C. Williams, editor, *Advances in Cryptology*, volume LNCS 218 of *CRYPTO 85*, pages 417–426. Springer-Verlag, 1986.

[MOV91] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. In ACM Press, editor, *23rd Annual ACM Symposium on Theory of Computing*, pages 80–89, 1991.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[Pap94] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, 1994.

[PH78] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. *IEEE Trans. on Info. Theory*, IT-24 :106–110, January 1978.

[Poi02a] D. Pointcheval. Asymmetric cryptography and pratical security. *Journal of Telecommunications and Information Technology*, 4 :41–56, 2002.

[Poi02b] D. Pointcheval. *Le chiffrement asymétrique et la sécurité prouvée.* PhD thesis, Université Paris VII, 2002. H.D.R.

[Poi02c] D. Pointcheval. Practical security in public-key cryptography. In Springer-Verlag, editor, *4th International Conference on Informational Security and Cryptology (ISIC '01)*, volume 2288 of *Lecture Notes in Computer Science*, pages 1–17, 2002.

[PP04] D. H. Phan and D. Pointcheval. On the security notions for public-key encryption schemes. In C. Blundo and S. Cimato, editors, *Fourth Conference*

on Security in Communication Networks '04, volume 3352 of *Lecture Notes in Computer Sciences*, pages 33–47. Springer Verlag, 2004.

[RS92] C. W. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Crypto '91*, volume 576 of *Advances in Cryptology*, pages 433–444. Springer Verlag, 1992.

[Sch85] R. Schoof. Elliptic curves over finite fields and computation of square roots mod p. *Mathematics of Computation*, 44(170) :483–494, April 1985. Errata disponible sur la page personnelle de René Schoof : http ://www.mat.uniroma2.it/ schoof/.

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. *Journal de Théorie des Nombres de Bordeaux*, 7 :219–254, 1995.

[Sem98] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. In American Mathematical Society, editor, *Mathematics of computation*, volume 67, pages 353–356. January 1998.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28 :656–715, Octobre 1949.

[Sho06] V. Shoup. Sequences of games : a tool for taming complexity in security proofs. January 2006.

[Sil85] J.H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 1985.

[Sil94] J.H. Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*, volume 151 of *Graduate Texts in Mathematics*. Springer, 1994.

[Sin99] S. Singh. *Histoire des codes secrets, De l'égypte des pharaons à l'ordinateur quantique*. LGF Livre de poche, 1999.

[Sma99] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12 :193–196, 1999.

[Ste98] J. Stern. *La science du secret*. Sciences. Odile Jacob, 1998.

[Ver06] D. Vergnaud. *Approximation diophantienne et courbes elliptiques. Protocoles asymétriques d'authentification non-transférable*. PhD thesis, Université de Caen, 2006.

[Vie03] J. Viega. Practical random number generation in software. In *19th Annual Computer Security Applications Conference*, Dec. 2003.

[Vir05] M. Virat. A cryptosystem " à la " ElGamal on an elliptic curve over $\mathbb{F}_p[\varepsilon]$. In Po-Wah Yau Christopher Wolf, Stephan Lucks, editor, *WEWoRC 2005*, volume P-74, pages 32–44. Lecture Notes in Informatics, 2005.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

Résumé.

Cette thèse a pour objectif d'étudier les applications cryptographiques des courbes elliptiques sur l'anneau $\mathbb{F}_p[\varepsilon]$, où $\mathbb{F}_p$ représente un corps fini d'ordre premier p et où ε vérifie $\varepsilon^2 = 0$.

Après avoir décrit ces courbes définies sur un anneau, nous en étudions l'aspect algorithmique en proposant des solutions concrètes d'implémentations des éléments et de la loi de groupe.

Enfin, nous illustrons leur intérêt cryptographique, en proposant :
- une attaque du problème du logarithme discret elliptique (sur un corps fini) utilisant ces courbes ;
- un cryptosystème de type ElGamal sur ces courbes, dont nous étudions les propiétés de sécurité.

Abstract.

The goal of this thesis is to study cryptographic applications of elliptic curves over the ring $\mathbb{F}_p[\varepsilon]$, with $\mathbb{F}_p$ a finite field of prime order p and with the relation $\varepsilon^2 = 0$.

In a first time, we describe these curves defined over a ring. Then, we study the algorithmic properties by proposing effective implementations for representing the elements and the group law.

Finally, we study some of their cryptographic properties, with the description of :
- an attack of the elliptic discrete logarithm problem (over a finite field) using these curves ;
- a cryptosystem "à la" ElGamal over these curves. We study its security properties.