Algorithmique des courbes elliptiques dans les corps finis

Reynald Lercier

ÉCOLE POLYTECHNIQUE

THÈSE

présentée pour obtenir le titre de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

Spécialité :

INFORMATIQUE

par

Reynald LERCIER

Sujet de la thèse :

ALGORITHMIQUE DES COURBES ELLIPTIQUES DANS LES CORPS FINIS

Soutenue le 16 juin 1997 devant le jury composé de :

M.	**Jean-Jacques QUISQUATER**	Président
M.	**Jean-Marc STEYAERT**	Directeur
Mme	**Brigitte VALLÉE**	Rapporteurs
M.	**Alfred MENEZES**	
MM.	**Jean-Marc COUVEIGNES**	Examinateurs
	François MORAIN	

LABORATOIRE D'INFORMATIQUE
Unité CNRS n°1439

THÈSE

présentée pour obtenir le titre de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

Spécialité :

INFORMATIQUE

par

Reynald LERCIER

Sujet de la thèse :

ALGORITHMIQUE DES COURBES ELLIPTIQUES DANS LES CORPS FINIS

Soutenue le 16 juin 1997 devant le jury composé de :

M.	**Jean-Jacques QUISQUATER**	Président
M.	**Jean-Marc STEYAERT**	Directeur
Mme	**Brigitte VALLÉE**	Rapporteurs
M.	**Alfred MENEZES**	
MM.	**Jean-Marc COUVEIGNES**	Examinateurs
	François MORAIN	

À mes parents

À mes professeurs

À Murielle

Remerciements

La fin d'une thèse est l'une des étapes qui marque une vie. Dans ces rares occasions, il est bon, je crois, de s'arrêter, de regarder en arrière et de se souvenir.

Mes premières pensées vont à mes professeurs. De l'école primaire jusqu'à maintenant, ils m'ont tout appris, toujours avec patience, parfois avec passion. Je n'ai pas eu l'occasion de le faire jusqu'à présent, mais je leur exprime aujourd'hui toute ma gratitude. Merci en particulier à Monsieur Sinègre, mon professeur de Mathématiques en classe de $1^{\text{ère}}$ S et terminale C qui, sans le savoir, n'est pas étranger à ma décision d'entreprendre ces études doctorales.

Je suis particulièrement reconnaissant à François Morain de m'avoir, depuis maintenant cinq ans, encadré avec une disponibilité sans faille. Je lui dois un sujet de thèse formateur et grâce à ses remarques, liées aussi bien à la programmation qu'aux aspects les plus théoriques de ce mémoire, il m'a aidé à appréhender des difficultés que je ne soupçonnais pas auparavant. Je le remercie, en ce concerne plus spécialement le chapitre 5, pour ses premiers essais en `MAPLE` et ses efforts de formalisation lors de la rédaction qui m'ont été des plus utiles.

Une grande partie de ces travaux n'aurait pas pu avoir lieu sans l'aide de Jean Marc Couveignes et je lui adresse mes plus sincères remerciements. Bénéficier de ses précieuses indications et de ses longues explications fut pour moi une source d'enrichissement inestimable.

Un grand merci à Jean Marc Steyaert qui me fait l'honneur d'être mon directeur de thèse et qui a toujours su prêter une oreille bienveillante à mes préoccupations les plus diverses.

J'adresse mes remerciements aux professeurs Brigitte Vallée et Alfred Menezes pour leur difficile travail de rapporteur.

Je suis redevable à la Délégation Générale pour l'Armement de m'avoir permis d'effectuer ces travaux de recherche. Merci, d'une part, à Alain Quenzer et Alain Lanusse pour leur gestion des optionnaires Recherche. Merci, d'autre part, à la direction du Centre d'Électronique de l'Armemement qui a accepté que je réalise cette thèse au LIX tout en étant en poste à Bruz. En particulier, merci à Antoine Joux pour son rôle en tant que parrain DGA.

Je remercie au passage tous ceux qui font du LIX un endroit où les conditions de travail sont exceptionnelles. Merci à Michel Weinfeld pour sa largesse d'esprit et sa compétence. Merci à Évelyne Raissac qui supporte sur ses frêles épaules les problèmes administratifs les plus lourds. Merci à Catherine Bensoussan pour sa gentillesse. Merci enfin à Philippe Chassignet, à Éric Delaunay et aux ingénieurs système qui se sont succédé au laboratoire, notamment Thierry Besançon, Laurent Amon, Damien Doligez, Francis Gauché et Jean

Marc Vinet. Sans eux, rien n'est possible.

Je dois beaucoup à ceux qui ont partagé mon bureau, en particulier à Thierry Saura pour sa bonne humeur, à Fabrice Lefebvre pour la maintenance experte de mon PC et plus récemment à Pierrick Gaudry pour ses commentaires éclairés et la relecture de certaines parties de ce mémoire.

J'aimerais tout spécialement remercier Florent Chabaud pour ses encouragements et son amitié. Qu'il sache que je suis heureux que notre complicité ait donné vie à `ZEN`.

Je tiens aussi à remercier le professeur Buchmann et Thomas Setz pour m'avoir accueilli chaleureusement pendant une semaine à Saarbrücken. L'environnement de programmation de `ZEN` hérite en grande part de celui utilisé pour la librairie `LiPS`. Merci aussi à Philippe Toffin de m'avoir communiqué le code de routines pour manipuler des courbes elliptiques en `AXIOM`.

Les résultats obtenus dans cette thèse n'auraient pu l'être sans l'accès à un nombre parfois important de machines. Ainsi, je remercie Gérard Guillerm pour le SITX, Olivier Perret pour l'ENSTA et Thierry Besançon pour le Laboratoire de Physique Statistique de l'ENS pour l'aide qu'ils m'ont apportée.

Merci à Murielle de partager quotidiennement mes joies et mes peines.

Table des matières

Introduction **5**

I Algorithme SEA **11**

1 Corps finis **13**
1.1 Définition . . . 13
1.2 Cardinalité . . . 14
1.3 Groupe multiplicatif . . . 14
1.4 Sous-corps d'un corps fini . . . 15
1.5 Automorphismes d'un corps fini . . . 15
1.6 Existence . . . 16

2 Courbes elliptiques définies sur des corps finis **17**
2.1 Généralités . . . 17
2.1.1 Définition . . . 17
2.1.2 Loi de groupe . . . 18
2.1.3 Morphismes . . . 19
2.1.4 Spécialisation à $\mathbb{C}$ . . . 21
2.1.5 Spécialisation aux corps finis . . . 22
2.2 Courbes particulières . . . 23
2.2.1 Cas $j_E = 0$ . . . 24
2.2.2 Cas $j_E = 1728$ . . . 25
2.2.3 Autres invariants supersinguliers . . . 26

3 Algorithme de Schoof **29**
3.1 Algorithme "Pas de bébé et pas de géant" . . . 30
3.1.1 Algorithme . . . 30
3.1.2 Exemple . . . 30
3.2 Algorithme original de Schoof . . . 30
3.2.1 Algorithme . . . 31
3.2.2 Exemple . . . 32
3.3 Premières optimisations d'Atkin . . . 32
3.3.1 Action du Frobenius sur $E[\ell]$ . . . 32
3.3.2 Équations modulaires . . . 34
3.3.3 Algorithme . . . 38
3.3.4 Exemple . . . 39
3.4 Idées d'Elkies et ses développements . . . 39
3.4.1 Principe . . . 39
3.4.2 Courbes isogènes et isogénies . . . 41
3.4.3 Cycle d'isogénies . . . 45
3.4.4 Exemple . . . 48

II Calculs d'isogénie entre courbes elliptiques 51

4 Algorithmes en grande caractéristique 53
4.1 Formules de Vélu . . . 53
4.1.1 Corps algébriquement clos . . . 53
4.1.2 Reformulation dans un corps fini . . . 54
4.1.3 Isogénies de degré 2 sur un corps fini . . . 55
4.2 Calcul d'isogénie en grande caractéristique . . . 57
4.2.1 Cas de $\mathbb{C}$ . . . 57
4.2.2 Application aux corps finis . . . 59
4.3 Résultats . . . 60

5 Premier algorithme de Couveignes en petite caractéristique 61
5.1 Prérequis . . . 61
5.1.1 Points de p-torsion . . . 62
5.1.2 Groupe formel . . . 63
5.2 Description théorique de l'algorithme . . . 67
5.2.1 Morphismes de groupe formel . . . 68
5.2.2 Conditions nécessaires vérifiées par un morphisme . . . 68
5.2.3 Énumérations . . . 70
5.3 Mise en œuvre pratique de l'algorithme . . . 71
5.3.1 Calculs incrémentaux avec des séries tronquées . . . 71
5.3.2 Programmation efficace de l'algorithme . . . 74
5.4 Résultats . . . 79
5.4.1 Exemples . . . 79
5.4.2 Statistiques . . . 82

6 Deuxième algorithme de Couveignes en petite caractéristique 85
6.1 Principe de l'algorithme . . . 85
6.1.1 Propriétés des points de p^k-torsion . . . 86
6.1.2 Algorithme . . . 86
6.2 Points de p^k-torsion . . . 87
6.2.1 Constructions de $E_a[p^k]$ . . . 88
6.2.2 Algorithme "naturel" . . . 90
6.2.3 Deuxième algorithme . . . 92
6.2.4 Efficacité des deux approches . . . 94
6.3 Isomorphismes entre $E_a[p^k]$ et $E_b[p^k]$ . . . 94
6.3.1 Isomorphismes entre extensions d'Artin-Shreier . . . 94
6.3.2 Résoudre $X^p - X = \delta$ dans une tour d'extensions d'Artin-Schreier . . . 95
6.4 Résultats . . . 98
6.4.1 Exemple . . . 98
6.4.2 Quelques temps de calcul . . . 101

7 Algorithme en caractéristique deux 103
7.1 Isogénies définies sur $\mathbb{F}_{2^n}$ . . . 103
7.1.1 Point de 2-torsion . . . 103
7.1.2 Caractérisation des isogénies . . . 104
7.2 Algorithmes . . . 107
7.2.1 Système linéaire défini sur $\mathbb{F}_{2^n}$ . . . 107
7.2.2 Système quadratique défini sur $\mathbb{F}_2$ . . . 108
7.2.3 Améliorations pratiques . . . 112
7.3 Résultats . . . 114

8 Combinaison d'algorithmes en caractéristique impaire **117**
8.1 Isogénies et points de 2-torsion . . . 117
8.1.1 Caractérisation . . . 118
8.1.2 Application . . . 119
8.2 Connexion entre petite et grande caractéristique . . . 122
8.2.1 Principe . . . 123
8.2.2 Algorithme générique . . . 123
8.2.3 Exploiter les formules de Vélu . . . 123
8.3 Exemple . . . 124

III Implantation efficace et résultats **127**

9 Plus grand diviseur commun de deux entiers **129**
9.1 Algorithmes de pgcd simples . . . 130
9.1.1 pgcd euclidiens . . . 130
9.1.2 pgcd binaires . . . 137
9.1.3 Bilan . . . 144
9.2 Algorithmes de pgcd étendus . . . 144
9.2.1 Algorithme de Lehmer . . . 145
9.2.2 Algorithmes binaire et plus-minus . . . 146
9.2.3 Algorithme binaire généralisé . . . 147
9.3 Résultats . . . 149
9.3.1 pgcd simples . . . 149
9.3.2 pgcd étendus . . . 152

10 Corps finis et programmation **155**
10.1 Librairie de programmation ZEN . . . 155
10.1.1 Trois principes fondateurs . . . 156
10.1.2 Utilisation . . . 160
10.2 Exponentiation . . . 163
10.2.1 Chaîne d'addition . . . 164
10.2.2 Chaîne d'addition-soustraction . . . 166
10.2.3 Résultats . . . 168

11 Quelques perfectionnements à l'algorithme SEA **171**
11.1 Optimisations de l'algorithme de Schoof. . . . 171
11.1.1 Factorisation des équations modulaires . . . 171
11.1.2 Algorithme d'Elkies . . . 172
11.1.3 Algorithme de Schoof . . . 173
11.2 Algorithmes "tri-recherche" . . . 174
11.2.1 Préparation des données . . . 175
11.2.2 Algorithme d'Atkin . . . 177
11.2.3 Variante du "tri-recherche" d'Atkin . . . 179

12 Programmation de l'algorithme SEA et résultats **183**
12.1 Implantation . . . 183
12.1.1 Algorithmes utilisés . . . 184
12.1.2 Stratégie statique . . . 184
12.1.3 Stratégie dynamique . . . 186
12.2 Résultats . . . 187
12.2.1 Corps de taille moyenne . . . 187
12.2.2 Corps de grande taille . . . 188
12.3 Application à la cryptographie . . . 189

12.3.1 Stratégie “Early abort” 190
12.3.2 Résultats 191

A Exemple d’exécution du programme SEA **193**

Bibliographie **205**

Introduction

L'intérêt suscité par les systèmes de chiffrement à clef publique, par exemple le système RSA[2] pour ne citer que le plus connu, fut le point de départ d'un nouvel engouement pour la théorie des nombres et l'arithmétique dans ses aspects calculatoires. Dans ce domaine où la construction explicite sur ordinateur "d'objets mathématiques abstraits" joue un rôle prépondérant, les résultats obtenus peuvent servir aussi bien à mettre en œuvre de nouveaux procédés qu'à montrer les faiblesses d'anciens schémas.

Les courbes elliptiques [111] définies sur des corps finis sont un exemple parmi d'autres de ces objets. Elles ont des applications aussi bien pour construire de grands nombres premiers [6], pour trouver de petits facteurs premiers d'un nombre de taille arbitraire [64] que pour permettre la construction de schémas cryptographiques très sûrs. Pour cette dernière application, le calcul du nombre de points de ces courbes est une étape incontournable. Or, seule l'utilisation de courbes elliptiques d'un type particulier était dans un premier temps possible. Il s'agit essentiellement des courbes à multiplication complexe par un ordre dans un corps quadratique imaginaire de nombre de classes petit [89, 59, 83, 84, 61, 24] et des courbes supersingulières [82, 57, 53, 8, 78]. Cependant, ces dernières se sont avéré désastreuses [77] car le problème du logarithme discret y est plus facile.

La détermination effective du nombre de points d'une courbe elliptique quelconque auquel ce mémoire est consacré n'est possible que depuis peu. Le premier algorithme à complexité polynomiale en la taille du corps pour obtenir ce nombre est dû à R. Schoof [101] et tire bénéfice des propriétés de l'anneau des endomorphismes de la courbe. Très vite, il est apparu que cette méthode était inefficace puisque seuls les corps de moins de 10^{50} éléments étaient envisageables dans des implantations pratiques et il a fallu attendre les résultats d'A.O.L. Atkin et de N.D. Elkies combinant formes modulaires et isogénies pour atteindre des corps finis beaucoup plus gros, plus de 10^{100} éléments pour ceux de grande caractéristique [5, 39, 40, 103, 91]. Pour les corps de petite caractéristique, le calcul des isogénies sous-jacent aux améliorations précédentes ne fut rendu possible que par un algorithme de J.M. Couveignes [30] qui fait un usage intensif du groupe formel défini par la courbe. L'ensemble de ces idées forme le corps de l'algorithme `SEA` (Schoof-Elkies-Atkin).

L'objectif de cette thèse est l'application de l'algorithme `SEA` à des corps finis de caractéristique petite. Notre premier travail en fut l'implantation pour des corps finis de caractéristique 2 [70]. Nous avons pour cela optimisé l'algorithme de J.M. Couveignes, ce qui nous a permis en collaboration avec F. Morain de calculer la cardinalité d'une courbe définie sur $\mathbb{F}_{2^n}$ avec $n = 1009$ [69]. Malheureusement, le calcul d'isogénie avec cette méthode est le coût prépondérant du calcul de la cardinalité, contrairement à ce qui se passe pour $\mathbb{Z}/p\mathbb{Z}$. Pour pallier cet inconvénient, nous avons proposé un nouvel

algorithme bien plus efficace en pratique [67]. Avec celui-ci, nous avons ainsi pu obtenir la cardinalité d'une courbe définie sur $\mathbb{F}_{2^n}$ avec $n = 1301$ en un temps moindre (103 jours) que celui qu'il nous avait été nécessaire pour $\mathbb{F}_{2^{1009}}$ (121 jours) [72, 68]. Notamment, le temps de calcul des isogénies fut de seulement 3 jours pour $\mathbb{F}_{2^{1301}}$ alors qu'il avait fallu 78 jours pour $\mathbb{F}_{2^{1009}}$.

Pour généraliser ces résultats à tout corps fini, notamment ceux de caractéristique moyenne comprise entre 3 et 1000, nous exhibons des propriétés vérifiées par les isogénies sur les points d'ordre 2 de la courbe qui nous permettent de faire collaborer efficacement des algorithmes de calcul d'isogénies initialement prévus pour les corps de petite et de grande caractéristique. De plus nous avons écrit en collaboration avec F. Chabaud [22] l'ensemble des routines arithmétiques nécessaires : la librairie de calcul `ZEN`. Grâce à cet outil, nous disposons d'un programme unifié qui met en œuvre la majeure partie des nombreux algorithmes décrits dans ce mémoire. Nous sommes ainsi en mesure de traiter tout corps fini ayant moins de 10^{100} éléments.

Ce mémoire est organisé en trois parties. La partie I est une description théorique de l'algorithme `SEA`. La partie II est un "zoom" sur les algorithmes de calcul d'isogénies. Enfin, la partie III se préoccupe plus des aspects pratiques indispensables à une mise en œuvre efficace des algorithmes précédents.

La partie I s'ouvre sur les chapitres 1 et 2 qui mettent en place les objets mathématiques dont nous faisons par la suite un large usage. Il s'avère que la cardinalité d'un nombre marginal de courbes elliptiques ne peut pas être obtenue avec une application brutale des améliorations d'A.O.L. Atkin et de N.D. Elkies à l'algorithme de R. Schoof. Nous en faisons donc aussi un catalogue dans le chapitre 2 et donnons leur nombre de points par une méthode directe. Le chapitre 3 fait ensuite un tour d'horizon des idées utilisées dans l'algorithme `SEA`. Il nous fournit ainsi le cadre dans lequel s'insèrent les travaux de la partie II.

Les cinq chapitres de la partie II traitent chacun d'un algorithme de calcul d'isogénies. Ces algorithmes sont dus à N. Elkies, A.O.L. Atkin, L.S. Charlap, R. Coley et D.P. Robbins pour les corps de grande caractéristique, à J.M. Couveignes pour les corps de petites caractéristiques. Outre un nouvel algorithme spécialement élaboré pour les corps de caractéristique 2 et de nouveaux résultats vérifiés par les isogénies sur les points de 2-torsion de la courbe qui permettent de faire la jonction entre les algorithmes précédents, notre contribution est la réalisation d'une implantation efficace de ces idées. Précisément, le chapitre 4 fait le rappel de quelques techniques utilisées dans $\mathbb{C}$ et précise comment celles-ci sont applicables aux corps finis de grande caractéristique. Le chapitre 5 est une description complète du premier algorithme de J.M. Couveignes et de son optimisation. Le chapitre 6 décrit en détail le deuxième algorithme de J.M. Couveignes pour calculer des isogénies dans un corps fini de petite caractéristique. Nous présentons une première implantation en langage `C` de ces idées et expliquons quelles sont les techniques qu'il faudrait lui appliquer pour l'optimiser complètement. Les idées exposées dans le chapitre 7 proposent une alternative aux idées de J.M. Couveignes. Ils ont permis le calcul de la cardinalité d'une courbe définie sur $\mathbb{F}_{2^{1301}}$, ce qui est le record actuel pour les corps finis de caractéristique 2. Enfin, nous montrons dans le chapitre 8 comment généraliser les idées du chapitre 7 pour "faire coopérer" dans un corps fini de caractéristique moyenne des algorithmes initialement élaborés pour calculer des isogénies dans des corps de grande

ou petite caractéristique.

La partie III est plus technique. Dans le chapitre 9, nous comparons l'efficacité des méthodes qui sont connues pour calculer le pgcd de deux entiers. Notamment, nous proposons une amélioration de l'algorithme de Lehmer intéressante pour le calcul de pgcd étendus qui permet de gagner un facteur compris entre 3 et 6 suivant les architectures par rapport à un algorithme d'Euclide classique. Le chapitre 10 traite de la librairie de calcul `ZEN`. Il est le fruit d'un travail de programmation réalisé en collaboration avec F. Chabaud. Celui-ci est déjà décrit avec précision dans la première partie de la thèse de doctorat de ce dernier. Nous n'en rappelons ici que les concepts principaux et certains aspects complémentaires. Puis, le chapitre 11 montre comment utiliser des techniques de type "pas de bébé, pas de géant" pour mettre en œuvre efficacement l'algorithme `SEA`. Il décrit notamment une nouvelle variante de l'algorithme "tri-recherche" d'A.O.L. Atkin utilisé dans la phase terminale des calculs qui permet de diviser le nombre d'opérations nécessaires par un facteur supérieur à 2. Enfin, l'implantation des algorithmes des parties I et II a donné corps au programme `SEA` décrit dans le chapitre 12 où nous mettons en lumière quelles sont les retombées pratiques de ces travaux pour la cryptographie. En particulier, il s'avère que dans des corps finis de taille suffisante pour résister aux attaques connues contre le logarithme discret, nous effectuons le calcul du nombre de points de ces courbes en quelques secondes. Une exécution de ce programme est commentée dans l'annexe A.

Notations

Dans tout ce document, nous fixons $q = p^n$ où p est un nombre premier.

$\mathbb{N}$,	ensemble des entiers naturels.
$\mathbb{Z}$,	anneau des entiers relatifs.
$\mathbb{C}$,	corps des nombres complexes.
$\mathbb{Q}$,	corps des nombres rationnels.
$\mathbb{R}$,	corps des nombres réels.
$\mathrm{SL}_2(\mathbb{Z})$,	groupe des matrices 2×2 à coefficients dans $\mathbb{Z}$ et de déterminant égal à 1.
$\#E$,	cardinalité d'un ensemble fini E.
$\mathrm{taille}(n)$,	nombre de bits de la décomposition binaire d'un entier n.
$\nu(n)$,	nombre de bits égaux à 1 dans la décomposition binaire d'un entier n.
Ω,	nombre de bits des registres d'un microprocesseur, en général 32 ou 64.
$n \div m$,	quotient de la division euclidienne d'un entier n par un entier m.
$\lfloor x \rfloor$,	plus grand entier inférieur ou égal à x.
$\lceil x \rceil$,	plus petit entier supérieur ou égal à x.
$n \bmod m$,	reste de la division euclidienne d'un entier n par un entier m.
$\mathrm{pgcd}(n, m)$,	plus grand diviseur commun de deux entiers m et n.
$\mathrm{ppcm}(n, m)$,	plus petit multiple commun de deux entiers m et n.
$\lvert n \rvert$,	valeur absolue d'un entier n.
$\mathrm{sgn}(n)$,	signe d'une entier relatif n.
$n!$,	factorielle d'un entier n, c'est-à-dire l'entier $1 \times 2 \times \cdots \times n$.
$\left(\frac{m}{n}\right)$,	symbole de Jacobi de l'entier m par rapport à l'entier n.
$\mathbb{Z}/\ell\mathbb{Z}$,	anneau des entiers modulo ℓ.
$\mathbb{Z}_\ell$,	anneau des entiers ℓ-adiques.
$\mathbb{F}_{p^n}$,	corps fini à p^n éléments.
$\mathbb{F}_q[X]/(P(X))$,	extension algébrique d'un corps fini $\mathbb{F}_q$ définie par un polynôme $P(X)$ à coefficients dans $\mathbb{F}_q$. Cet ensemble peut être vu comme celui des polynômes modulo $P(X)$.
$\overline{\mathbb{K}}$,	la clôture algébrique d'un corps $\mathbb{K}$. C'est-à-dire l'ensemble des racines des polynômes à coefficients dans $\mathbb{K}$.
$\tilde{a}$,	$\sqrt[p]{a}$ pour un élément a de $\mathbb{F}_{p^n}$, c'est-à-dire $a^{p^{n-1}}$.
$\tilde{A}(X)$,	polynôme (resp. série) dont les coefficients sont les racines p-ième des coefficients du polynôme (resp. série) $A(X)$ défini sur $\mathbb{F}_{p^n}$.
$\mathrm{ord}(x)$	ordre d'un élément x appartenant à un groupe fini.
$\mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(x)$,	trace d'un élément x d'un corps fini $\mathbb{F}_{q^n}$ par rapport à $\mathbb{F}_q$. C'est-à-dire $x + x^q + \cdots + x^{q^{n-1}}$.
$\mathrm{N}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(x)$,	norme d'un élément x d'un corps fini $\mathbb{F}_{q^n}$ par rapport à $\mathbb{F}_q$. C'est-à-dire $x \times x^q \times \cdots \times x^{q^{n-1}}$.
$\mathrm{Résultant}(P, Q)$,	résultant de deux polynômes P et Q.

$\deg(P)$,	degré d'un polynôme P.
Id,	l'application identité.
$\mathrm{End}(E)$,	groupe des endomorphismes d'un groupe E.
$\mathrm{Hom}(E_a, E_b)$,	groupe des morphismes entre deux groupes E_a et E_b.
$\mathrm{Ker}(\phi)$,	noyau d'un morphisme ϕ.
$\mathrm{Im}(\phi)$,	image d'un morphisme ϕ.
$\phi \circ \psi$,	composition de deux morphismes.
$E[m]$,	groupe des points de m-torsion d'une courbe elliptique E.

Ordinateurs utilisés

Lorsqu'aucune référence n'est donnée ou lorsque "station DEC" est mentionné, les temps d'exécution indiqués dans ce mémoire ont été obtenus sur un ordinateur du constructeur Digital. Il s'agit précisément d'une machine "DEC station alpha 250" de quatrième génération cadencée à 266 MHz. Le système d'exploitation est OSF1, Digital UNIX V3.2. L'une des particularités de cette machine est que son microprocesseur dispose de registres de taille $\Omega = 64$ bits.

Lorsqu'en revanche "station SUN" est mentionné, il s'agit d'un ordinateur de type "sparc station 20" du constructeur SUN. Il est construit autour d'un microprocesseur de type "hypersparc" cadencé à 125 MHz dont la taille des registres est $\Omega = 32$ bits. Le système d'exploitation est SunOS 5.5.

Première partie

Algorithme SEA

Chapitre 1

Corps finis

Il est communément admis que la théorie des corps finis naît avec les travaux de Carl Friedrich Gauss (1777-1855) et ceux d'Évariste Galois (1811-1832). Avec l'avènement des mathématiques discrètes et leurs applications aux ordinateurs, cette théorie a pris ces dernières décennies une place prépondérante comme l'attestent les nombreux ouvrages consacrés au sujet. Les objets mathématiques que nous manipulons dans cette thèse sont tous construits dans des corps finis. Pour familiariser un lecteur non averti à ces concepts, nous en synthétisons ici les aspects principaux.

Pour essayer autant que possible d'éviter les travers d'une succession aride de définitions, théorèmes..., nous fournissons quand cela nous semble intéressant quelques schémas de démonstration. Notre présentation s'inspire pour une grande part du cours donné par D. Augot en Diplôme d'Étude Approfondi (DEA). Des justifications plus rigoureuses sont disponibles dans de nombreux ouvrages de références, citons parmi d'autres [74, 76, 112, 10].

1.1 Définition

Classiquement, un corps est un ensemble muni de deux lois de groupe vérifiant une relation de distributivité ; une loi d'addition commutative $+$ et une de loi de multiplication $\times$.

Définition 1. *Un corps est un anneau dont les éléments non nuls sont inversibles.*

Il n'est pas difficile d'exhiber des corps dont le nombre d'éléments est fini. Un mécanisme général utilisé dans les implantations (cf. chapitre 10) consiste à quotienter certains anneaux euclidiens $\mathbb{A}$ par l'un de leurs idéaux premiers $\mathbb{I}$. Ainsi ;

- avec $\mathbb{A} = \mathbb{Z}$ et $\mathbb{I} = p\mathbb{Z}$ où p est un nombre premier, nous obtenons $\mathbb{Z}/p\mathbb{Z}$, le corps à p éléments des "entiers modulo p".
- avec $\mathbb{A} = \mathbb{F}_q[X]$, l'ensemble des polynômes à coefficients dans un corps fini à q éléments et $\mathbb{I} = P(X)\mathbb{F}_q$ où $P(X)$ est un polynôme irréductible de degré n de $\mathbb{F}_q[X]$, nous obtenons un corps à q^n éléments que nous appellerons une "extension de degré n" de $\mathbb{F}_q$.

Dans la suite, nous allons voir que ces corps finis sont uniques à isomorphisme près. Auparavant, notons qu'un corps est parfois défini dans la littérature comme étant commutatif (c'est-à-dire de groupe multiplicatif commutatif). Dans notre cas, la finitude lève toute ambiguïté grâce à un théorème de Wedderburn.

Théorème 1. *Tout corps fini est commutatif.*

La démonstration de ce théorème est assez lourde. Par contre, il est beaucoup plus aisé d'obtenir le résultat suivant.

Proposition 1. *Tout anneau fini intègre, c'est-à-dire sans diviseur de zéro, est un corps fini.*

Esquisse de démonstration. Si a est un élément d'un tel anneau, il suffit de montrer que lorsque b décrit cet anneau, les éléments ab sont distincts et donc il existe un élément $\overline{a}$ tel que $a\overline{a} = \overline{a}a = 1$. □

1.2 Cardinalité

La cardinalité des corps finis est assez contrainte comme il apparaît dans le théorème suivant.

Théorème 2. *Soit $\mathbb{F}_q$ un corps fini de cardinal q. Alors il existe un nombre premier p et un entier n strictement positif tel que $q = p^n$.*

Esquisse de démonstration. Tout découle du morphisme d'anneau f défini de $\mathbb{Z}$ vers $\mathbb{F}_q$ et qui envoie l'entier n sur $n \times 1$. En effet, $\mathrm{Ker}(f)$ étant un idéal non nul de $\mathbb{Z}$, il est égal à $p\mathbb{Z}$ où p doit être un nombre premier. Donc $\mathrm{Im}(f)$, étant isomorphe à $\mathbb{Z}/\mathrm{Ker}(f)$, est en fait isomorphe à $\mathbb{Z}/p\mathbb{Z}$. Il suffit ensuite de considérer $\mathbb{F}_q$ comme un espace vectoriel de dimension finie n sur $\mathrm{Im}(f)$ pour conclure. □

1.3 Groupe multiplicatif

Nous prouvons ici par une succession de lemmes que le groupe multiplicatif $\mathbb{F}_q^*$ d'un corps fini $\mathbb{F}_q$ est cyclique (corollaire 1). Nous aurons besoin pour cela de la fonction d'Euler.

Définition 2. *La fonction d'Euler φ est l'application définie de $\mathbb{N}^*$ dans $\mathbb{N}^*$ par*

$$\forall n \in \mathbb{N}^*, \varphi(n) = \#\{1 \leqslant i \leqslant n, \mathrm{pgcd}(i, n) = 1\}.$$

Outre que cette fonction soit multiplicative, $\forall (p, q) \in \mathbb{N}^{*2}$, $\mathrm{pgcd}(p, q) = 1$, $\varphi(pq) = \varphi(p)\varphi(q)$, cette fonction a aussi la propriété suivante.

Proposition 2. *La fonction d'Euler φ vérifie*

$$\sum_{d \text{ divise } n} \varphi(d) = n.$$

Le premier lemme n'est que le théorème de Lagrange spécialisé à notre cas.

Lemme 1. *Soit $\alpha \in \mathbb{F}_q^*$. Alors* $\mathrm{ord}(\alpha)$ *divise $q - 1$.*

Le lemme suivant se démontre alors par un simple raisonnement sur les ordres des éléments.

Lemme 2. *Soit $\alpha \in \mathbb{F}_q^*$. Alors* $\mathrm{ord}(\alpha^i) = \mathrm{ord}(\alpha)/\mathrm{pgcd}(i, \mathrm{ord}(\alpha))$.

Le point crucial est alors ce lemme.

Lemme 3. *Soit un entier t divisant $q - 1$ où $\mathbb{F}_q$ est un corps fini. Alors, soit il n'y a pas d'éléments d'ordre t, soit il y a $\varphi(t)$ éléments d'ordre t.*

Esquisse de démonstration : supposons qu'il existe un élément α d'ordre t. De manière générale, les éléments d'ordre t sont les racines de $X^t - 1$. Or les t racines de ce polynôme sont α^i pour i variant de 1 à t. Comme d'autre part $\mathrm{ord}(\alpha^i) = t/\mathrm{pgcd}(i, t)$, il y a $\varphi(t)$ éléments d'ordre t. □

Et nous aboutissons ainsi au résultat suivant.

Théorème 3. *Soit un entier t divisant $q - 1$ où $\mathbb{F}_q$ est un corps fini. Alors il y a $\varphi(t)$ éléments d'ordre t.*

Esquisse de démonstration : soit $\psi(t)$ le nombre d'éléments d'ordre t, nous avons d'une part $\sum_{t \text{ divise } q-1} \psi(t) = q - 1$ et d'autre part $\psi(t) \leqslant \varphi(t)$ d'après le lemme 3. D'après la proposition 2, nous avons donc $\sum_{t \text{ divise } q-1} \varphi(t) - \psi(t) = 0$ et comme les termes de cette somme sont positifs, nous avons finalement $\psi(t) = \varphi(t)$. □

Par conséquent, il y a $\varphi(q - 1)$ éléments d'ordre $q - 1$ dans un corps fini $\mathbb{F}_q$.

Corollaire 1. *Le groupe multiplicatif $(\mathbb{F}_q^*, \times)$ est un groupe cyclique.*

À ce stade, il est facile d'introduire la notion de racine primitive.

Définition 3. *Un générateur de $(\mathbb{F}_q^*, \times)$ est appelé une racine primitive de $\mathbb{F}_q$.*

Ainsi deux corps finis de même cardinalité sont isomorphes.

1.4 Sous-corps d'un corps fini

Les sous-ensembles d'un corps fini qui sont eux-mêmes des corps sont clairement caractérisés.

Théorème 4. *Soit $\mathbb{K}$ un sous-corps d'un corps fini $\mathbb{F}_{p^n}$. Alors $\mathbb{K} = \mathbb{F}_{p^m}$ avec m divise n.*

Esquisse de démonstration : elle consiste à prouver que $\mathbb{F}_{p^n}$ est un $\mathbb{K}$-espace vectoriel. □

Le théorème suivant nous donne un moyen de tester si un élément d'un corps fini appartient à un de ses sous-corps.

Théorème 5. *Soit $\mathbb{F}_{p^m}$ un sous-corps d'un corps fini $\mathbb{F}_{p^n}$. Alors un élément x de $\mathbb{F}_{p^n}$ appartient à $\mathbb{F}_{p^m}$ si et seulement si $x^{p^m} = x$.*

Esquisse de démonstration : d'une part, on a $\forall x \in \mathbb{F}_{p^m}, x^{p^m} = x$ et réciproquement le polynôme $X^{p^m} - X$ a au plus p^m racines qui se trouvent être les éléments de $\mathbb{F}_{p^m}$. □

1.5 Automorphismes d'un corps fini

Il est d'usage d'associer à un élément α d'un corps fini un unique polynôme $p_\alpha(X)$ appelé polynôme minimal de α et défini comme suit.

Théorème 6. *Soit $\alpha \in \mathbb{F}_{p^n}$. Alors il existe un unique polynôme unitaire et irréductible $p_\alpha(X) \in \mathbb{F}_p[X]$ tel que $p_\alpha(\alpha) = 0$ et $p_\alpha(X)$ divise tout polynôme $f(X)$ de $\mathbb{F}_p[X]$ ayant α comme zéro. De plus* $\deg(p_\alpha) \leqslant n$.

Esquisse de démonstration : il suffit de remarquer que l'ensemble $\mathbb{I} = \{f \in \mathbb{F}_p[X], f(\alpha) = 0\}$ est un idéal principal distinct de $\{0\}$. Il est alors égal à $p_\alpha(X)\mathbb{F}_p[X]$ où $p_\alpha(X)$ est le polynôme recherché. □

Définissons les conjugués d'un élément α comme suit.

Définition 4. *Soit $\alpha \in \mathbb{F}_{p^n}$. Alors les conjugués de α sont les racines dans $\mathbb{F}_{p^n}$ de $p_\alpha(X)$.*

Grâce au lemme suivant fort utile dans les applications, il n'est pas difficile de prouver que α^p, α^{p^2}, $\ldots$,$\alpha^{p^{n-1}}$ sont racines de $p_\alpha(X)$.

Lemme 4. *Soient $\alpha_1, \ldots, \alpha_k$, k éléments d'un corps de caractéristique p. Alors*

$$\forall i \in \mathbb{N}, (\alpha_1 + \cdots + \alpha_k)^{p^i} = \alpha_1^{p^i} + \cdots + \alpha_k^{p^i}.$$

Esquisse de démonstration : il suffit de voir que les coefficients de la somme

$$(\alpha_1 + \cdots + \alpha_k)^{p^i} = \sum_{j_1 + \cdots + j_k = p^i} \frac{p^i!}{j_1! \cdots j_k!} \alpha_1^{j_1} \cdots \alpha_k^{j_k},$$

sont des multiples de p excepté pour $j_l = p^i$ $(0 \leqslant l \leqslant k)$. □

Le degré de p_α ou encore le nombre de conjugués associés à α est alors donné après une démonstration purement technique par le théorème suivant.

Théorème 7. *Soit $\alpha \in \mathbb{F}_{p^n}$. Le nombre de conjugués de α divise n. C'est le plus petit entier d tel que* $p^d \equiv 1 \bmod \mathrm{ord}(\alpha)$.

Enfin, le groupe des automorphismes d'un corps fini est lui aussi cyclique.

Proposition 3. *Les automorphismes d'un corps fini $\mathbb{F}_{p^n}$ sont* $\mathrm{Id}, \phi, \phi^2, \ldots, \phi^{n-1}$ *où l'automorphisme ϕ est donné par*

$$\begin{array}{rcl} \mathbb{F}_{p^n} & \to & \mathbb{F}_{p^n}, \\ x & \mapsto & x^p. \end{array}$$

Esquisse de démonstration : d'après le lemme 4, ϕ et ses composés ϕ^i sont des automorphismes, distincts dès que $i < n$. Réciproquement, soit f un morphisme et α une racine primitive de $\mathbb{F}_q$ de polynôme minimal $p_\alpha(X)$. Nous avons $p_\alpha(f(\alpha)) = f(p_\alpha(\alpha)) = 0$, et donc il existe un indice $i < n$ tel que $f(\alpha) = \alpha^{p^i}$ d'où on conclut facilement que $f = \phi^i$. □

1.6 Existence

Nous savons grâce au théorème 2 que le cardinal d'un corps fini est p^n. Inversement, la question est maintenant de savoir s'il existe pour tout entier p^n un corps fini de cette cardinalité. Pour $n = 1$, le corps $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ répond à la question. Pour $n > 1$, nous allons prouver qu'il existe toujours un polynôme irréductible $P(X)$ de degré n à coefficients dans $\mathbb{F}_p$ car le corps $\mathbb{F}_p[X]/P(X)\mathbb{F}_p[X]$ répond à la question.

En fait, nous sommes en mesure plus généralement de compter le nombre de polynômes irréductibles à l'aide de la fonction de Möbius μ définie de $\mathbb{N}^*$ dans $\mathbb{N}^*$ par

$$\forall n \in \mathbb{N}^*,\ \mu(n) = \begin{cases} 1, & \text{si } n = 1, \\ 0, & \text{si } n = p_1^{e_1} \cdots p_k^{e_k} \text{ et s'il existe } i \text{ tel que } e_i \geqslant 2, \\ (-1)^k & \text{si } n = p_1 \cdots p_k. \end{cases}$$

Pour cela, nous partons du théorème suivant.

Théorème 8. *Soient $V_d(X)$, le produit des polynômes irréductibles de degré d sur $\mathbb{F}_q[X]$. Alors*

$$X^{q^n} - X = \prod_{d \text{ divise } n} V_d(X). \tag{1.1}$$

Par passage aux degrés des polynômes qui interviennent dans l'égalité (1.1), nous obtenons alors le corollaire suivant.

Corollaire 2. *Soit I_d le nombre de polynômes irréductibles de degré d de $\mathbb{F}_q$. Alors*

$$q^n = \sum_{d \text{ divise } n} dI_d.$$

Il ne reste plus alors qu'à utiliser la formule classique "d'inversion de Möbius" pour compléter notre calcul.

Proposition 4. *Le nombre de polynômes irréductibles de degré n à coefficients dans un corps fini $\mathbb{F}_q$ est donné par*

$$I_n = \frac{1}{n} \sum_{d \text{ divise } n} \mu(d) q^{n/d}. \tag{1.2}$$

Un examen plus approfondi de la formule (1.2) montre que l'entier I_n est strictement positif pour tout couple (n, q), ce qui permet de conclure.

Corollaire 3. *Pour tout nombre premier p et tout entier n strictement positif, il existe un corps fini à p^n éléments.*

Chapitre 2

Courbes elliptiques définies sur des corps finis

Les courbes elliptiques ont une histoire que l'on peut, par certains côtés, faire remonter jusqu'aux travaux de Diophantus (250 après Jésus Christ). Derrière la définition simple de ces courbes, se cachent de multiples ramifications qui ont été depuis lors intensivement étudiées. Elles ont ainsi mené à des développements spectaculaires dans des domaines comme la géométrie algébrique ou encore la théorie des nombres. Récemment, Lenstra pour la factorisation d'entiers [64], puis Atkin et Morain pour la primalité d'entiers [88] ont fait des courbes elliptiques définies sur des corps finis l'un des outils privilégiés de la théorie algorithmique des nombres. Devant un tel panorama, nous limiterons notre récapitulatif de la section 2.1 aux définitions et propriétés indispensables à la compréhension des chapitres suivants.

Il se trouve qu'une application directe de certains des algorithmes que nous allons décrire par la suite se solde par un échec pour un nombre marginal de ces courbes. Cela est le cas pour les algorithmes de la partie II avec des courbes supersingulières de $\mathbb{F}_{2^n}$ et $\mathbb{F}_{3^n}$ ou avec des courbes à j-invariant égal à 0 ou 1728 dans $\mathbb{F}_{p^n}$, $p \geqslant 5$. C'est pourquoi nous donnons dans la section 2.2 le nombre de points de certaines de ces courbes.

2.1 Généralités

Nous nous intéressons principalement aux courbes elliptiques définies sur des corps finis. Néanmoins, certaines techniques utilisées lors du calcul de la cardinalité étendent des propriétés vérifiées uniquement dans des corps de caractéristique 0 (cf. par exemple le chapitre 4). Nous commençons donc par considérer la notion de courbe elliptique dans un corps quelconque $\mathbb{K}$ avant de spécialiser au corps des complexes $\mathbb{C}$ puis aux corps finis $\mathbb{F}_q$.

L'ouvrage de référence est le livre de Silverman [111].

2.1.1 Définition

Nous définissons une courbe elliptique comme suit.

Définition 5. *Une courbe elliptique E est une équation algébrique de la forme*

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6 \text{ avec } (a_1, a_2, a_3, a_4, a_6) \in \mathbb{K}^5 \tag{2.1}$$

où, si l'on définit les éléments d_2, d_4, d_6 et d_8 de $\mathbb{K}$ par

$$\begin{aligned} d_2 &= a_1^2 + 4a_2,\ d_4 = 2a_4 + a_1a_3,\ d_6 = a_3^2 + 4a_6, \\ d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \end{aligned}$$

l'on a

$$\Delta_E = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \text{ avec } \Delta_E \neq 0.$$

L'équation (2.1) est appelée paramétrisation de Weierstrass de E.

Définition 6. *On appelle espace projectif de dimension 2 associé à un corps $\mathbb{K}$ et on note $\mathbb{P}^2(\mathbb{K})$, l'ensemble des classes $(X : Y : Z)$ de la relation d'équivalence*

$$\forall (X,Y,Z) \in \mathbb{K}^3 \setminus \{0,0,0\}, \forall (X',Y',Z') \in \mathbb{K}^3 \setminus \{0,0,0\},$$
$$(X,Y,Z) \equiv (X',Y',Z') \Leftrightarrow \exists c \in \mathbb{K}^*, X' = cX, Y' = cY, Z' = cZ.$$

Nous pouvons maintenant définir $E(\mathbb{K})$.

Définition 7. *L'ensemble des points d'une courbe elliptique E définie sur un corps $\mathbb{K}$ est égal à*

$$E(\mathbb{K}) = \{(X : Y : Z) \in \mathbb{P}^2(\mathbb{K}), Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3\}.$$

Un seul point correspond à $Z = 0$, il s'agit de $O_E = (0 : 1 : 0)$ appelé point à l'infini et chaque classe différente de O_E a un unique représentant $(X : Y : 1)$. Nous considérons donc à partir de maintenant que $E(\mathbb{K})$ est la réunion de O_E avec l'ensemble

$$\{(X,Y) \in \mathbb{K}^2, Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6\}. \tag{2.2}$$

2.1.2 Loi de groupe

Théorème 9. *L'ensemble $E(\mathbb{K})$ est un groupe abélien si on le munit de la loi d'addition $+$ suivante :*

- *pour tout point $P = (x_P, y_P)$ de $E(\mathbb{K})$,*

$$P + O_E = O_E + P = P \text{ et } -P = (x_P, -y_P - a_1x_P - a_3).$$

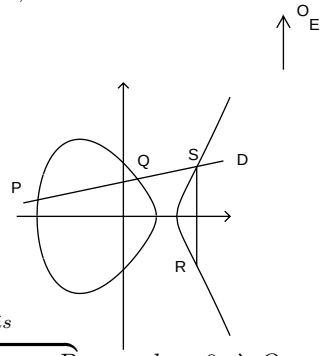
- *pour tous points $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ de $E(\mathbb{K})$ avec $P \neq -Q$, les coordonnées (x_{P+Q}, y_{P+Q}) du point $R = P + Q$ sont égales à*

$$\begin{cases} x_{P+Q} &= \lambda^2 + a_1\lambda - a_2 - x_P - x_Q, \\ y_{P+Q} &= -(\lambda + a_1)x_{P+Q} - \nu - a_3, \end{cases}$$

avec

$$\lambda = \begin{cases} (y_Q - y_P)/(x_Q - x_P) & \text{si } P \neq Q, \\ (3x_P^2 + 2a_2x_P + a_4 - a_1y_P)/(2y_P + a_1x_P + a_3) & \text{sinon,} \end{cases} \quad \text{et } \nu = y_P - \lambda x_P.$$

Sur $\mathbb{R}$, cela revient à associer à deux points P et Q le symétrique R par rapport à l'axe des abscisses du point intersection S de la droite D passant par P et Q avec la courbe. Lorsque les deux points sont égaux, on prend pour D la tangente en P à la courbe.



Soit $k \in \mathbb{Z}$. Il est d'usage de noter kP la quantité égale à $\overbrace{P + P + \cdots + P}^{k\ fois}$ pour $k > 0$, à O_E pour $k = 0$ et à $(-k)(-P)$ pour $k < 0$. Nous définissons alors l'endomorphisme "multiplication par un entier m" comme étant égal à

$$\begin{array}{rccc} [m]_E & : & E(\mathbb{K}) & \to & E(\mathbb{K}), \\ & & P & \mapsto & mP. \end{array}$$

Dans la suite, nous notons $E[m]$ le noyau de $[m]_E$ défini sur $\overline{\mathbb{K}}$, la clôture algébrique de $\mathbb{K}$, c'est-à-dire

$$E[m] = \left\{(X,Y) \in E(\overline{\mathbb{K}}), [m]_E(X,Y) = O_E\right\}.$$

Grâce au théorème suivant, nous avons une expression explicite pour $[m]_E$ à l'aide "des polynômes de division" de E.

Définition 8. *Avec les notations de la définition 5, les polynômes de division $f_m(X)$ d'une courbe elliptique E sont définis par*

$$f_0(X) = 0,\ f_1(X) = 1,\ f_2(X) = 1,\ f_3(X) = 3X^4 + d_2X^3 + 3d_4X^2 + 3d_6X + d_8,$$
$$f_4(X) = 2X^6 + d_2X^5 + 5d_4X^4 + 10d_6X^3 + 10d_8X^2 + (d_2d_8 - d_4d_6)X + (d_4d_8 - d_6^2),$$

et en posant $F(X) = 4X^3 + d_2X^2 + 2d_4X + d_6$,

- $f_{2m} = f_m(f_{m+2}f_{m-1}^2 - f_{m-2}f_{m+1}^2)$,
- $f_{2m+1} = \begin{cases} F^2 f_{m+2}f_m^3 - f_{m-1}f_{m+1}^3 & \text{si } m \text{ est pair,} \\ f_{m+2}f_m^3 - f_{m-1}f_{m+1}^3F^2 & \text{sinon.} \end{cases}$

Le polynôme f_m est donc de degré au plus $(m^2-1)/2$ si m est impair ou de degré au plus $(m^2-2)/2$ si m est pair. Nous avons alors le résultat suivant.

Théorème 10. *Soient E une courbe elliptique définie sur un corps $\mathbb{K}$, P un point de $E(\mathbb{K})$ et $m \in \mathbb{N}^*$. Alors*

$$[m]_E(P) = \begin{cases} O_E & \text{si } P \in E[m], \\ \left(\dfrac{\phi_m(X,Y)}{\psi_m^2(X,Y)}, \dfrac{\omega_m(X,Y)}{\psi_m^3(X,Y)}\right) & \text{si } P = (X,Y) \in E(\overline{\mathbb{K}}) - E[m], \end{cases}$$

où les polynômes ψ_m, ϕ_m et ω_m satisfont

$$\psi_m = \begin{cases} (2Y + a_1X + a_3)f_m & \text{si } m \text{ est pair,} \\ f_m & \text{sinon,} \end{cases}$$

et

$$\phi_m = X\psi_m^2 - \psi_{m-1}\psi_{m+1},\ 2\psi_m\omega_m = \psi_{2m} - \psi_m^2(a_1\phi_m + a_3\psi_m^2).$$

Notons que pour des corps de caractéristique différente de 2, ce théorème nous fournit une construction explicite de $[m]_E$. Il est aussi valide pour ceux de caractéristique 2, néanmoins quelques précautions sont à prendre quant à l'obtention de ω_m.

Remarquons enfin qu'il est possible de déterminer si un point P appartient à $E[m]$ avec le théorème suivant.

Théorème 11. *Soit $P \in E(\overline{\mathbb{K}})$. Alors $P \in E[m]$ si et seulement si $P = O_E$ ou lorsque $P = (X,Y)$, si $f_m(X) = 0$.*

2.1.3 Morphismes

Nous nous intéressons dans cette section à l'ensemble $\mathrm{Hom}_{\mathbb{K}}(E_a, E_b)$ des morphismes définis entre deux courbes elliptiques E_a et E_b sur un corps $\mathbb{K}$. Une fois donnés quelques résultats généraux à ce sujet, nous précisons la structure de $\mathrm{End}(E)$ classiquement défini comme étant égal à $\mathrm{End}_{\overline{\mathbb{K}}}(E)$ où $\mathrm{End}_{\mathbb{K}} = \mathrm{Hom}_{\mathbb{K}}(E,E)$. Nous terminons par une caractérisation des isomorphismes entre courbes elliptiques.

Isogénies

Une isogénie est définie comme suit [111, p. 70].

Définition 9. *Une isogénie $\mathcal{I}$ est un morphisme entre $E_a(\mathbb{K})$ et $E_b(\mathbb{K})$.*

Par exemple, l'endomorphisme $[m]_{E_a}$, multiplication par m sur une courbe elliptique E_a, est une isogénie. Plus généralement, le noyau et l'image d'une isogénie satisfont les conditions suivantes.

Théorème 12. *Soit $\mathcal{I}$ une isogénie non nulle définie d'une courbe E_a vers une courbe E_b sur un corps $\mathbb{K}$. Alors :*

1. $\mathrm{Ker}(\mathcal{I})$ *est un groupe fini.*
2. *Si $\mathbb{K}$ est un corps algébriquement clos, $\mathcal{I}$ est surjective.*

Lorsque $\mathcal{I}$ est surjective, elle induit l'injection suivante sur les corps de fonctions associés à E_a et E_b ;

$$\begin{array}{rccc} \mathcal{I}^* & : & \mathbb{K}(E_b) & \to & \mathbb{K}(E_a), \\ & & f & \mapsto & f \circ \mathcal{I}. \end{array}$$

Nous pouvons alors définir le degré d'une isogénie.

Définition 10. *Si $\mathbb{K}$ est un corps algébriquement clos, une isogénie $\mathcal{I}$ entre deux courbes elliptiques E_a et E_b définies sur $\mathbb{K}$ est dite séparable ou inséparable si l'extension $[\mathbb{K}(E_a) : \mathcal{I}^*(\mathbb{K}(E_b))]$ a ces propriétés. On note alors $\deg_s(\mathcal{I})$ et $\deg_i(\mathcal{I})$ les degrés correspondants. Le degré de $\mathcal{I}$ est alors égal à $\deg(\mathcal{I}) = \deg_s(\mathcal{I})\deg_i(\mathcal{I})$.*

Nous avons alors les propriétés suivantes.

Théorème 13. *Soit $\mathcal{I}$ une isogénie non constante d'une courbe E_a vers une courbe E_b sur un corps algébriquement clos $\mathbb{K}$. Alors :*

1. *pour tout point S de $E_b(\mathbb{K})$, nous notons $\mathcal{I}^{-1}(S)$ le sous-ensemble de $E_a(\mathbb{K})$ qui a pour image S. Le cardinal de $\mathcal{I}^{-1}(S)$ est fini et égal à $\deg_s(\mathcal{I})$;*
2. *si m est un entier positif, $[m]_{E_a}$ est une isogénie de degré m^2.*

Lorsque le corps $\mathbb{K}$ n'est pas algébriquement clos, nous définissons le degré d'une isogénie comme étant le degré de l'isogénie considérée comme définie sur la clôture algébrique de $\mathbb{K}$. Enfin, nous donnons une dernière propriété importante.

Théorème 14. *Soit $\mathcal{I}$ une isogénie non constante définie d'une courbe E_a vers une courbe E_b sur un corps $\mathbb{K}$. Alors, il existe une unique isogénie $\hat{\mathcal{I}}$ de E_b vers E_a telle que $\hat{\mathcal{I}} \circ \mathcal{I} = [\deg(\mathcal{I})]_{E_a}$. De plus $\deg(\hat{\mathcal{I}}) = \deg(\mathcal{I})$. Cette isogénie est appelée* l'isogénie duale *de $\mathcal{I}$.*

Ainsi, lorsqu'il existe une isogénie de degré $\ell \neq 0$ entre deux courbes elliptiques, nous dirons que ces courbes sont *isogènes de degré ℓ.*

Endomorphismes

Sauf dans le cas des corps finis (cf. section 2.1.5), $\mathrm{End}(E)$ est généralement réduit aux multiplications $[m]_E$. Dans ce cas $\mathrm{End}(E)$ est isomorphe à $\mathbb{Z}$. Il est d'usage de distinguer les courbes qui ne sont pas de ce type.

Définition 11. *Une courbe elliptique E est dite à multiplication complexe si* $\mathrm{End}(E)$ *contient un endomorphisme différent de $[m]_E$ pour tout entier m.*

Dans tous les cas, la structure de $\mathrm{End}(E)$ est donnée par le théorème suivant.

Théorème 15. $\mathrm{End}(E)$ *est soit isomorphe à $\mathbb{Z}$, soit isomorphe à un ordre dans un corps de nombres quadratique imaginaire, soit isomorphe à un ordre dans une algèbre de quaternions.*

Isomorphismes

Deux courbes elliptiques sont dites isomorphes si elles sont isomorphes en tant que variétés algébriques projectives. On montre alors le théorème suivant.

Théorème 16. *Deux courbes elliptiques E_a et E_b définies sur un corps $\mathbb{K}$ par*

$$\begin{array}{rccc} E_a & : & Y^2 + a_1XY + a_3Y & = & X^3 + a_2X^2 + a_4X + a_6, \\ E_b & : & Y^2 + b_1XY + b_3Y & = & X^3 + b_2X^2 + b_4X + b_6, \end{array}$$

sont isomorphes si et seulement si

$$\exists (u,r,s,t) \in \mathbb{K}^* \times \mathbb{K}^3, \begin{cases} ub_1 &= a_1 + 2s, \\ u^2b_2 &= a_2 - sa_1 + 3r - s^2, \\ u^3b_3 &= a_3 + ra_1 + 2t, \\ u^4b_4 &= a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st, \\ u^6b_6 &= a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1. \end{cases}$$

L'isomorphisme de E_a vers E_b est alors donné par $(X,Y) \to (u^2X + r, u^3Y + u^2sX + t)$.

Il est possible d'obtenir une condition simplifiée d'isomorphisme à partir des j-invariants des courbes.

Définition 12. *On appelle j-invariant d'une courbe elliptique E_a définie sur un corps $\mathbb{K}$, la quantité*

$$j_{E_a} = \frac{c_4^3}{\Delta_{E_a}} \text{ où } c_4 = d_2^2 - 24d_4.$$

Nous avons alors.

Théorème 17. *Si deux courbes elliptiques E_a et E_b définies sur un corps $\mathbb{K}$ sont isomorphes, alors $j_{E_a} = j_{E_b}$. La réciproque est vraie si $\mathbb{K}$ est un corps algébriquement clos.*

Pour alléger en pratique le calcul de la loi d'addition, on n'utilise pas directement la paramétrisation de Weierstrass (2.1) mais plutôt l'une des paramétrisations canoniques de la proposition suivante.

Proposition 5. *Soit E une courbe elliptique donnée dans un corps $\mathbb{K}$ par une équation de Weierstrass. Alors il existe un isomorphisme $(X, Y) \to (u^2X + r, u^3Y + u^2sX + t)$ qui ramène cette courbe à l'une des courbes "canoniques" du tableau 2.1.*

Condition		Équation	Δ	j-invariant
$\mathrm{Car}(\mathbb{K}) \neq 2, 3$		$Y^2 = X^3 + a_4X + a_6$	$-16(4a_4^3 + 27a_6^2)$	$1728\dfrac{4a_4^3}{4a_4^3 + 27a_6^2}$
$\mathrm{Car}(\mathbb{K}) = 3,$	$j_E \neq 0$	$Y^2 = X^3 + a_2X^2 + a_6$	$-a_2^3a_6$	$-a_2^3/a_6$
	$j_E = 0$	$Y^2 = X^3 + a_4X + a_6$	$-a_4^3$	0
$\mathrm{Car}(\mathbb{K}) = 2$	$j_E \neq 0$	$Y^2 + XY = X^3 + a_2X^2 + a_6$	a_6	$1/a_6$
	$j_E = 0$	$Y^2 + a_3Y = X^3 + a_4X + a_6$	a_3^4	0

FIG. 2.1 – Courbes elliptiques canoniques définies sur un corps $\mathbb{K}$.

Enfin, en appliquant le théorème 16 au cas $E_a = E_b$, nous obtenons le théorème suivant.

Théorème 18. *Soit E_a une courbe elliptique définie sur un corps $\mathbb{K}$. Alors son groupe d'automorphisme est un groupe fini d'ordre divisant 24. Précisément, cet ordre est égal à*

$$\begin{cases} 2 & \text{si } j_{E_a} \neq 0, 1728, \\ 4 & \text{si } j_{E_a} = 1728 \text{ et } \mathbb{K} \text{ est un corps de caractéristique } \neq 2, 3, \\ 6 & \text{si } j_{E_a} = 0 \text{ et } \mathbb{K} \text{ est un corps de caractéristique } \neq 2, 3, \\ 12 & \text{si } j_{E_a} = 0 \text{ et } \mathbb{K} \text{ est un corps de caractéristique } 3, \\ 24 & \text{si } j_{E_a} = 0 \text{ et } \mathbb{K} \text{ est un corps de caractéristique } 2. \end{cases}$$

2.1.4 Spécialisation à $\mathbb{C}$

La structure de courbes elliptiques définies sur $\mathbb{C}$ est dès plus simple puisqu'il s'agit à isomorphisme près du quotient de $\mathbb{C}$ par un réseau $\Lambda = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ ($\mathrm{Im}\ \omega_1/\omega_2 > 0$). Précisément, associées à un réseau Λ, nous définissons les séries d'Eisenstein G_{2k} ($k \in \mathbb{N}$, $k > 1$) par

$$G_{2k} = \sum_{\omega \in \Lambda - \{0\}} \omega^{-2},$$

et la fonction de Weierstrass $\wp$ par

$$\forall z \in \mathbb{C},\ \wp(z) = z^{-2} + \sum_{\omega \in \Lambda - \{0\}} \left((z - \omega)^{-2} - \omega^{-2}\right).$$

Notons que $\wp$ est paire, périodique modulo le réseau Λ et qu'on peut la développer comme

$$\wp(z) = z^{-2} + \sum_{k=1}^{\infty}(2k+1)G_{2k}z^{2k}.$$

Nous avons alors le résultat suivant.

Théorème 19. *Soient Λ un réseau de $\mathbb{C}$ et $g_2 = 60G_4$, $g_3 = 140G_6$. Alors, la correspondance définie de $\mathbb{C}/\Lambda$ vers la courbe elliptique $E : Y^2 = 4X^3 - g_2X - g_3$, par*

$$z \mapsto \begin{cases} (\wp(z), \wp'(z)) \text{ si } z \neq 0, \\ O_E \text{ sinon,} \end{cases}$$

est un isomorphisme.

L'un des intérêts de cet isomorphisme est la caractérisation aisée des endomorphismes d'une courbe. En effet, tout endomorphisme d'une courbe E peut être identifié à un nombre complexe α vérifiant $\alpha\Lambda \subset \Lambda$. Nous avons alors dans ce cas $\forall z \in \mathbb{C} - \Lambda$, $(\wp(\alpha z), \wp'(\alpha z)) \in E$.

2.1.5 Spécialisation aux corps finis

Cardinalité

Contrairement aux autres corps, notons tout d'abord que toute courbe elliptique E définie sur un corps fini $\mathbb{F}_q$ est à multiplication complexe puisque l'on peut montrer que l'endomorphisme Frobenius de $\mathrm{End}(E)$ défini comme suit est distinct de $[m]_E$ pour tout entier m.

Définition 13. *Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de caractéristique p. Alors l'endomorphisme Frobenius de E est défini par*

$$\begin{array}{rccl} \phi_E : & E(\bar{\mathbb{F}}_q) & \to & E(\bar{\mathbb{F}}_q), \\ & P & \mapsto & \begin{cases} O_E & \text{si } P = O_E, \\ (X^q, Y^q) & \text{si } P = (X, Y). \end{cases} \end{array}$$

Le résultat[1] suivant joue dans la suite un rôle particulièrement important.

Théorème 20. *Le polynôme caractéristique de ϕ_E est*

$$\phi_E^2 - [c]_E \circ \phi_E + [q]_E = 0$$

où c est défini par

$$\#E(\mathbb{F}_q) = q + 1 - c \text{ et vérifie de plus } |c| \leq 2\sqrt{q}.$$

Pour une courbe définie sur $\mathbb{F}_q$, il est facile d'obtenir sa cardinalité sur $\mathbb{F}_{q^k}$ $(k \in \mathbb{N}^*)$ à l'aide de celle sur $\mathbb{F}_q$ avec le théorème de Weil.

Théorème 21. *Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de cardinalité $q + 1 - c$. Alors la cardinalité de $E(\mathbb{F}_{q^k})$ avec $k \in \mathbb{N}^*$ est égale à $q^k + 1 - \alpha^k - \beta^k$ où α et β sont les racines complexes du polynôme $qX^2 - cX + 1$.*

Enfin, nous avons ce dernier résultat.

Théorème 22. *Deux courbes elliptiques sont isogènes sur le corps $\mathbb{F}_q$ si et seulement si elles ont même cardinalité.*

[1] Démontré pour la première fois par Hasse [47].

Structure

La structure de $E(\mathbb{F}_q)$ est caractérisée par le théorème suivant.

Théorème 23. *Le groupe abélien $E(\mathbb{F}_q)$ est de rang 1 ou 2. Il est isomorphe à $\mathbb{Z}/n_1\mathbb{Z} \oplus \mathbb{Z}/n_2\mathbb{Z}$ où n_2 divise n_1 et de plus n_2 divise $q-1$.*

Morphismes

De nombreux algorithmes des chapitres suivants s'appuient sur $E[m]$. Avant d'en préciser aussi la structure, définissons la notion de courbe supersingulière.

Définition 14. *Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de caractéristique p et de cardinalité $q+1-c$. La courbe E est dite supersingulière quand p divise c.*

Pour de telles courbes, $\mathrm{End}(E)$ est donné par le théorème suivant.

Théorème 24. *Soit E une courbe supersingulière, alors $\mathrm{End}(E)$ est isomorphe à un ordre dans une algèbre de quaternions.*

D'autre part, notons que la notion *d'invariant de Hasse* caractérise ces courbes. Une définition rigoureuse de cet invariant fait appel à la notion de groupe formel (cf. théorème 5.14 du chapitre 5). Pour le cas particulier des courbes canoniques du tableau 2.1, la définition suivante est cependant d'un emploi plus aisé.

Définition 15. *Nous appelons invariant de Hasse d'une courbe elliptique canonique E définie sur un corps fini $\mathbb{F}_q$ de caractéristique p, la quantité c_E égale à a_1 si $p=2$, ou égale au coefficient de X^{p-1} du polynôme $(X^3+a_2X^2+a_4X+a_6)^{\frac{p-1}{2}}$ sinon.*

Nous avons alors.

Théorème 25. *Une courbe elliptique E est supersingulière si et seulement si son invariant de Hasse est nul.*

Notons que si E n'est pas une courbe supersingulière de $\mathbb{F}_q$, il existe une courbe canonique qui lui est isomorphe, éventuellement dans $\bar{\mathbb{F}}_q$, avec un invariant de Hasse égal à 1. Enfin, nous sommes en mesure de préciser $E[m]$.

Théorème 26. *Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de caractéristique p et ℓ un entier non nul. Alors :*

- *si $\mathrm{pgcd}(\ell, q) = 1$, $E[\ell]$ est isomorphe à $\mathbb{Z}/\ell\mathbb{Z} \oplus \mathbb{Z}/\ell\mathbb{Z}$.*
- *si $\ell = p^i$, $i \in \mathbb{N}^*$, $E[\ell]$ est isomorphe à* $\begin{cases} \{O_E\} & \text{si } E \text{ est supersingulière,} \\ \mathbb{Z}/\ell\mathbb{Z} & \text{sinon.} \end{cases}$

2.2 Courbes particulières

Nous faisons ici l'inventaire des courbes "pathologiques" pour les calculs d'isogénie de la partie II. L'objectif est, à partir d'une courbe elliptique E donnée sous sa forme la plus générale par $E : Y^2 + a_1XY + a_3X = X^3 + a_2X^2 + a_4X + a_6$, dans un corps fini $\mathbb{F}_{p^n}$, d'être capable de déterminer si cette courbe est pathologique et dans l'affirmative d'obtenir son nombre de points, voire dans certains cas la structure du groupe abélien $E(\mathbb{F}_{p^n})$. Nous classifions ces courbes selon leurs invariants j_E puisque cette quantité est aisément calculable à partir de l'équation de E. Ainsi, les cas $j_E = 0$ (section 2.2.1) et $j_E = 1728$ (section 2.2.2) sont complètement traités, avant de considérer succinctement les autres cas (section 2.2.3). Les références en la matière sont les travaux de Deuring [35], Waterhouse [116] et Schoof [102].

Notons par ailleurs que la factorisation de certaines équations modulaires est perturbée par des courbes à multiplication complexe par un ordre dans un corps quadratique imaginaire de nombre de classes petit (chapitre 3). Néanmoins, nous ne disposons dans ce dernier cas d'aucune méthode générale

pour les détecter et les perturbations qu'elles entraînent ne rendent inutilisables qu'un petit nombre des équations modulaires préconisées par Atkin. Il est donc tout à fait possible d'obtenir la cardinalité de ces courbes avec l'algorithme de Schoof et ses variantes, quitte à supprimer le calcul de leurs cardinalités modulo un nombre négligeable de nombres premiers. En conséquence, nous ne traiterons pas de ces dernières ici.

2.2.1 Cas $j_E = 0$

Pour un corps fini $\mathbb{F}_{p^n}$ de caractéristique égale à 2 ou 3, les courbes supersingulières posent problème. Le théorème suivant permet aisément de les identifier.

Théorème 27. *Une courbe elliptique E définie sur $\mathbb{F}_{2^n}$ ou $\mathbb{F}_{3^n}$ est supersingulière si et seulement si $j_E = 0$.*

Pour le cas des corps finis de caractéristique supérieure ou égale à 5, toute courbe est isomorphe à une courbe d'équation $E : Y^2 = X^3 + a_4X + a_6$. Or, notre restriction $j_E = 0$ impose $a_4 = 0$. Nous sommes donc réduits dans ce cas à l'étude de courbes d'équation

$$E : Y^2 = X^3 + a_6,\ a_6 \in \mathbb{F}_{p^n}^*. \tag{2.3}$$

Corps finis de caractéristique 2

Une courbe elliptique de $\mathbb{F}_{2^n}$ à j-invariant nul est isomorphe à une courbe d'équation

$$E : Y^2 + a_3Y = X^3 + a_4X + a_6,\ a_3 \in \mathbb{F}_{p^n}^*,\ (a_4, a_6) \in \mathbb{F}_{p^n}^2. \tag{2.4}$$

Les travaux de Menezes [79] nous fournissent une équation explicite de ces courbes.

Théorème 28. *Soient α, β, γ et δ des éléments de $\mathbb{F}_{2^n}$ avec γ un non-cube, $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(\gamma^{-2}\alpha) = 1$, $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(\gamma^{-4}\beta) = 1$, $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(\omega) = 1$ et $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_4}(\delta) \neq 0$ pour n pair. Alors une courbe elliptique supersingulière définie sur $\mathbb{F}_{2^n}$ est isomorphe à l'une des courbes E du tableau 2.2.*

Du point de vue algorithmique, il est donc aisé à partir d'une courbe elliptique supersingulière E de se ramener à une équation du type (2.4) pour ensuite trouver la classe d'isomorphismes du tableau 2.2 à laquelle elle se rattache et finalement en déduire la cardinalité et la structure de groupe.

Corps finis de caractéristique 3

Une courbe elliptique de $\mathbb{F}_{3^n}$ à j-invariant nul est isomorphe à une courbe d'équation

$$E : Y^2 = X^3 + a_4X + a_6,\ a_4 \in \mathbb{F}_{p^n}^*,\ a_6 \in \mathbb{F}_{p^n}.$$

Grâce aux travaux de Morain [92], nous pouvons exploiter un résultat analogue au théorème 28.

Théorème 29. *Soient γ et δ, deux éléments de $\mathbb{F}_{3^n}$ avec γ un non carré et $\mathrm{Tr}_{\mathbb{F}_{3^n}/\mathbb{F}_3}(\delta) = 1$. Alors une courbe elliptique supersingulière définie sur $\mathbb{F}_{3^n}$ est isomorphe à une des courbes E du tableau 2.3.*

Autres corps finis

Dans les corps finis $\mathbb{F}_{p^n}$ avec $p \geqslant 5$, toute courbe elliptique avec j-invariant nul est isomorphe à une courbe E d'équation (2.3). Lorsque $p \equiv 1 \bmod 3$, un théorème de Katre et Rajwade [54] nous donne directement la cardinalité de E.

Théorème 30. *Si $\mathbb{F}_{p^n}$ est un corps fini avec $p \equiv 1 \bmod 3$, alors le nombre de points de la courbe elliptique $E : Y^2 = X^3 + a_6$ définie sur $\mathbb{F}_{p^n}$ est*

$$p^n + 1 + a_6^{(p^n-1)/2}(1 + \Phi_3(a_6^{-1})) \text{ avec } \Phi_3(a) = \begin{cases} -1 + s \text{ si } 2a \text{ est un cube,} \\ -1 - \dfrac{1}{2}(s - 9t) \text{ sinon,} \end{cases}$$

où s et t sont définis par $4p^n = s^2 + 27t^2$, p ne divise pas s, $s \equiv 1 \bmod 3$ et lorsque $2a$ n'est pas un cube, $(2a)^{(p^n-1)/3} \equiv (s + 9t)/(s - 9t) \bmod p$.

E	n	$\#E(\mathbb{F}_{2^n})$	Structure de groupe
$Y^2+Y=X^3$	*impair*	p^n+1	cyclique
$Y^2+Y=X^3+X$	$n\equiv 1,7 \bmod 8$	$p^n+1+\sqrt{2p^n}$	cyclique
	$n\equiv 3,5 \bmod 8$	$p^n+1-\sqrt{2p^n}$	cyclique
$Y^2+Y=X^3+X+1$	$n\equiv 1,7 \bmod 8$	$p^n+1-\sqrt{2p^n}$	cyclique
	$n\equiv 3,5 \bmod 8$	$p^n+1+\sqrt{2p^n}$	cyclique
$Y^2+Y=X^3+\delta X$	*pair*	p^n+1	cyclique
$Y^2+Y=X^3$	$n\equiv 0 \bmod 4$	$p^n+1-2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}-1)\mathbb{Z}}\oplus\mathbb{Z}_{/(\sqrt{p^n}-1)\mathbb{Z}}$
	$n\equiv 2 \bmod 4$	$p^n+1+2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}+1)\mathbb{Z}}\oplus\mathbb{Z}_{/(\sqrt{p^n}+1)\mathbb{Z}}$
$Y^2+Y=X^3+\omega$	$n\equiv 0 \bmod 4$	$p^n+1+2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}+1)\mathbb{Z}}\oplus\mathbb{Z}_{/(\sqrt{p^n}+1)\mathbb{Z}}$
	$n\equiv 2 \bmod 4$	$p^n+1-2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}-1)\mathbb{Z}}\oplus\mathbb{Z}_{/(\sqrt{p^n}-1)\mathbb{Z}}$
$Y^2+\gamma Y=X^3$	$n\equiv 0 \bmod 4$	$p^n+1+\sqrt{p^n}$	cyclique
	$n\equiv 2 \bmod 4$	$p^n+1-\sqrt{p^n}$	cyclique
$Y^2+\gamma Y=X^3+\alpha$	$n\equiv 0 \bmod 4$	$p^n+1-\sqrt{p^n}$	cyclique
	$n\equiv 2 \bmod 4$	$p^n+1+\sqrt{p^n}$	cyclique
$Y^2+\gamma^2 Y=X^3$	$n\equiv 0 \bmod 4$	$p^n+1+\sqrt{p^n}$	cyclique
	$n\equiv 2 \bmod 4$	$p^n+1-\sqrt{p^n}$	cyclique
$Y^2+\gamma^2 Y=X^3+\beta$	$n\equiv 0 \bmod 4$	$p^n+1-\sqrt{p^n}$	cyclique
	$n\equiv 2 \bmod 4$	$p^n+1+\sqrt{p^n}$	cyclique

FIG. 2.2 – Courbes supersingulières définies sur $\mathbb{F}_{2^n}$.

Du point de vue algorithmique, l'algorithme de Cornacchia (cf. [27, Algorithme 1.5.3] ou [88, 96] pour une justification théorique) permet de trouver efficacement deux entiers u et v tels que $4p=u^2+27v^2$ pour $p\equiv 1 \bmod 3$. Il suffit de remarquer qu'alors $(\pm u\pm 3v\sqrt{-3})/2$ est un entier algébrique de norme p dans $\mathbb{Q}(\sqrt{-3})$ pour obtenir un entier algébrique $(s+3t\sqrt{-3})/2$ de norme p^n par

$$(s+3t\sqrt{-3})/2=((\pm u\pm 3v\sqrt{-3})/2)^n$$

où les signes de u et v sont choisis pour que p ne divise pas s. Ainsi, le calcul de la cardinalité d'une telle courbe est aisé.

Pour $p\equiv 2 \bmod 3$, la courbe est alors supersingulière et en suivant les travaux de Morain [92], nous avons le théorème suivant.

Théorème 31. *Si $\mathbb{F}_{p^n}$ est un corps fini avec $p\equiv 2 \bmod 3$, alors le nombre de points de la courbe elliptique $E: Y^2=X^3+a_6$ définie sur $\mathbb{F}_{p^n}$ est*

$$p^n+1-c \text{ avec } c=\begin{cases} 0 \text{ si } n \text{ est impair},\\ (-1)^{(n+2)/2}b^{(p^n-1)/2}\sqrt{p^n} \text{ si } n \text{ est pair et } b \text{ n'est pas un cube},\\ (-1)^{n/2}b^{(p^n-1)/2}2\sqrt{p^n} \text{ si } n \text{ est pair et } b \text{ est un cube}. \end{cases}$$

2.2.2 Cas $j_E=1728$

Le cas des corps finis de caractéristique 2 et 3 étant réglé dans la section précédente puisque $1728=2^6 3^3$, il reste le cas de $\mathbb{F}_{p^n}$ avec $p\geqslant 5$. La condition $j_E=1728$ nous permet alors de restreindre notre étude aux courbes d'équation

$$E: Y^2=X^3+a_4X,\ a_4\in\mathbb{F}_{p^n}^*, \tag{2.5}$$

car il est facile de s'y ramener par un changement linéaire de variable. Comme pour le cas $j_E=0$, deux types de corps finis sont à considérer : $p^n\equiv\pm 1 \bmod 4$. Pour $p^n\equiv 1 \bmod 4$, un autre théorème de Katre et Rajwade [55] nous tire d'affaire.

E	n	$\#E(\mathbb{F}_{3^n})$	Structure de groupe
$Y^2 = X^3 + X$	impair	$p^n + 1$	cyclique
$Y^2 = X^3 - X$	impair	$p^n + 1$	$\mathbb{Z}_{/((p^n+1)/2)\mathbb{Z}} \oplus \mathbb{Z}_{/2\mathbb{Z}}$
$Y^2 = X^3 - X + \delta$	impair	$p^n + 1 + (-1)^{(n-1)/2}\sqrt{3p^n}$	cyclique
$Y^2 = X^3 - X - \delta$	impair	$p^n + 1 + (-1)^{(n+1)/2}\sqrt{3p^n}$	cyclique
$Y^2 = X^3 - X$	pair	$p^n + 1 + (-1)^{(n+2)/2}2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}\pm1)\mathbb{Z}} \oplus \mathbb{Z}_{/(\sqrt{p^n}\pm1)\mathbb{Z}}$
$Y^2 = X^3 - \gamma^2 X$	pair	$p^n + 1 + (-1)^{n/2}2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}\pm1)\mathbb{Z}} \oplus \mathbb{Z}_{/(\sqrt{p^n}\pm1)\mathbb{Z}}$
$Y^2 = X^3 - \gamma X$	pair	$p^n + 1$	cyclique
$Y^2 = X^3 - \gamma^3 X$	pair	$p^n + 1$	cyclique
$Y^2 = X^3 - X + \delta$	pair	$p^n + 1 + (-1)^{n/2}\sqrt{p^n}$	cyclique
$Y^2 = X^3 - \gamma^2 X + \gamma^3\delta$	pair	$p^n + 1 + (-1)^{(n+2)/2}\sqrt{p^n}$	cyclique

FIG. 2.3 – Courbes supersingulières définies sur $\mathbb{F}_{3^n}$.

Théorème 32. *Si $\mathbb{F}_{p^n}$ est un corps fini avec $p^n \equiv 1 \bmod 4$, alors le nombre de points de la courbe elliptique $E : Y^2 = X^3 + a_4X$ définie sur $\mathbb{F}_{p^n}$ est*

$$p^n + 1 + \Phi_2(a_4) \text{ avec } \Phi_2(a) = \begin{cases} -2s \text{ si } a \text{ est une puissance quatrième,} \\ 2s \text{ si } a \text{ est un carré et pas une puissance quatrième,} \\ 2t \text{ si } a \text{ n'est pas un carré,} \end{cases}$$

où s et t sont définis, si $p \equiv -1 \bmod 4$ par $s = (-p)^{n/2}$, $t = 0$ et sinon, par $p^n = s^2 + t^2$, p ne divise pas s, $s \equiv 1 \bmod 4$ et lorsque a n'est pas un carré, $a^{(p^n-1)/4} \equiv s/t \bmod p$.

Pour déterminer explicitement s et t, nous procédons comme pour le cas $j_E = 0$. Il suffit de trouver deux entiers u et v tels que $p = u^2 + v^2$ pour $p \equiv 1 \bmod 4$ avec l'algorithme de Cornacchia à partir desquels nous obtenons un entier algébrique $s + t\sqrt{-1}$ de norme p^n dans $\mathbb{Q}(\sqrt{-1})$.

Pour $p^n \equiv -1 \bmod 4$, la courbe est supersingulière et en suivant encore les travaux de Morain [92], nous avons le théorème suivant.

Théorème 33. *Si $\mathbb{F}_{p^n}$ est un corps fini avec $p^n \equiv -1 \bmod 4$, alors le nombre de points de la courbe elliptique $E : Y^2 = X^3 + a_4X$ définie sur $\mathbb{F}_{p^n}$ est $p^n + 1$.*

2.2.3 Autres invariants supersinguliers

Comme expliqué dans l'introduction, les seules courbes avec un j-invariant différent de 0 et 1728 qui peuvent éventuellement perturber les variantes de l'algorithme de Schoof sont les courbes à multiplication complexe par un ordre dans un corps quadratique imaginaire à petit nombre de classes. Néanmoins les courbes supersingulières ont des liens étroits avec celles à multiplication complexe et il est donc intéressant d'en obtenir la cardinalité directement.

Outre les résultats de la section 2.1, ces courbes satisfont un certain nombre de propriétés remarquables qui peuvent nous aider à les détecter. Ainsi, leur cardinalité et la structure de groupe de leur ensemble de points sont caractéristiques comme le montre le tableau 2.4 [102].

D'autre part leur j-invariant appartient toujours au sous-corps d'ordre p^2 de $\mathbb{F}_{p^n}$ [111, p. 137].

n	$\#E(\mathbb{F}_{p^n})$	Structure de groupe
impair	p^n+1	cyclique ou $\mathbb{Z}_{/((p^n+1)/2)\mathbb{Z}} \oplus \mathbb{Z}_{/2\mathbb{Z}}$
pair	p^n+1	cyclique
	$p^n+1\pm\sqrt{p^n}$	cyclique
	$p^n+1\pm 2\sqrt{p^n}$	$\mathbb{Z}_{/(\sqrt{p^n}\pm 1)\mathbb{Z}} \oplus \mathbb{Z}_{/(\sqrt{p^n}\pm 1)\mathbb{Z}}$

FIG. 2.4 – Courbes supersingulières définies sur $\mathbb{F}_{p^n}$, $p \geqslant 5$.

Chapitre 3

Algorithme de Schoof

Avec la publication en 1985 d'un algorithme déterministe de complexité en $O(\log^8 q)$ pour calculer le nombre de points d'une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ [101], Schoof offrait une alternative fort alléchante aux méthodes de complexité en $O(q^{1/4}\log^2 q)$ connues jusqu'alors. En effet, avec par exemple l'algorithme "des pas de bébé et de géant" de Shanks [107], il n'est pas envisageable d'avoir $q > 10^{30}$. Néanmoins, l'algorithme de Schoof n'aurait pas pris l'importance que nous lui connaissons aujourd'hui sans d'une part, les travaux d'Atkin qui, au travers de deux communications importantes [4, 5], montre comment calculer le nombre de points d'une courbe définie sur $\mathbb{F}_{10^{275}+693}$, et d'autre part, les travaux d'Elkies [39] qui complètent ceux d'Atkin. À l'invitation d'Atkin qui écrit en 1992, "since there is now considerable variety in the mathematics and programming involved in the problem, it is to be hoped that more people will try their hand at it", de nombreuses améliorations ont suivi, notamment l'obtention de la cardinalité d'une courbe elliptique modulo des puissances de nombres premiers [34, 33]. Tous ces travaux forment le corps d'un algorithme probabiliste appelé l'algorithme `SEA` dont la complexité est de $O(\log^6 q)$.

Notre but est ici de donner un panorama relativement complet de ces idées en nous plaçant délibérément dans un corps fini $\mathbb{F}_q$ *quelconque*. Nous serons ainsi mieux à même d'appréhender l'utilité des algorithmes de la partie II. Une fois rappelé comment s'applique la méthode de Shanks aux courbes elliptiques (section 3.1), nous faisons un rappel de l'algorithme original de Schoof (section 3.2) avant d'expliquer en quoi les méthodes efficaces de calcul d'équations modulaires développées par Atkin améliorent sensiblement cet algorithme (section 3.3). Enfin, nous expliquons comment, à partir de ces équations modulaires, la détermination d'isogénies préconisée par Elkies permet aux implantations les plus optimisées pour $\mathbb{Z}/p\mathbb{Z}$ comme celle de Morain [91] d'atteindre des corps aussi gros que $\mathbb{F}_{10^{499}+153}$ (section 3.4).

Dans chacun de ces paragraphes, nous donnons, outre la complexité asymptotique des algorithmes, un exemple détaillé d'application pour un corps fini de caractéristique 2. Par contre, la mise en œuvre efficace de certaines de ces idées nécessite des optimisations qui ne changent pas la complexité asymptotique de l'algorithme `SEA` et que nous décrivons seulement dans le chapitre 11.

Remarque : Pour une courbe elliptique donnée, la plupart de ces algorithmes retournent, non pas sa cardinalité, mais en toute rigueur, un multiple m de l'ordre d'un de ses points. À quelques très rares exceptions près que nous n'avons personnellement jamais rencontrées en pratique, il se trouve que m est réellement le nombre de points recherché dès que le corps fini de définition est de taille conséquente (plus de 10^{50} éléments). Certes, Atkin a tenté d'en donner une preuve qui ne nécessite pas la factorisation de grands entiers [5], mais il nous faut bien admettre que les seules méthodes connues à ce jour, pour le prouver *indépendamment* de l'algorithme utilisé pour obtenir m, nécessitent la factorisation de m [27, p. 396] [103]. Or, comme nous sommes aujourd'hui en mesure de calculer des nombres de points de largement plus de 200 chiffres (cf. chapitre 12), une telle preuve est hors de notre portée. Dans la suite, nous évacuons donc ce problème même si nous en gardons à l'esprit "l'éventuelle possibilité".

3.1 Algorithme "Pas de bébé et pas de géant"

3.1.1 Algorithme

Comme l'ensemble des points d'une courbe elliptique E définie sur corps fini $\mathbb{F}_q$ est un groupe fini dont on peut borner l'ordre, il est naturel d'utiliser l'algorithme des "pas de bébé et pas de géant" proposé par Shanks [107]. Dans ce cadre, cela revient à rechercher l'entier c de l'intervalle $[-2\sqrt{q}, 2\sqrt{q}]$ tel que pour un point P choisi aléatoirement dans $E(\mathbb{F}_q)$, nous ayons

$$(q+1-c)P = 0.$$

Une énumération naïve nécessiterait $O(\sqrt{q})$ opérations arithmétiques dans $\mathbb{F}_q$. Nous améliorons grandement cette complexité en alliant l'idée de Shanks au fait que le calcul d'un opposé d'un point de E est immédiat (cf. théorème 9). Ainsi, nous remplaçons la recherche de c par celle de l'entier c_0 de $\{0, \ldots, 2\lceil \sqrt[4]{q} \rceil\}$ et celle de l'entier c_1 de $\{-\lceil \sqrt[4]{q} \rceil/2, \ldots, \lceil \sqrt[4]{q} \rceil/2\}$ tels que

$$(q+1-c_1 4\lceil \sqrt[4]{q} \rceil)P = \pm c_0 P.$$

Lorsque cela se produit pour un couple $(\pm c_0, c_1)$, nous posons $c = \pm c_0 + c_1 4\lceil \sqrt[4]{q} \rceil$ et $q+1-c$ est un multiple de l'ordre de P.

Précisément, l'algorithme procède en deux phases. Dans la phase des "pas de géant", nous stockons dans une table les $\lceil \sqrt[4]{q} \rceil + 1$ points $(q+1-c_1 4\lceil \sqrt[4]{q} \rceil)P$. Puis, dans la phase des "pas de bébé", nous calculons les points $c_0 P$ pour c_0 variant de 0 à $2\lceil \sqrt[4]{q} \rceil$ en vérifiant pour chaque c_0 que les points $c_0 P$ et $-c_0 P$ correspondent à une entrée de la table des points de la première phase.

Remarquons que les bornes données ci-dessus sont choisies de façon à ce qu'en moyenne les temps des phases pas de bébé et pas de géant soient sensiblement égaux.

Complexité : comme nous espérons, en moyenne, une coïncidence après $\lceil \sqrt[4]{q} \rceil$ telles additions, cela conduit à un coût moyen de $2\lceil \sqrt[4]{q} \rceil$ additions dans $E(\mathbb{F}_q)$. À ce coût, s'ajoute la recherche pour chaque point $c_0 P$ d'une coïncidence dans la table. En triant au préalable cette table ou encore en la hachant, une telle recherche peut ce faire en $O(\log q)$, ce qui est négligeable par rapport aux sommes dans $E(\mathbb{F}_q)$. Cet algorithme a donc une complexité totale de $O(q^{1/4} \log^2 q)$.

3.1.2 Exemple

Brièvement, nous allons calculer, dans $\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8+t^4+t^3+t+1)$, la cardinalité de la courbe

$$E \ : Y^2 + XY = X^3 + t^2 + t + 1.$$

Par commodité, nous utilisons la notation $\overline{\tau_0 + \tau_1 2 + \cdots + \tau_7 2^7} = \tau_0 + \tau_1 t + \cdots \tau_7 t^7$ (ainsi $\overline{12} = t^3 + t^2$ puisque $12 = 2^3 + 2^2$). Prenons par exemple $P = (\overline{1}, \overline{2})$, les points de la première phase sont

$$225P = (\overline{16}, \overline{190}),\ 241P = (\overline{251}, \overline{226}),\ 257P = (\overline{166}, \overline{219}),$$
$$273P = (\overline{1}, \overline{2}) \text{ et } 289P = (\overline{0}, \overline{249}).$$

Nous avons alors $273P = P$, d'où, après vérification, $\#E_{\overline{7}}(\mathbb{F}_{2^8}) = 2^8 + 1 + 15 = 272$.

3.2 Algorithme original de Schoof

L'algorithme de Schoof [101] auquel nous nous intéressons dans cette section fut le premier algorithme à complexité polynomiale en $\log q$ pour déterminer la cardinalité d'une courbe elliptique. Après en avoir rappelé le principe, nous l'appliquons à l'exemple de la section précédente.

3.2.1 Algorithme

Le Frobenius ϕ_E (cf. définition 13) d'une courbe elliptique E définie sur $\mathbb{F}_q$ par

$$E \ : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6,$$

entretient des liens étroits avec $E(\mathbb{F}_q)$. Rappelons que son équation caractéristique est

$$\phi_E^2 - [c]_E \circ \phi_E + [q]_E = 0 \text{ où } \#E(\mathbb{F}_q) = q + 1 - c.$$

L'idée de Schoof est de restreindre cette relation aux sous-groupes $E[\ell]$ de $E(\bar{\mathbb{F}}_q)$, ce qui donne

$$\forall \ell \in \mathbb{N}^*,\ \phi^2_{E[\ell]} + [q \bmod \ell]_{E[\ell]} = [c \bmod \ell]_{E[\ell]} \circ \phi_{E[\ell]}, \tag{3.1}$$

ou encore

$$\forall \ell \in \mathbb{N}^*, \forall (X,Y) \in E[\ell],\ (X^{q^2}, Y^{q^2}) + [q \bmod \ell](X,Y) = [c \bmod \ell](X^q, Y^q).$$

L'algorithme consiste alors à calculer la partie gauche de l'équation (3.1) pour un point P d'ordre ℓ, dans $E(\bar{\mathbb{F}}_q)$ puis à calculer $[\theta]_{E[\ell]}(\phi_{E[\ell]}(P))$ pour θ dans $\mathbb{F}_\ell$ jusqu'à ce que l'équation (3.1) soit vérifiée. Lorsque cela est le cas, nous avons $c \equiv \theta \mod \ell$. Il ne nous reste alors plus qu'à répéter ce calcul pour des entiers ℓ premiers entre eux et vérifiant

$$\prod \ell > 4\sqrt{q}$$

pour trouver c avec le théorème chinois.

La mise en œuvre de l'algorithme découle du théorème 11. Pour un entier ℓ donné, nous calculons le polynôme de ℓ-division $f_\ell(X)$ à l'aide des formules de la définition 8. Puis nous considérons la courbe $E(\mathbb{Y})$ obtenue par extension des scalaires à

$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6) \text{ avec } \mathbb{X} = \mathbb{F}_q[X]/(f_\ell(X)).$$

Le point $P = (X, Y)$ de $E(\mathbb{Y})$ est alors un point de ℓ-division à partir duquel nous pouvons exploiter la relation (3.1) comme expliqué précédemment. Remarquons ici que nous ne savons pas *a priori* si le polynôme $f_\ell(X)$ est irréductible sur $\mathbb{F}_q$ et donc $\mathbb{X}$ et $\mathbb{Y}$ ne sont pas toujours des corps finis. Même si cela est peu probable, il peut donc arriver qu'en cours de calcul un inverse nécessaire à l'addition de deux points n'existe pas. Dans ce cas, il est cependant aisé de trouver un facteur de $f_\ell(X)$ ou de $Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6$ modulo lequel nous pouvons, dans un premier temps, réduire les points précédemment calculés afin d'assurer, dans un deuxième temps, la continuation de l'algorithme.

Complexité : la complexité des calculs est bien entendu fortement conditionnée par le degré de l'extension $\mathbb{X}$, autrement dit, par le degré du polynôme f_ℓ. Il est donc naturel de prendre des entiers ℓ les plus petits possibles. Comme d'autre part nous devons, pour appliquer le théoreme chinois, avoir les ℓ premiers entre eux, il est finalement judicieux de choisir des puissances de petits nombres premiers. Ainsi, $O(\log q)$ entiers ℓ de l'ordre de $O(\log q)$ sont nécessaires pour mener à bien le calcul de la cardinalité de E. De plus, $\mathbb{X}$ est une extension de degré $O(\log^2 q)$ sur $\mathbb{F}_q$ puisque $f_\ell(X)$ est de degré

- $(\ell^2 - 1)/2$ si ℓ est impair et premier avec p,
- $p^i(p^i - 1)/2$ si $\ell = p^i$,
- $2^{2i-1} - 2$ si $\ell = 2^i$ et $p \neq 2$.

Ainsi le produit de deux eléments de $\mathbb{X}$ est réalisé en $O(\log^4 q)$ multiplications dans $\mathbb{F}_q$ et donc, $O(\log^5 q)$ multiplications sont nécessaires pour calculer $\phi_E(P)$ ou $\phi_E^2(P)$, et autant pour trouver $c \mod \ell$ à l'aide de $O(\ell)$ sommes dans $E(\mathbb{Y})$. Le coût de l'algorithme de Schoof est en conséquence de $O(\log^6 q)$ multiplications dans $\mathbb{F}_q$, ce qui, avec des algorithmes classiques, conduit à une complexité totale de $O(\log^8 q)$.

3.2.2 Exemple

Nous reprenons les notations de l'exemple de la section 3.1 où nous nous proposons de calculer c mod 3 avec l'algorithme de Schoof. Nous partons pour cela du polynôme $f_3(X) = X^4 + X^3 + \overline{7}$ à partir duquel nous construisons les extensions

$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + XY + X^3 + \overline{7}) \text{ avec } \mathbb{X} = \mathbb{F}_{2^8}[X]/(f_3(X)).$$

Nous avons alors dans $\mathbb{Y}$, $[q \bmod 3](X, Y) = (X, Y)$ où $q = 2^8$ et

$$\begin{aligned} X^q &= \overline{235}X^3 + \overline{163}X^2 + \overline{73}X + \overline{1}, \\ Y^q &= \overline{2}X^3 + \overline{87}X^2 + \overline{136}X + \overline{122} + Y\left(\overline{209}X^3 + \overline{58}X^2 + \overline{163}X + \overline{73}\right). \end{aligned}$$

Donc $X^{q^2} = X, Y^{q^2} = X + Y$, et $(X^{q^2}, Y^{q^2}) + [q](X, Y) = O_{E_{\overline{7}}}$. D'où nous déduisons,

$$c = 0 \bmod 3.$$

3.3 Premières optimisations d'Atkin

Très rapidement, il est apparu que les degrés des polynômes mis en jeu dans l'algorithme de Schoof, en particulier le degré $(\ell^2-1)/2$ des polynômes f_ℓ, sont trop élevés pour espérer calculer la cardinalité d'une courbe elliptique E modulo des nombres premiers ℓ plus grands que 40. Ceci nous limite à l'utilisation de corps finis à moins de 10^{40} éléments. Néanmoins, les premières améliorations d'Atkin nous permettent d'obtenir des informations partielles sur c modulo des entiers ℓ bien plus grands que 40.

À cet effet, Atkin a su faire le lien entre l'action du Frobenius ϕ_E de E sur $E[\ell]$ et le degré des extensions dans lesquelles sont définies des courbes isogènes de degré ℓ à E [4]. Une fois rappelées ces idées dans une première partie, il sera clair que le calcul d'équations modulaires en est le point crucial, ce qui a motivé le calcul de trois types d'équations modulaires que nous passons rapidement en revue dans une deuxième partie, avant de donner dans une troisième partie l'algorithme qui découle de ces résultats puis de finir par un exemple d'application.

3.3.1 Action du Frobenius sur $E[\ell]$

Plaçons nous dans le cas où ℓ est un nombre premier impair distinct de p. D'après, le théorème 26, $E[\ell]$ est engendré par deux points P_1 et P_2. Cet ensemble est alors la réunion de $\ell + 1$ sous-groupes $\mathbb{E}_i$ d'ordre ℓ définis sur des extensions de degré d_i de $\mathbb{F}_q$ ($\mathbb{E}_1 =< P_1 >$, $\mathbb{E}_2 =< P_2 >$ et $\mathbb{E}_i =< P_1 + (i-2)P_2 >$ pour $i = 3, \ldots, \ell + 1$).

Classiquement, chaque $\mathbb{E}_i$ est aussi le noyau d'une isogénie de degré ℓ définie à partie de E (cf. par exemple les formules de Vélu du chapitre 4 pour s'en convaincre). Il se trouve que le degré r (un diviseur de certains d_i) de l'extension de $\mathbb{F}_q$ dans laquelle est définie une telle isogénie conditionne fortement c mod ℓ comme nous nous proposons de l'expliquer ici.

En fait, le lien entre r et c découle du théorème suivant [103].

Théorème 34. *Soient E une courbe elliptique définie sur $\mathbb{F}_q$ supposée non supersingulière et à j-invariant distinct de 0 et 1728, et $\mathbb{E}$ un sous-groupe d'ordre ℓ de $E[\ell]$. Alors le degré r de la plus petite extension sur laquelle on peut définir une courbe isogène à E via une isogénie de noyau $\mathbb{E}$ est égal au plus petit entier ρ tel que $\phi^\rho_{E[\ell]}(\mathbb{E}) = \mathbb{E}$.*

Or l'action de $\phi_{E[\ell]}$ sur $E[\ell]$ est déterminée en grande partie par son polynôme minimal $X^2 - (c \bmod \ell)X + (q \bmod \ell)$. Précisément, soient κ_1 et κ_2, les racines dans $\mathbb{F}_{\ell^2}$ de $X^2 - cX + q$ et d_1, d_2, leurs ordres respectifs, nous avons trois cas à distinguer suivant la diagonalisation du Frobenius $\phi_{E[\ell]}$ dans $E[\ell] \simeq \mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$:

1. si $(\kappa_1, \kappa_2) \in \mathbb{F}_\ell^2$, $\kappa_1 \neq \kappa_2$. Il existe alors P_1 et P_2, deux points de $E[\ell]$, qui satisfont

$$\phi_{E[\ell]}(P_1) = \kappa_1 P_1 \text{ et } \phi_{E[\ell]}(P_2) = \kappa_2 P_2.$$

Il n'est pas difficile de montrer que P_1 et P_2 appartiennent à deux sous-groupes d'ordre ℓ distincts, $\mathbb{E}_1$ et $\mathbb{E}_2$, et donc engendrent $E[\ell]$. Nous avons déjà $\phi^{d_1}_{E[\ell]}(P_1) = P_1$ et $\phi^{d_2}_{E[\ell]}(P_1) = P_2$, donc $\mathbb{E}_1 \subset E(\mathbb{F}_{q^{d_1}})$ et $\mathbb{E}_2 \subset E(\mathbb{F}_{q^{d_2}})$ où d_1 et d_2 divisent $\ell - 1$. D'autre part, comme

$$\forall Q \in E[\ell],\ \text{il existe un couple } (x_1, x_2) \in \mathbb{F}_\ell^2 \text{ tels que } Q = x_1P_1 + x_2P_2,$$

nous avons pour chaque sous-groupe $\mathbb{E}_i$, $\mathbb{E}_i \subset E(\mathbb{F}_{q^d})$ avec $d = \text{ppcm}(d_1, d_2)$. De plus, pour tout entier k, $\phi^k_{E[\ell]}(Q) = \kappa_1^k x_1 P_1 + \kappa_2^k x_2 P_2$. Le plus petit entier k pour lequel $\phi^k_{E[\ell]}(Q)$ est un multiple de Q est donc l'ordre r de κ_1/κ_2 dans $\mathbb{F}_\ell$ puisqu'alors $\kappa_1^r = \kappa_2^r$ et en définitive,

$$\forall i \in \{3, \ldots, \ell + 1\}, \phi^r_{E[\ell]}(\mathbb{E}_i) = \mathbb{E}_i.$$

auquel s'ajoute par définition $\phi_{E[\ell]}(\mathbb{E}_1) = \mathbb{E}_1$ et $\phi_{E[\ell]}(\mathbb{E}_2) = \mathbb{E}_2$.

2. si $(\kappa_1, \kappa_2) \in \mathbb{F}^2_{\ell^2}$. Notons qu'alors $\kappa_1 \neq \kappa_2$ et $\kappa_2 = \kappa_1^\ell$. Par un raisonnement analogue à celui du cas précédent, mais ici appliqué à $E[\ell^2]$, nous avons

$$\forall i \in \{1, \ldots, \ell + 1\}, \mathbb{E}_i \subset E(\mathbb{F}_{q^d}) \text{ avec } d = \text{ppcm}(d_1, d_2).$$

Soit r le plus petit entier ρ tel que $\kappa_1^\rho \in \mathbb{F}_\ell$, alors r divise $\ell + 1$ et $d = rs$ où s divise $\ell - 1$. De plus il n'est pas difficile de montrer que r est aussi l'ordre de κ_1/κ_2 dans $\mathbb{F}_{\ell^2}$, ce qui permet par un raisonnement analogue à celui du cas précédent de déduire

$$\forall i \in \{1, \ldots, \ell + 1\}, \phi^r_{E[\ell]}(\mathbb{E}_i) = \mathbb{E}_i.$$

3. si $(\kappa_1, \kappa_2) \in \mathbb{F}^2_\ell$, $\kappa_1 = \kappa_2$. Nous avons déjà $c = \pm 2\sqrt{q} \bmod \ell$. D'autre part, il existe un point P_1 de $E[\ell]$ associé à un sous-groupe $\mathbb{E}_1$ qui satisfait $\phi_{E[\ell]}(P_1) = \kappa_1 P_1$, donc

$$\mathbb{E}_1 \subset E(\mathbb{F}_{q^{d_1}}) \text{ et } \phi_{E[\ell]}(\mathbb{E}_1) = \mathbb{E}_1.$$

Par ailleurs, il existe un point $P_2 \notin \mathbb{E}_1$ tel que $\phi_{E[\ell]}(P_2) = \kappa_1 P_2 + lP_1$. Deux sous-cas sont donc à distinguer.

a. Si $l = 0$: il n'est alors pas difficile de démontrer que

$$\forall i \in \{2, \ldots, \ell + 1\}, \mathbb{E}_i \subset E(\mathbb{F}_{q^{d_1}}) \text{ et } \phi_{E[\ell]}(\mathbb{E}_i) = \mathbb{E}_i.$$

Remarquons de plus qu'ici, $c = \pm 2\sqrt{q} \bmod \ell^2$.

b. Si $l \neq 0$: alors $\phi^{d_1\ell}_{E[\ell]}(P_2) = P_2$, d'où

$$\forall i \in \{2, \ldots, \ell + 1\}, \mathbb{E}_i \subset E(\mathbb{F}_{q^{d_1\ell}}) \text{ et } \phi^\ell_{E[\ell]}(\mathbb{E}_i) = \mathbb{E}_i.$$

L'intérêt est qu'il est possible, sans connaître c, d'obtenir le type de factorisation de $X^2 - cX + q$ dans $\mathbb{F}_\ell[X]$ et l'entier r correspondant, par l'intermédiaire du polynôme modulaire $\Phi_\ell(X, Y)$ comme l'énonce le théorème 35. Nous étudions plus en détail ce polynôme dans la section suivante, mais disons déjà qu'il s'agit d'un polynôme symétrique à coefficients dans $\mathbb{Z}$ égal à $X^{\ell+1} - X^\ell Y^\ell + Y^{\ell+1}$ plus des termes de la forme $X^i Y^j$ avec $i, j \leqslant \ell$ et $i + j < 2\ell$. Ce polynôme a la propriété que pour tout corps $\mathbb{K}$ de caractéristique distincte de ℓ et tout invariant j_E d'une courbe elliptique E à coefficients dans $\mathbb{K}$, les $\ell + 1$ zéros dans $\overline{\mathbb{K}}$ de $\Phi_\ell(j_E, Y)$ soient les invariants de $\ell + 1$ courbes isogènes associés aux $\ell + 1$ sous-groupes $\mathbb{E}_i$ d'ordre ℓ de $E[\ell]$.

Théorème 35. *Soit E une courbe non supersingulière de $\mathbb{F}_q$ à j-invariant distinct de 0 et 1728. Soit $f_1(Y) \cdots f_s(Y)$ la factorisation sur $\mathbb{F}_q[Y]$ de $\Phi_\ell(j_E, Y)$ en polynômes irréductibles. Alors nous avons trois possibilités pour les degrés des polynômes $f_1, f_2, \ldots, f_s$:*

(i)

$$\begin{cases} (a)\ 1, \ell \\ (b)\ 1, 1, 1, \ldots, 1 \end{cases} \quad \textit{si } c^2 - 4q = 0 \bmod \ell.$$

(ii) $1, 1, r, \ldots, r$ *si* $c^2 - 4q$ *est un carré dans* $\mathbb{F}_\ell$.

(iii) $r, \ldots, r$ *avec* $r > 1$ *si* $c^2 - 4q$ *n'est pas un carré dans* $\mathbb{F}_\ell$.

Dans tous les cas, r *est le plus petit entier tel que*

$$\forall P \in E[\ell], \exists \kappa \in \mathbb{N}, \phi^r(P) = \kappa P.$$

En conséquence, si nous connaissons pour un ℓ donné, la façon dont se factorise $\Phi_\ell(j_E, Y)$, nous en déduisons l'entier r correspondant et donc comment $X^2 - cX + q$ se factorise dans $\mathbb{F}_\ell[X]$. Il ne reste alors plus qu'à conserver pour $c \mod \ell$ les candidats θ pour lesquels le rapport des racines de $X^2 - \theta X + q$ est d'ordre r. Par exemple, si dans le cas (iii), nous avons $r = 2$, seul $\theta = 0$ convient.

3.3.2 Équations modulaires

Les polynômes modulaires $\Phi_\ell(X, Y)$ ont en pratique deux inconvénients majeurs ;
- la taille de leurs coefficients croît rapidement en fonction de ℓ,
- leur degré en Y est trop élevé.

En se plaçant sur $\mathbb{C}$, cette équation est en fait l'équation minimale du j-invariant d'une courbe isogène à E considéré comme "fonction modulaire du sous-groupe $\Gamma_0(\ell)$ de $\mathrm{SL}_2(\mathbb{Z})$". Nous devons à Atkin [4] l'idée d'avoir, dans un premier temps, remplacé l'utilisation de cette équation par celle d'une équation minimale pour une autre fonction modulaire, $f_\ell(\tau)$, qui comporte, elle, des coefficients entiers bien plus petits, puis, dans un deuxième temps, remplacé $\Gamma_0(\ell)$ par un autre sous-groupe $\Gamma_0^*(\ell)$ de $\mathrm{SL}_2(\mathbb{Z})$ et $f_\ell(\tau)$ par une fonction modulaire déterminée pour chaque ℓ de façon à avoir simultanément des tailles de coefficients et des degrés en Y petits.

Nous utilisons pour notre part les équations précalculées par Morain jusqu'à $\ell \simeq 1000$. Nous ne décrirons donc pas ici en détail les méthodes prônées par Atkin pour leurs calculs et précisément décrites par ailleurs dans [91, 94]. Par contre, nous nous proposons de rassembler ici les quelques rappels mathématiques extraits de [104, 100, 85] et nécessaires à la compréhension de ces méthodes par un lecteur non averti.

Fonctions modulaires

Nous nous plaçons sur $\mathbb{C}$. Nous avons vu au chapitre 2 qu'une courbe elliptique E pouvait être considérée comme le quotient de $\mathbb{C}$ par un réseau $\Lambda = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}$. Par un éventuel réordonnancement de ω_1 et ω_2, la partie imaginaire de $\tau = \omega_2/\omega_1$ est strictement positive et comme de plus la multiplication de Λ par tout complexe non nul correspond à une courbe isomorphe à E, nous pouvons supposer, en toute généralité, qu'une courbe elliptique E est isomorphe au quotient de $\mathbb{C}$ par le réseau $\mathbb{Z} + \tau\mathbb{Z}$ où τ décrit le demi plan de Poincaré $\mathcal{H} = \{z \in \mathbb{C}, \mathrm{Im} z > 0\}$. Il est classique d'étendre $\mathcal{H}$ en

$$\mathcal{H}^* = \{z \in \mathbb{C}, \mathrm{Im} z > 0\} \cup \mathbb{Q} \cup \{i\infty\}.$$

L'invariant $j(\tau)$ d'un telle courbe E_τ se développe en série de Fourier comme suit.

Proposition 6. *Il existe des coefficients entiers* c_n *tels que si* $x = e^{2i\pi\tau}$, *nous avons pour tout* $\tau \in \mathcal{H}^*$,

$$j(\tau) = \frac{1}{x} + 744 + \sum_{n \geqslant 1} c_n x^n.$$

De nombreuses méthodes existent pour calculer ce développement. Par exemple, il peut être obtenu à partir des développements en séries de Fourier donnés dans [104] pour les séries d'Eisenstein G_{2k} (cf. équation (3.4)).

Pour toute matrice M de $\mathrm{SL}_2(\mathbb{Z}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix}, (a, b, c, d) \in \mathbb{Z}^4, ad - bc = 1 \right\}$, nous avons de plus le réseau généré par $a\tau + b$ et $c\tau + d$ qui est isomorphe à celui de E_τ. Il est donc intéressant à ce stade de déterminer un sous-ensemble $\mathcal{F}$ de $\mathcal{H}^*$ tels que pour tout $\tau \in \mathcal{H}^*$, τ et $M(\tau) = \dfrac{a\tau + b}{c\tau + d}$ ne puissent simultanément appartenir à l'intérieur de $\mathcal{F}$. Cela conduit à la notion de "domaine fondamental".

Définition 16. *Soit Γ un sous-groupe de $\mathrm{SL}_2(\mathbb{Z})$.*

Deux éléments τ et τ' de $\mathcal{H}^*$ sont dits équivalents si et seulement s'il existe $M \in \Gamma$ tel que $\tau' = M(\tau)$. Nous notons $\mathcal{H}^*/\Gamma$, l'ensemble des classes d'équivalence de cette relation.

Un ensemble fondamental pour Γ est un ensemble qui ne contient qu'un unique représentant de chaque classe d'équivalence de $\mathcal{H}^*/\Gamma$.

Un ensemble $\mathcal{F}$ est appelé un domaine fondamental pour Γ si son intérieur est un ensemble fondamental pour Γ et si $(\tau, M(\tau)) \in \mathcal{F}^2$ pour $M \in \Gamma \setminus \{\mathrm{Id}\}$ implique que τ est sur la frontière de $\mathcal{F}$.

Ainsi, il est classique de donner, en exemple, le domaine fondamental $\mathcal{F}$ de la figure 3.1 pour $\mathrm{SL}_2(\mathbb{Z})$.

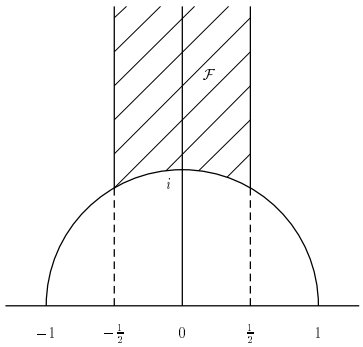


FIG. 3.1 – Domaine fondamental pour $\mathrm{SL}_2(\mathbb{Z})$.

Comme E_τ et $E_{M(\tau)}$ sont isomorphes, nous avons

$$\forall \tau \in \mathcal{H}^*, \forall M \in \mathrm{SL}_2(\mathbb{Z}), j(M(\tau)) = j(\tau).$$

Cette propriété fait de $j(\tau)$ ce que l'on appelle une fonction modulaire pour $\mathrm{SL}_2(\mathbb{Z})$. Une telle fonction est précisément définie ainsi.

Définition 17. *Soit Γ un sous-groupe d'indice fini de $\mathrm{SL}_2(\mathbb{Z})$. Une fonction f de $\mathcal{H}^*$ dans $\mathbb{C} \cup \mathbb{Q} \cup \{i\infty\}$ est une fonction modulaire pour Γ si et seulement si*

1. *f est méromorphe sur $\mathcal{H}^*$ (c'est-à-dire holomorphe sauf en les pôles) ;*
2. *pour tout $M \in \Gamma$, on a $f(M(\tau)) = f(\tau)$;*
3. *à une "pointe" $-d/c \in \mathbb{Q}$, $\mathrm{pgcd}(d, c) = 1$, f admet pour "$\mathrm{Im}\tau$ assez grand" un développement de la forme*

$$f(\tau) = \sum_{v \geqslant v_0} c_v e^{\frac{2\pi i}{\varkappa_1} A(\tau) v} \text{ avec } A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma \text{ et } \varkappa_1 \in \mathbb{N}.$$

Une fonction modulaire vérifie alors deux propriétés qui nous seront utiles dans la suite.

Proposition 7. *Si f est une fonction modulaire pour Γ et $M \in \mathrm{SL}_2(\mathbb{Z})$, alors la fonction*

$$\begin{array}{rccc} f|M & : & \mathcal{H}^* & \to & \mathbb{C} \cup \mathbb{Q} \cup \{i\infty\}, \\ & & \tau & \mapsto & f(M(\tau)), \end{array}$$

est une fonction modulaire pour $M^{-1}\Gamma M$.

Proposition 8. *Si f est une fonction modulaire pour Γ et si Γ' est un sous-groupe d'indice fini de Γ, alors f est une fonction modulaire pour Γ'.*

Enfin, nous avons le théorème important suivant.

Théorème 36. *Toute fonction modulaire f sur un sous-groupe Γ de $\mathrm{SL}_2(\mathbb{Z})$ d'indice fini μ satisfait une équation de degré μ,*

$$G(f,j) = \sum_{v=0}^{\mu} R_v(j) f^v = 0$$

où les coefficients $R_v(j)$ sont des fonctions rationnelles en $j(\tau)$ sur $\mathbb{C}$. Cette équation est en fait

$$G(X,j) = \prod_{v=1}^{\mu} (X - f|S_v) \tag{3.2}$$

où S_v sont les classes de $\mathrm{SL}_2(\mathbb{Z})/\Gamma$.

Ainsi, les équations modulaires utilisées dans l'algorithme `SEA` sont construites par application du théorème précédent à des groupes Γ et des fonctions $f(\tau)$ particuliers.

Polynôme modulaire

Pour un nombre premier ℓ, le polynôme modulaire $\Phi_\ell(X,Y)$ est l'équation minimale de la fonction modulaire $\tau \mapsto j(\ell\tau)$ sur le sous-groupe $\Gamma_0(\ell)$ de $\mathrm{SL}_2(\mathbb{Z})$ défini par

$$\Gamma_0(\ell) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix}, ad - bc = 1, c = 0 \bmod \ell \right\}.$$

L'intérêt de considérer ce sous-groupe de $SL_2(\mathbb{Z})$ est donné par la proposition suivante.

Proposition 9. *La fonction*

$$\begin{array}{rccc} j_\ell : & \mathcal{H}^* & \to & \mathbb{C} \cup \mathbb{Q} \cup \{i\infty\}, \\ & \tau & \mapsto & j(\ell\tau), \end{array}$$

est une fonction modulaire pour $\Gamma_0(\ell)$.

Pour pouvoir appliquer le théorème 36 à j_ℓ et $\Gamma_0(\ell)$, il ne reste plus qu'à déterminer la structure de $\mathrm{SL}_2(\mathbb{Z})/\Gamma_0(\ell)$. Ce qui est chose faite par la proposition suivante.

Proposition 10. *$\Gamma_0(\ell)$ est un sous-groupe de $\mathrm{SL}_2(\mathbb{Z})$ d'indice $\ell+1$ et les $\ell+1$ classes de $\mathrm{SL}_2(\mathbb{Z})/\Gamma_0(\ell)$ sont engendrées par*

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ et } \begin{pmatrix} 0 & -1 \\ 1 & v \end{pmatrix} \text{ pour } 0 \leqslant v < \ell.$$

Et donc le polynôme minimal de j_ℓ est donné par

$$G_\ell(X, j_\ell(\tau)) = (X - j(\ell\tau)) \prod_{v=0}^{\ell-1} \left(X - j\left(\frac{-\ell}{\tau + v} \right) \right).$$

On peut alors montrer que ce polynôme correspond au polynôme $\Phi_\ell(X,Y)$ que nous avons précédemment utilisé. Le calcul des coefficients de $G_\ell(X, j_\ell)$ se fait en partant de l'équation $G_\ell(X, j_\ell(-1/\tau))$ et en identifiant les développements en série des sommes de Newton des coefficients,

$$\sum_{v=0}^{\ell-1} j^r \left(\frac{\tau + v}{\ell} \right),$$

avec ceux des puissances de j puis en inversant le système triangulaire obtenu pour retrouver G. Par exemple, pour $\ell = 3$, nous avons

$$\begin{aligned} \Phi_3(X,Y) = {} & X^4 + Y^4 + 1855425871872000000000X + 452984832000000X^2 + 36864000X^3 \\ & + 1855425871872000000000Y - 770845966336000000XY + 8900222976000X^2Y \\ & - 1069956X^3Y + 452984832000000Y^2 + 8900222976000XY^2 + 2587918086X^2Y^2 \\ & + 2232X^3Y^2 + 36864000Y^3 - 1069956XY^3 + 2232X^2Y^3 - X^3Y^3. \end{aligned}$$

En règle générale, cette équation a des coefficients énormes, ce qui est lié au fait que la fonction j_ℓ a un pôle d'ordre trop grand à l'infini.

Équation modulaire canonique pour $X_0(\ell)$

La première idée d'Atkin fut de remplacer j_ℓ par la fonction f_ℓ définie par

$$f_\ell(\tau) = \ell^s \left(\frac{\eta(\ell\tau)}{\eta(\tau)}\right)^{2s} \text{ avec } s = \frac{12}{\text{pgcd}(12, \ell-1)} \text{ et } v = s(\ell-1)/12,$$

et

$$\eta(\tau) = x^{1/24} \prod_{n \geqslant 1} (1-x^n) \text{ où } x = e^{2i\pi\tau}.$$

On peut alors montrer, d'une part, que $f_\ell(\tau)$ admet le développement en série de Fourier suivant,

$$f_\ell(\tau) = x^v + \sum_{n \geqslant v+1} a_n x^n,$$

et d'autre part, la proposition suivante.

Proposition 11. *f_ℓ est une fonction modulaire sur $\Gamma_0(\ell)$.*

On appelle alors "équation modulaire canonique", le polynôme minimal de f_ℓ donné par

$$\Phi_\ell^c(X, j(\tau)) = (X - f_\ell(\tau)) \prod_{v=0}^{\ell-1} \left(X - f_\ell\left(\frac{-1}{\tau+v}\right)\right).$$

Nous aurons besoin dans l'algorithme `SEA` d'obtenir j_ℓ en fonction de j. Il suffit pour cela de se rendre compte, en utilisant les propriétés satisfaites par f_ℓ et j, que $\Phi_\ell^c(f_\ell, j) = 0$ est équivalent à $\Phi_\ell^c(\ell^s/f_\ell, j_\ell) = 0$.

En pratique, le calcul de Φ_ℓ^c a lieu de la même façon que pour Φ_ℓ, principalement, l'identification des sommes de Newton comme des polynômes en j. Pour $\ell = 5$, nous trouvons par exemple,

$$\Phi_5^c(X, Y) = X^6 + 30X^5 + 315X^4 + 1300X^3 + 1575X^2 + (-Y + 750)X + 125.$$

L'inconvénient de ces équations canoniques est que leur degré en Y augmente rapidement avec ℓ, ce qui est dû à la valuation v de f_ℓ qui augmente linéairement en fonction de ℓ.

Équation modulaire pour $X_0^*(\ell)$

Pour trouver des fonctions de valuations les plus petites possibles, Atkin utilise le groupe

$$\Gamma_0^*(\ell) = \Gamma_0(\ell) \cup w_\ell \Gamma_0(\ell)$$

où w_ℓ est l'involution d'Atkin Lehner définie sur $\mathcal{H}^*$ par $w_\ell(\tau) = -1/\ell\tau$. Il est d'usage de noter,

$$X_0^*(\ell) = X_0(\ell)/w_\ell.$$

Remarquons déjà que w_ℓ normalise $\Gamma_0(\ell)$, c'est-à-dire

$$w_\ell \Gamma_0(\ell) w_\ell^{-1} = \Gamma_0(\ell).$$

L'involution d'Atkin Lehner transforme donc les générateurs de $\text{SL}_2(\mathbb{Z})/\Gamma_0(\ell)$ en

$$\begin{pmatrix} 0 & -1 \\ \ell & 0 \end{pmatrix} \text{ et } \begin{pmatrix} 1 & v \\ \ell & 0 \end{pmatrix} \text{ pour } 0 \leqslant v < \ell.$$

Donc, une équation minimale pour une fonction f_ℓ^* sur $\Gamma_0^*(\ell)$ est

$$\Phi_\ell^*(X, j(\tau)) = (X - f_\ell^*(\tau)) \prod_{v=0}^{\ell-1} \left(X - f_\ell^*\left(\frac{\tau+v}{\ell}\right)\right),$$

et par application de l'involution d'Atkin Lehner, nous avons

$$\Phi_\ell^*(f_\ell^*, j) = \Phi_\ell^*(f_\ell^*, j_\ell \circ w_\ell) = 0.$$

La difficulté est ici de trouver des fonctions f_ℓ^* d'ordre petit. Une manière de procéder est de quotienter des combinaisons linéaires de fonctions θ définies comme

$$\theta_{a,b,c}(\tau) = \sum_{m,n} x^{am^2+bmn+cn^2}$$

par $\eta(\tau)\eta(\ell\tau)$. Par exemple, une fonction sur $X_0^*(11)$ est donnée par

$$\left(\frac{\theta_{1,1,3}(\tau)}{\eta(\tau)\eta(\ell\tau)}\right)^2 - 1 = \frac{1}{x} + 5 + 17x + 46x^2 + 116x^3.$$

Atkin a mis au point, pour un ℓ quelconque, un algorithme de génération de ces fonctions appelé "blanchiment" ainsi qu'un algorithme de génération des polynômes $\Phi_\ell^*(X, Y)$ que nous ne décrirons pas ici (cf. [91, 94]). Ainsi, l'équation modulaire pour $\ell = 11$ obtenue à partir de la fonction précédente est

$$\begin{aligned}\Phi_{11}^*(X,Y) = {} & X^{12} + X^{11}(-Y + 684) + X^{10}(55Y + 157410) + X^9(-1188Y + 12515580) \\ & + X^8(12716Y + 75763215) + X^7(-69630Y + 76077144) + X^6(177408Y - 207606564) \\ & + X^5(-133056Y - 34321320) + X^4(-132066Y + 418524975) + X^3(187407Y - 477130500) \\ & + X^2(-40095Y + 270641250) + X(-24300Y - 82012500) + Y^2 + 6750Y + 11390625.\end{aligned}$$

Factorisation des équations modulaires

L'algorithme `SEA` nécessite la factorisation des équations modulaires pour déterminer, tout d'abord, à quel cas de la proposition 35 le nombre premier courant ℓ correspond, puis ensuite, l'entier r. Dans cette dernière étape, il est possible de ne considérer que certains r d'après le théorème suivant [103].

Proposition 12. *Soit E une courbe non supersingulière de $\mathbb{F}_q$ à j-invariant distinct de 0 et 1728. Soit ℓ un nombre premier et s le nombre de facteurs irréductibles de $\Phi_\ell(X, j_E) \in \mathbb{F}_q[X]$. Alors*

$$(-1)^s = \left(\frac{q}{\ell}\right).$$

3.3.3 Algorithme

Une fois précalculés les polynômes modulaires, une implantation naïve des idées d'Atkin est quasi immédiate. D'après les résultats précédents, l'algorithme procède en deux étapes. Dans une première étape, il cherche le type de factorisation de $\Phi_\ell(X, j_E)$ (identique par ailleurs à celle de $\Phi_\ell^c(X, j_E)$ ou $\Phi_\ell^*(X, j_E)$) qu'il utilise dans une deuxième étape pour déterminer des candidats pour $c \bmod \ell$.

Nous procédons donc ainsi.

1. Nous recherchons les racines de $\Phi_\ell(X, j_E)$ (ou de $\Phi_\ell^c(X, j_E)$, $\Phi_\ell^*(X, j_E)$) par
$$\mathrm{pgcd}(X^q - X, \Phi_\ell(X, j_E)) \text{ dans } \mathbb{F}_q.$$
2. Selon le nombre de racines, nous déterminons le cas de la proposition 35 auquel ℓ correspond et nous effectuons en conséquence l'une des actions suivantes.
   Cas (i) : nous posons $c = \pm 2\sqrt{q} \bmod \ell$.
   Cas (ii) : si $\Phi_\ell(X, j_E)$ à $\ell+1$ racines, nous posons $c = \pm 2\sqrt{q} \bmod \ell^2$, sinon nous cherchons parmi les entiers r vérifiant $(-1)^{\frac{\ell-1}{r}} = \left(\frac{q}{\ell}\right)$ celui tel que
   $$\mathrm{pgcd}(X^{q^r} - X, \Phi_\ell(X, j_E)) \text{ dans } \mathbb{F}_q$$
   soit non trivial. Les candidats θ pour $c \bmod \ell$ sont alors ceux tels que $\left(\frac{\theta^2-4q}{\ell}\right) = 1$, et tels que le rapport des racines dans $\mathbb{F}_\ell$ de $X^2 - \theta X + q = 0$, soit d'ordre r.

Cas (iii) : nous cherchons parmi les entiers r vérifiant $(-1)^{\frac{\ell+1}{r}} = \left(\frac{q}{\ell}\right)$ celui tel que

$$\text{pgcd}(X^{q^r} - X, \Phi_\ell(X, j_E)) \text{ dans } \mathbb{F}_q$$

soit non trivial. Les candidats θ pour $c \bmod \ell$ sont alors ceux tels que $\left(\frac{\theta^2-4q}{\ell}\right) = -1$, et tels que le rapport des racines dans $\mathbb{F}_{\ell^2}$ de $X^2 - \theta X + q = 0$, soit d'ordre r.

Complexité : la phase 1 nécessite $O(\ell^2 \log q)$ multiplications dans $\mathbb{F}_q$. Une façon classique d'obtenir ensuite X^{q^i} pour i variant de 1 à $r = O(\ell)$ consiste à substituer X^q à X dans $X^{q^{i-1}}$. Pour cela, il est possible de précalculer X^{2q}, ..., $X^{(\ell-1)q}$ au prix de $O(\ell^3)$ multiplications dans $\mathbb{F}_q$ afin d'obtenir $X^{q^{i-1}}(X^q)$ avec $O(\ell^2)$ multiplications dans $\mathbb{F}_q$. Ce qui fait ainsi un total, à ℓ fixé, de $O(\ell^2 \max(\log q, \ell))$ multiplications dans $\mathbb{F}_q$.

3.3.4 Exemple

Nous reprenons l'exemple de la section 3.1, c'est-à-dire la courbe $E_{\overline{7}}$ dans $\mathbb{F}_q$ avec $q = 2^8$ et nous allons appliquer les premières idées d'Atkin à $\ell = 7$. Nous avons,

$$\Phi_7(X, Y) \equiv X^8 + X^4 + XY + 1 \bmod 2.$$

Donc

$$\begin{aligned} \Phi_7(X, 1/\overline{7}) &= X^8 + X^4 + \overline{209}X + 1, \\ &= (X^4 + \overline{51}X^3 + \overline{225}X^2 + \overline{123}X + \overline{141})(X^4 + \overline{51}X^3 + \overline{147}X^2 + \overline{78}X + \overline{2}). \end{aligned}$$

Par conséquent $r = 4$ et nous cherchons les entiers θ tels que $\left(\frac{\theta^2-4q}{\ell}\right) = -1$ et tels que le rapport des racines de $X^2 - \theta X + q = 0$, soit d'ordre 4. Ici, cela donne

$$c \bmod 7 \in \{1, 6\}.$$

3.4 Idées d'Elkies et ses développements

Les idées d'Elkies conduisent à un algorithme qui, pour la moitié des nombres premiers ℓ, regroupe les avantages de l'algorithme de Schoof et d'Atkin sans leurs inconvénients respectifs. Autrement dit, les travaux d'Elkies nous permettent de trouver $c \bmod \ell$ tout en travaillant dans des extensions de degré raisonnable, en l'occurrence $(\ell - 1)/2$ pour la moitié des ℓ au lieu de $(\ell^2 - 1)/2$.

Après en avoir donné le principe, nous expliquons comment obtenir une courbe isogène de degré ℓ à une courbe E à partir des équations modulaires, ce qui nous servira de base aux algorithmes de calcul d'isogénie de la partie II. Enfin, nous expliquons comment étendre l'algorithme obtenu pour déterminer c modulo une puissance d'un nombre premier ℓ et terminons par un exemple complet.

Dans cette section, nous nous limitons à des nombres premiers ℓ impairs mais l'ensemble de ces idées, à quelques modifications près, s'appliquent à $\ell = 2$.

3.4.1 Principe

Nous supposons ici que l'on cherche à déterminer la trace du Frobenius c pour une courbe elliptique E non supersingulière à j-invariant distinct de 0 et 1728 et définie sur un corps fini $\mathbb{F}_q$ par

$$E \ : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6.$$

L'idée consiste à rechercher $c \bmod \ell$ en projetant l'équation caractéristique de ϕ_E, non pas dans $E[\ell]$ mais directement dans l'un des $\ell + 1$ sous-groupes $\mathbb{E}$ d'ordre ℓ de $E[\ell]$. Ce qui donne

$$\phi_{\mathbb{E}}^2 + [q \bmod \ell]_{\mathbb{E}} = [c \bmod \ell]_{\mathbb{E}} \circ \phi_{\mathbb{E}}. \tag{3.3}$$

L'intérêt de travailler directement dans $\mathbb{E}$ plutôt que dans $E[\ell]$ est que les points de ce groupe sont définis dans une extension de degré rs de $\mathbb{F}_q$ où s divise $\ell-1$ et r est d'après les résultats de la section 3.3 le plus petit diviseur de $\ell+1$ ou $\ell-1$ tel que $\phi^r(\mathbb{E}) \subset \mathbb{E}$. Or pour la moitié des nombres premiers ℓ, nous pouvons choisir $\mathbb{E}$ tel que $r=1$. Notons que dans ce dernier cas, il est possible de légèrement simplifier l'équation (3.3) puisqu'alors $X^2 - cX + q$ possède une racine κ dans $\mathbb{F}_\ell$. Ainsi,

$$\phi_{\mathbb{E}} = [\kappa]_{\mathbb{E}},$$

équation à partir de laquelle nous trouvons $c = \kappa + \frac{q}{\kappa} \bmod \ell$.

Le calcul d'isogénies de degrés ℓ définies à partir de E nous permet de déterminer directement un tel $\mathbb{E}$ sans "passer" par $E[\ell]$. En effet, soit E_1 l'une des courbes isogène à E définie sur $\mathbb{F}_{q^r}$ et $\mathcal{I}_1$ l'isogénie de degré ℓ associée. D'après le théorème 14, il existe alors une isogénie duale de E_1 vers E notée $\hat{\mathcal{I}}_1$ telle que $\hat{\mathcal{I}}_1 \circ \mathcal{I}_1 = [\ell]_E$ et donc,

$$\mathrm{Ker}\mathcal{I}_1 \subset \mathrm{Ker}[\ell]_E \text{ avec } \#\mathrm{Ker}\mathcal{I}_1 = \ell.$$

En conséquence, $\mathrm{Ker}\mathcal{I}_1$ est l'un des sous-groupes $\mathbb{E}$ de $E[\ell]$. Pour choisir la "bonne courbe isogène", c'est-à-dire celle avec r minimal, nous utilisons le théorème suivant [103].

Théorème 37. *Soient E, une courbe elliptique définie sur $\mathbb{F}_q$ supposée non supersingulière et à j-invariant distinct de 0 et 1728. Alors le polynôme modulaire $\Phi_\ell(X, j_E)$ a une racine dans $\mathbb{F}_{q^r}$ si et seulement si le noyau $\mathbb{E}$ de l'isogénie $\mathcal{I}_1$ correspondante est tel que $\phi_E^r(\mathbb{E}) = \mathbb{E}$.*

Nous aboutissons ainsi au schéma algorithmique suivant pour déterminer c modulo un nombre premier ℓ.

1. Nous cherchons $f(X)$, un facteur de plus petit degré ρ d'une équation modulaire dans laquelle nous substituons j_E à Y. Nous posons alors
$$\mathbb{K} = \begin{cases} \mathbb{F}_q & \text{si } \rho = 1, \\ \mathbb{F}_q[t]/(f(t)) & \text{sinon [36]}, \end{cases}$$
et prenons pour F l'une des racines dans $\mathbb{K}$ de $f(X)$.
2. Nous déterminons une courbe elliptique E_1 isogène à E de degré ℓ et d'invariant j_{E_1} déterminé à l'aide de F.
3. Nous calculons une isogénie de degré ℓ entre E et E_1 d'abscisse $g_1(X)/h_1^2(X)$.
4. Comme les abscisses des points de $\mathrm{Ker}\mathcal{I}_1$ sont les racines de $h_1(X)$, nous posons
$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6) \text{ avec } \mathbb{X} = \mathbb{K}[X]/(h_1(X)).$$
Le point $P = (X, Y)$ est alors un point de ℓ-torsion.
5. Si $\rho = 1$, nous trouvons $c \mod \ell$ en énumérant les entiers $\kappa \in \mathbb{F}_\ell$ jusqu'à obtenir
$$\phi_E(P) = [\kappa]_E(P).$$
Dans ce cas, $c = \kappa + q/\kappa \bmod \ell$.
Sinon, nous énumérons comme pour l'algorithme original de Schoof les entiers θ de $\mathbb{F}_\ell$ jusqu'à obtenir
$$\phi_E^2(P) + [q \bmod \ell]_E(P) = [\theta]_E(\phi_E(P)),$$
et donc $c = \theta \bmod \ell$.

Remarques : si le point 1 de ce schéma ne pose pas de problèmes théoriques puisqu'il se fait comme pour les premières idées d'Atkin de la section 3.3, il n'en reste pas moins l'un des coûts majeurs de l'algorithme en pratique. Les points 2 et 3 en sont les points cruciaux. L'objet de la section suivante est de régler complètement le point 2. Quant au point 3, il fait l'objet de la partie II. Une fois trouvé $h_1(X)$, le point 4 est immédiat. Enfin, d'un point de vue mathématique, le point 5 est aussi clair si ce n'est qu'il faille, comme pour l'algorithme original de Schoof, faire éventuellement attention à la non irréductibilité de $h_1(X)$ au cours de l'algorithme. Du point de vue informatique, cette étape est aussi très coûteuse et a donné lieu à des améliorations que nous décrivons dans le chapitre 11.

3.4.2 Courbes isogènes et isogénies

Nous cherchons ici à déterminer une courbe E_b isogène à E_a de degré ℓ. Notons qu'à partir de l'invariant j_{E_a} (que nous abrégerons j_a), il est possible de calculer une racine F du polynôme obtenu en substituant j_a à Y dans l'une des équations modulaires de la section 3.4. Ensuite, nous obtenons, suivant les cas, des candidats pour j_{E_b} (que nous abrégerons j_b). Ainsi :

- $j_b = F$ dans le cas du polynôme modulaire,
- toute racine en Y de $\Phi_\ell^c(W_\ell(F), Y)$ avec $W_\ell(F) = \ell^s/F$ convient pour j_b dans le cas d'une équation pour $X_0(\ell)$,
- j_b est l'une des racines en Y de $\Phi_\ell^*(W_\ell(F), Y)$ distinctes de j_a avec, $W_\ell(F) = F$ dans le cas d'une équation pour $X_0^*(\ell)$.

Notons que dans ce dernier cas, la seule méthode connue pour distinguer j_b parmi ces candidats est de brutalement essayer de calculer pour chacune de ces valeurs une isogénie avec les algorithmes de la partie II.

La connaissance de j_b, ne suffit pas pour déterminer complètement les coefficients de E_b et notamment pour distinguer la courbe isogène à E_a de sa tordue. Nous rappelons donc ici les méthodes qui sont connues pour déterminer à partir de j_a et de F les coefficients de la courbe isogène à E_a pour des corps finis de caractéristique petite [70] et grande [39]. Dans ce dernier cas nous calculons aussi p_1, la demi somme des abscisses des points non nuls du noyau de l'isogénie, ce qui nous sera utile pour mettre en œuvre les algorithmes du chapitre 4.

Corps finis de caractéristique deux

Pour des corps finis $\mathbb{F}_{2^n}$, l'équation canonique d'une courbe elliptique à j-invariant non nul est $Y^2 + XY = X^3 + a_2X^2 + a_6$ (cf. tableau 2.1). Comme deux courbes elliptiques ayant une équation de ce type sont isomorphes si elles ont le même coefficient a_6 et des coefficients a_2 de même trace, ou tordues l'une de l'autre sinon, il est d'usage, pour des raisons d'efficacité, de se restreindre à des courbes elliptiques d'équation

$$E_a \ : Y^2 + XY = X^3 + a \text{ avec } j_a = 1/a.$$

Dans ces conditions, une courbe elliptique E_b isogène à E_a se déduit simplement de j_b par

$$E_b \ : Y^2 + XY = X^3 + b \text{ avec } b = 1/j_b.$$

Corps finis de caractéristique trois

Pour des corps finis $\mathbb{F}_{3^n}$, l'équation canonique d'une courbe elliptique E_a à j-invariant non nul est $E_a \ : Y^2 = X^3 + a_2X^2 + a_6$ (cf. tableau 2.1). Dans ce cas, le théorème suivant permet de nous tirer d'affaire [111].

Théorème 38. *L'invariant de Hasse H_E d'une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de caractéristique p satisfait*

$$\mathrm{N}_{\mathbb{F}_q/\mathbb{F}_p}(H_E) = \mathrm{Tr}(\phi_E).$$

Nous pouvons donc choisir $H_{E_b} = H_{E_a} = a_2$, ce qui conduit à une courbe E_b déduite de j_b par

$$E_b \ : Y^2 = X^3 + b_2X^2 + b_6 \text{ avec } b_2 = a_2 \text{ et } b_6 = -a_2^3/j_b.$$

Corps finis de caractéristique supérieure à cinq

Pour des corps finis de grande taille, il devient irréalisable de calculer l'invariant de Hasse d'une courbe elliptique et donc d'utiliser le théorème 38 pour distinguer la courbe isogène à E_a de sa tordue. Heureusement, les idées d'Elkies permettent de nous tirer d'affaire en exploitant les renseignements fournis par les équations modulaires. En reprenant les notations utilisées dans la section 3.3 pour des courbes considérées sur $\mathbb{C}$, cela revient donc, si l'on suppose que E_a est isomorphe au quotient de $\mathbb{C}$ par le réseau $\mathbb{Z} + \tau\mathbb{Z}$, à trouver les coefficients de la courbe elliptique E_b isomorphe au quotient de $\mathbb{C}$ par $\mathbb{Z} + \ell\tau\mathbb{Z}$.

Dans ce cadre, nous avons l'équation de E_a est classiquement donnée en fonction des séries d'Eisenstein normalisées par

$$E_a \ : Y^2 = X^3 - 3\lambda^4 E_4(\tau)X - 2\lambda^6 E_6(\tau),$$

où $\lambda = \pi/\sqrt{3}$,

$$E_4(\tau) = 1 + 240\sum_{n=1}^{\infty} \frac{n^3 x^n}{1-x^n} \text{ et } E_6(\tau) = 1 - 504\sum_{n=1}^{\infty} \frac{n^5 x^n}{1-x^n} \text{ et } x = e^{2i\pi\tau}.$$

Ceci, au passage, nous permet d'avoir $j(\tau)$ et $\Delta(\tau)$ par

$$j(\tau) = 1728\frac{E_4^3(\tau)}{E_4^3(\tau) - E_6^2(\tau)} \text{ et } \Delta(\tau) = 1728^{-1}\lambda^{12}(E_4^3(\tau) - E_6^2(\tau)). \tag{3.4}$$

L'équation de E_b est alors

$$E_b \ : Y^2 = X^3 - 3\lambda^4 E_4(\ell\tau)X - 2\lambda^6 E_6(\ell\tau),$$

D'autre part, il nous faut aussi déterminer $p_1(\tau)$, la demi somme des abscisses du noyau de l'isogénie pour les algorithmes du chapitre 4, c'est-à-dire

$$p_1(\tau) = \sum_{n=1}^{(\ell-1)/2} \wp_a\left(\frac{n}{\ell}\right). \tag{3.5}$$

Posons

$$E_2(\tau) = 1 - 24\sum_{n=1}^{\infty} \frac{nx^n}{1-x^n},$$

Elkies montre alors, en substituant n/ℓ à z dans le développement

$$\wp(z) = \left(\frac{2\pi}{\omega_1}\right)^2 \left[-\frac{1}{12} + \sum_{n=1}^{\infty} \frac{x^n}{(1-x^n)^2} - \sum_{n=-\infty}^{\infty} \frac{x^n y}{(1-x^n y)^2}\right] \tag{3.6}$$

où $x = e^{2i\pi\tau}$ et $y = e^{2i\pi z}$, puis en identifiant les séries obtenues, que

$$p_1(\tau) = \frac{\ell}{2}(\ell E_2(\ell\tau) - E_2(\tau)).$$

Le problème est donc de déterminer $E_2(\ell\tau)$, $E_4(\ell\tau)$ et $E_6(\ell\tau)$ en fonction de $E_2(\tau)$, $E_4(\tau)$ et $E_6(\tau)$. Pour cela, la méthode proposée par Elkies consiste pour l'essentiel à utiliser le lien entre $j(\tau)$ et $j(\ell\tau)$ fournie par l'équation modulaire utilisée. Comme en pratique nous n'utilisons pas le polynôme modulaire $\Phi_\ell(X,Y)$, nous renvoyons à [103] pour ce cas. Par contre nous détaillons les calculs à réaliser pour $\phi_\ell^c(X,Y)$ et $\phi_\ell^*(X,Y)$. Nous aurons besoin pour cela des dérivées logarithmiques de $E_2(\tau)$, $E_4(\tau)$ et $E_6(\tau)$ par rapport à τ obtenues aussi en identifiant les séries correspondantes,

$$12\frac{E_2'(\tau)}{E_2(\tau)} = E_2(\tau) - \frac{E_4(\tau)}{E_2(\tau)},\ 3\frac{E_4'(\tau)}{E_4(\tau)} = E_2(\tau) - \frac{E_6(\tau)}{E_4(\tau)} \text{ et } 2\frac{E_6'(\tau)}{E_6(\tau)} = E_2(\tau) - \frac{E_4^2(\tau)}{E_6(\tau)}. \tag{3.7}$$

Ceci nous permet de trouver les dérivées logarithmiques de $\Delta(\tau)$ et $j(\tau)$,

$$\frac{\Delta'(\tau)}{\Delta(\tau)} = E_2(\tau) \text{ et } \frac{j'(\tau)}{j(\tau)} = -\frac{E_6(\tau)}{E_4(\tau)}. \tag{3.8}$$

Cas de $X_0(\ell)$: nous avons ici $\Phi_\ell^c(f(\tau), j(\tau)) = 0$ où

$$f(\tau) = \ell^s \left(\frac{\eta(\ell\tau)}{\eta(\tau)}\right)^{2s} \text{ avec } s = \frac{12}{\text{pgcd}(12, \ell-1)} \text{ et } v = s(\ell-1)/12.$$

D'autre part, on peut montrer que $E_2(\tau) = 24\dfrac{\eta'(\tau)}{\eta(\tau)}$. Donc $\dfrac{f'(\tau)}{f(\tau)} = \dfrac{s}{12}(\ell E_2(\ell\tau) - E_2(\tau))$, et ainsi

$$p_1(\tau) = \frac{6\ell f'(\tau)}{s f(\tau)}.$$

Pour déterminer la dérivée logarithmique de $f(\tau)$, nous différencions $\Phi_\ell^c(f(\tau), j(\tau)) = 0$, ce qui donne

$$f'(\tau)\frac{\partial\Phi_\ell^c}{\partial X}(f(\tau), j(\tau)) + j'(\tau)\frac{\partial\Phi_\ell^c}{\partial Y}(f(\tau), j(\tau)) = 0.$$

En posant $D_X(\tau) = f(\tau)\frac{\partial\Phi_\ell^c}{\partial X}(f(\tau), j(\tau))$ et $D_Y(\tau) = j(\tau)\frac{\partial\Phi_\ell^c}{\partial Y}(f(\tau), j(\tau))$ nous avons donc

$$\frac{f'(\tau)}{f(\tau)} = \frac{E_6(\tau)D_Y(\tau)}{E_4(\tau)D_X(\tau)} \tag{3.9}$$

et

$$\ell E_2(\ell\tau) = E_2(\tau) + \frac{12}{s}\frac{E_6(\tau)D_Y(\tau)}{E_4(\tau)D_X(\tau)}. \tag{3.10}$$

Ainsi,

$$\boxed{p_1(\tau) = \frac{6\ell}{s}\frac{E_6(\tau)D_Y(\tau)}{E_4(\tau)D_X(\tau)}.}$$

Nous déterminons ensuite $E_4(\ell\tau)$ en dérivant l'équation (3.10) puis en simplifiant les dérivées de $f(\tau)$, $j(\tau)$, $E_4(\tau)$ et $E_6(\tau)$ à l'aide des équations (3.7), (3.8) et (3.9). Les termes en $E_2(\tau)$ s'annulent alors et nous trouvons finalement

$$\boxed{\begin{aligned}\ell^2 E_4(\ell\tau) = E_4(\tau) \quad &+\frac{144\,{D_Y}^2(\tau){E_6}^2(\tau)}{s^2{D_X}^2(\tau){E_4}^2(\tau)} - \frac{48\,D_Y(\tau){E_6}^2(\tau)}{sD_X(\tau){E_4}^2(\tau)} - \frac{288\,D_Y(\tau)D_{XY}(\tau){E_6}^2(\tau)}{s{D_X}^2(\tau){E_4}^2(\tau)} \\ &+\frac{144\,D_{YY}(\tau){E_6}^2(\tau)}{sD_X(\tau){E_4}^2(\tau)} + \frac{72\,D_Y(\tau)E_4(\tau)}{sD_X(\tau)} + \frac{144\,{D_Y}^2(\tau)D_{XX}(\tau){E_6}^2(\tau)}{s{D_X}^3(\tau){E_4}^2(\tau)}\end{aligned}}$$

en posant $D_{XX}(\tau) = D_X(\tau) + f^2(\tau)\dfrac{\partial^2\Phi_\ell^c}{\partial X^2}(f(\tau), j(\tau))$, $D_{XY}(\tau) = f(\tau)j(\tau)\dfrac{\partial^2\Phi_\ell^c}{\partial XY}(f(\tau), j(\tau))$ et $D_{YY}(\tau) = D_Y(\tau) + j^2(\tau)\dfrac{\partial^2\Phi_\ell^c}{\partial Y^2}(f(\tau), j(\tau))$. À ce stade, nous pouvons déterminer $j(\ell\tau)$ en remarquant que $\Delta(\tau) = (2i\pi)^{12}\eta^{24}(\tau)$. Ainsi

$$\Delta(\ell\tau) = f^{12/s}(\tau)\Delta(\tau)/\ell^{12}$$

et donc

$$\boxed{j(\ell\tau) = \frac{E_4^3(\ell\tau)}{\Delta(\ell\tau)} = 1728\frac{E_4^3(\ell\tau)\ell^{12}}{f^{12/s(\tau)}\left(E_4^3(\tau) - E_6^2(\tau)\right)}.}$$

Enfin, pour déterminer $E_6(\ell\tau)$, nous différencions l'équation $\Phi_\ell^c(\ell^s/f(\tau), j(\ell\tau)) = 0$, ce qui donne

$$-\ell^s\frac{f'(\tau)}{f^2(\tau)}\frac{\partial\Phi_\ell^c}{\partial X}\left(\frac{\ell^s}{f(\tau)}, j(\ell\tau)\right) + \ell j'(\ell\tau)\frac{\partial\Phi_\ell^c}{\partial Y}\left(\frac{\ell^s}{f(\tau)}, j(\ell\tau)\right) = 0.$$

D'où

$$\boxed{E_6(\ell\tau) = -E_4(\ell\tau)\frac{j'(\ell\tau)}{j(\ell\tau)} = -E_4(\ell\tau)\frac{\dfrac{\ell^s}{f(\tau)}\dfrac{\partial\Phi_\ell^c}{\partial X}\left(\dfrac{\ell^s}{f(\tau)}, j(\ell\tau)\right)D_Y(\tau)E_6(\tau)}{\ell j(\ell\tau)\dfrac{\partial\Phi_\ell^c}{\partial Y}\left(\dfrac{\ell^s}{f(\tau)}, j(\ell\tau)\right)D_X(\tau)E_4(\tau)}.}$$

Cas de $X_0^*(\ell)$: nous sommes ici dans l'obligation de factoriser l'équation modulaire pour trouver $j(\ell\tau)$ à partir de $f^*(\tau)$. Alors $E_4(\ell\tau)$ et $E_6(\ell\tau)$ sont déterminés à partir de $j(\ell\tau)$ et $j'(\ell\tau)$ par

$$E_4(\ell\tau) = \frac{{j'}^2(\ell\tau)}{j(\ell\tau)\,(j(\ell\tau)-1728)} \text{ et } E_6(\ell\tau) = -\frac{{j'}^3(\ell\tau)}{j^2(\ell\tau)\,(j(\ell\tau)-1728)}.$$

Pour trouver $j'(\ell\tau)$, nous calculons le couple $(f^{*\prime}(\tau), j'(\ell\tau))$ solution du système linéaire obtenu par différenciation de $\Phi_\ell^*(f^*(\tau), j(\tau)) = 0$ et $\Phi_\ell^*(f^*(\tau), j(\ell\tau)) = 0$, ce qui conduit à

$$\frac{f^{*\prime}(\tau)}{f^*(\tau)} = \frac{E_6(\tau)D_Y(\tau)}{E_4(\tau)D_X(\tau)} \text{ et } \frac{j'(\ell\tau)}{j(\ell\tau)} = -\frac{E_6(\tau)D_Y(\tau)D_Y^*(\tau)}{E_4(\tau)D_X(\tau)D_X^*(\tau)}, \tag{3.11}$$

en posant

$$D_X(\tau) = f^*(\tau)\frac{\partial\Phi_\ell^*}{\partial X}(f^*(\tau), j(\tau)), D_Y(\tau) = j(\tau)\frac{\partial\Phi_\ell^*}{\partial Y}(f^*(\tau), j(\tau))$$

et

$$D_X^*(\tau) = f^*(\tau)\frac{\partial\Phi_\ell^*}{\partial X}(f^*(\tau), j(\ell\tau)), D_Y^*(\tau) = \ell j(\ell\tau)\frac{\partial\Phi_\ell^*}{\partial Y}(f^*(\tau), j(\ell\tau)).$$

Nous obtenons finalement

$$\boxed{E_4(\ell\tau) = \frac{D_X^{*2}(\tau)D_Y^2(\tau)E_6^2(\tau)j(\ell\tau)}{D_Y^{*2}(\tau)D_X^2(\tau)E_4^2(\tau)\,(j(\ell\tau)-1728)},}$$

et

$$\boxed{E_6(\ell\tau) = \frac{D_X^{*3}(\tau)D_Y^3(\tau)E_6^3(\tau)j(\ell\tau)}{D_Y^{*3}(\tau)D_X^3(\tau)E_4^3(\tau)\,(j(\ell\tau)-1728)}.}$$

Pour déterminer $p_1(\tau)$, nous différencions deux fois l'équation $\Phi_\ell^*(f^*(\tau), j(\tau)) = 0$,

$$f^{*\prime\prime}(\tau)D_X(\tau) + {f^{*\prime}}^2(\tau)\frac{\partial^2\Phi_\ell^*}{\partial X^2}(f^*(\tau), j(\tau)) + 2f^{*\prime}(\tau)j'(\tau)\frac{\partial^2\Phi_\ell^*}{\partial XY}(f^*(\tau), j(\tau))$$
$$+ j''(\tau)D_Y(\tau) + {j'(\tau)}^2\frac{\partial^2\Phi_\ell^*}{\partial Y^2}(f^*(\tau), j(\tau)) = 0.$$

En différenciant l'équation (3.8), nous avons par ailleurs

$$j''(\tau) = j(\tau)\left(\frac{2E_6^2(\tau)}{3E_4^2(\tau)} - \frac{E_2(\tau)E_6(\tau)}{2E_4(\tau)} + \frac{E_4(\tau)}{2}\right).$$

Nous injectons cette valeur dans la relation précédente et nous remplaçons $j'(\tau)$ par sa valeur, nous trouvons alors

$$\frac{f^{*\prime\prime}(\tau)}{f^{*\prime}(\tau)} = \frac{1}{D_X(\tau)}\left(-f^{*\prime}(\tau)\frac{\partial^2\Phi_\ell^*}{\partial X^2}(f^*(\tau), j(\tau)) + 2j(\tau)\frac{\partial^2\Phi_\ell^*}{\partial XY}(f^*(\tau), j(\tau))\frac{E_6(\tau)}{E_4(\tau)}\right.$$
$$\left. - \frac{E_6^2(\tau)}{f^{*\prime}(\tau)E_4^2(\tau)}\left(j(\tau)D_Y(\tau) + j^2(\tau)\frac{\partial^2\Phi_\ell^*}{\partial Y^2}(f^*(\tau), j(\tau))\right)\right)$$
$$+ \frac{j(\tau)D_Y(\tau)}{f^{*\prime}(\tau)D_X(\tau)}\left(\frac{E_2(\tau)E_6(\tau)}{6E_4(\tau)} + \frac{E_6^2(\tau)}{3E_4^2(\tau)} - \frac{E_4(\tau)}{2}\right).$$

Nous utilisons alors l'expression de $f^{*\prime}(\tau)$ obtenue avec l'équation (3.11) pour simplifier cette relation,

$$\frac{f^{*\prime\prime}(\tau)}{f^{*\prime}(\tau)} = \frac{1}{D_X(\tau)}\left(-f^{*\prime}(\tau)\frac{\partial^2\Phi_\ell^*}{\partial X^2}(f^*(\tau), j(\tau)) + 2j(\tau)\frac{\partial^2\Phi_\ell^*}{\partial XY}(f^*(\tau), j(\tau))\frac{E_6(\tau)}{E_4(\tau)}\right.$$
$$\left. - \frac{E_6^2(\tau)}{f^{*\prime}(\tau)E_4^2(\tau)}\left(j(\tau)D_Y(\tau) + j^2(\tau)\frac{\partial^2\Phi_\ell^*}{\partial Y^2}(f^*(\tau), j(\tau))\right)\right)$$
$$+ \frac{E_2(\tau)}{6} + \frac{E_6(\tau)}{3E_4(\tau)} - \frac{E_4^2(\tau)}{2E_6(\tau)}.$$

Avec le même procédé, mais cette fois ci appliqué à $\Phi_\ell^c(f^*(\tau), j(\ell\tau)) = 0$, nous obtenons

$$\frac{f^{*\prime\prime}(\tau)}{f^{*\prime}(\tau)} = \frac{1}{D_X^*(\tau)}\left(-f^{*\prime}(\tau)\frac{\partial^2\Phi_\ell^*}{\partial X^2}(f^*(\tau), j(\ell\tau)) + 2\ell j(\ell\tau)\frac{\partial^2\Phi_\ell^*}{\partial XY}(f^*(\tau), j(\ell\tau))\frac{E_6(\ell\tau)}{E_4(\ell\tau)}\right.$$
$$\left. -\ell^2\frac{E_6(\ell\tau)^2}{f^{*\prime}(\tau)E_4(\ell\tau)^2}\left(j(\ell\tau)D_Y^*(\tau) + j(\ell\tau)^2\frac{\partial^2\Phi_\ell^*}{\partial Y^2}(f^*(\tau), j(\ell\tau))\right)\right)$$
$$+\ell\left(\frac{E_2(\ell\tau)}{6} + \frac{E_6(\ell\tau)}{3E_4(\ell\tau)} - \frac{E_4(\ell\tau)^2}{2E_6(\ell\tau)}\right).$$

Il suffit de soustraire ces deux dernières équations pour trouver $\ell E_2(\ell\tau) - E_2(\tau)$, puis $p_1(\tau)$.

3.4.3 Cycle d'isogénies

Pour déterminer la trace c du Frobenius ϕ_E modulo une puissance de nombre premier ℓ^k, nous utilisons la même idée que pour déterminer $c \bmod \ell$, c'est-à-dire projeter l'équation caractéristique du Frobenius sur $\mathbb{E}$, l'un des ℓ^k sous-groupes d'ordre ℓ^k de $E[\ell^k]$. La difficulté étant le calcul explicite de $\mathbb{E}$, il est naturel, suite aux idées précédentes, de le rechercher comme le noyau d'une isogénie $\mathcal{I}$ de degré ℓ^k.

Couveignes, Dewaghe et Morain [34, 33] proposent deux variantes d'une même idée pour calculer $\mathcal{I}$. Ainsi, s'ils réitèrent k fois le calcul d'une isogénie de degré ℓ dont la composition mène à une isogénie de degré ℓ^k, c'est l'ensemble de définition de $\mathcal{I}$ qu'ils font varier. Dans une première variante, $\mathcal{I}$ a pour ensemble de définition E et est obtenue par récurrence comme la composition d'une isogénie de degré ℓ dont l'ensemble de définition est la courbe d'arrivée d'une isogénie de degré ℓ^{k-1}. L'inconvénient de cette méthode est qu'il n'est possible de tirer parti de la présence d'un facteur de petit degré du dénominateur $\mathcal{I}$ qu'en factorisant le numérateur de toutes les isogénies de degré ℓ. La deuxième variante permet de remplacer ces k factorisations par la seule factorisation d'une isogénie de degré ℓ définie à partir de E en composant une isogénie de degré ℓ dont l'ensemble d'arrivée est cette fois l'image d'une isogénie de degré ℓ^{k-1}.

Composition "en avant"

Avant de décrire plus en détail l'algorithme, regardons d'un peu plus près comment se comporte la composition "à droite" d'une isogénie de degré ℓ^{k-1} avec celle d'une isogénie de degré ℓ. Soient donc $\mathcal{I}$ et $\mathcal{I}_k$, deux isogénies de degrés respectifs ℓ^{k-1} et ℓ définies par

$$\mathcal{I} : \begin{array}{c} E \to E_{k-1}, \\ (X,Y) \mapsto \left(\dfrac{g(X)}{h^2(X)}, \dfrac{l(X)+Yk(X)}{h^3(X)}\right), \end{array} \quad \mathcal{I}_k \begin{array}{c} E_{k-1} \to E_k, \\ (X,Y) \mapsto \left(\dfrac{g_k(X)}{h_k^2(X)}, \dfrac{l_k(X)+Yk_k(X)}{h_k^3(X)}\right). \end{array}$$

où les dénominateurs $h(X)$ et $h_k(X)$ sont des polynômes de degrés respectifs $\ell^{k-2}(\ell-1)/2$ et $(\ell-1)/2$. Alors il n'est pas difficile de voir que l'isogénie $\mathcal{I}_k \circ \mathcal{I}$ est de degré ℓ^k et que son dénominateur est égal au numérateur de $h(X)h_k\left(g(X)/h^2(X)\right)$. Cette formule appelle à deux remarques :

- parmi les ℓ^k points d'un sous-groupe $\mathbb{E}$ d'ordre ℓ^k de $E[\ell^k]$, se trouvent les ℓ^{k-1} points de $E[\ell^{k-1}] \cap \mathbb{E}$. C'est pourquoi le dénominateur de $\mathcal{I}_k \circ \mathcal{I}$ est le produit de $h(X)$ dont les racines sont les abscisses des points de $E[\ell^{k-1}] \cap \mathbb{E}$ par celui d'un polynôme dont les racines sont les abscisses de points d'ordre ℓ^k. Ce qui signifie qu'il est possible de trouver un point de ℓ^k torsion dans une extension de degré au plus $\ell^{k-1}(\ell-1)$ sur $\mathbb{F}_q$ et définie par le numérateur de $h_k\left(g(X)/h(X)\right)$.
- si $h_k(X)$ possède un facteur $\overline{h}_k(X)$ de degré d, il est préférable en pratique de travailler avec l'extension définie à partir du numérateur de $\overline{h}_k\left(g(X)/h(X)\right)$, ce qui permet de manipuler un point de ℓ^k torsion dans une extension de degré $2d\ell^{k-1}$. Remarquons, que lorsque ℓ correspond au cas (ii) de la proposition 35, il est important de choisir pour $\mathcal{I}_k$ une isogénie qui corresponde à la racine κ de $X^2 - cX + q$ de plus petit ordre dans $\mathbb{F}_\ell$ puisque le semi ordre de κ défini comme suit est égal au degré du facteur $\overline{h}_k(X)$ correspondant.

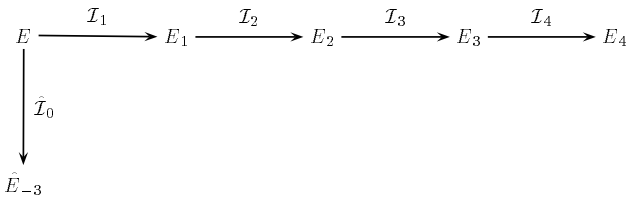


FIG. 3.2 – Cycle d'isogénies.

Définition 18. *Soit δ l'ordre d'un entier κ dans $\mathbb{Z}/\ell\mathbb{Z}$, alors le semi ordre de κ est égal à δ si δ est impair ou à $\delta/2$ sinon.*

Globalement, ce sont donc k isogénies de degré ℓ qui sont calculées, comme nous l'avons par exemple représenté sur la figure 3.2 pour une isogénie $\mathcal{I}_4 \circ \mathcal{I}_3 \circ \mathcal{I}_2 \circ \mathcal{I}_1$ de degré ℓ^4 entre E et E_4. Nous obtenons ainsi une généralisation de l'algorithme `ComputeTModLn` [33] pour calculer $c \bmod \ell^k$ sur une courbe elliptique E éventuellement plongée dans une extension $\mathbb{F}_{q^r}$ de $\mathbb{F}_q$.

1. Nous cherchons $f_1(X) \cdots f_s(X)$ la factorisation de $\Phi_\ell^{c,*}(X, j_E)$ en polynômes irréductibles de degrés croissants. Nous posons alors, en faisant référence à la proposition 35,
$$\mathbb{K} = \begin{cases} \mathbb{F}_q[t]/(f_2(t)) & \text{dans le cas } (i), \\ \mathbb{F}_q & \text{dans le cas } (ii), \\ \mathbb{F}_q[t]/(f_1(t)) & \text{dans le cas } (iii). \end{cases}$$
Nous appelons ρ le degré de l'extension $\mathbb{K}$ par rapport à $\mathbb{F}_q$ et nous prenons pour F l'une des racines dans $\mathbb{K}$ de $\Phi_\ell^{c,*}(X, j_E)$.
2. Nous calculons à partir de F une courbe E_1 isogène à E de degré ℓ ainsi que l'abscisse $g_1(X)/h_1^2(X)$ de l'isogénie associée.
3. Nous posons $\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6)$ avec $\mathbb{X} = \mathbb{K}[X]/(h_1(X))$. Le point $P = (X, Y)$ est alors un point de ℓ-torsion.
4. Si $\rho = 1$, alors
   (a) nous trouvons le plus petit entier κ tel que
   $$\phi_E(P) = [\kappa]_E(P)$$
   et nous posons $\theta = \kappa + q/\kappa \bmod \ell$. Alors, nous avons $c = \theta \bmod \ell$ et nous posons δ le plus petit des semi ordres de κ et $q/\kappa \bmod \ell$;
   (b) si le semi ordre de κ est strictement plus grand que δ, nous choisissons pour F l'autre racine de $\Phi_\ell^{c,*}(X, j_E)$ dans $\mathbb{F}_q$ et nous recalculons E_1 et $\mathcal{I}_1$ en conséquence.

   Sinon,
   (a) nous énumérons comme pour l'algorithme original de Schoof les entiers θ de $\mathbb{F}_\ell$ jusqu'à obtenir $\phi_E^2(P) + [q \bmod \ell]_E(P) = [\theta]_E(\phi_E(P))$. Alors $c = \theta \bmod \ell$ et nous posons $\delta = (\ell - 1)/2$.
5. Posons $E_0 = E$. Si $\rho = 1$, alors pour i variant de 2 à k,
   (a) nous calculons une courbe E_i isogène à E_{i-1} non isomorphe à E_{i-2}, et l'abscisse $g_i(X)/h_i^2(X)$ de l'isogénie associée.
   (b) nous calculons un facteur $\overline{h}_i(X)$ de degré δ de $h_i(X)$ et prenons pour $h(X)$ le numérateur de $\overline{h}_i\left(\frac{G(X)}{H(X)}\right)$ où $G(X)/H^2(X)$ est l'abscisse de l'isogénie $\mathcal{I}_{i-1} \circ \cdots \circ \mathcal{I}_1$.
   (c) nous posons $\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6)$ avec $\mathbb{X} = \mathbb{K}[X]/(h(X))$. Le point $P = (X, Y)$ est alors un point de ℓ^i-torsion.
   (d) nous trouvons le plus petit entier μ (borné par ℓ) tel que
   $$\phi_E(P) = [\kappa + \mu\ell^{i-1}]_E(P)$$
   et nous posons $\kappa = \kappa + \mu\ell^{i-1}$ et $\theta = \kappa + q/\kappa \bmod \ell$. Alors $c = \theta \bmod \ell$.

Composition "à reculons"

La méthode précédente a pour inconvénient la factorisation à l'étape 5b. de l'algorithme d'un polynôme $h_i(X)$ de degré $(\ell-1)/2$. L'amélioration proposée dans [33] supprime ce désagrément en calculant non pas des isogénies de degré ℓ^i partant de la même courbe mais plutôt arrivant sur la même courbe. Sur l'exemple de la figure 3.3, une isogénie de degré ℓ^4 est ainsi égale à $\mathcal{I}_0 \circ \mathcal{I}_{-1} \circ \mathcal{I}_{-2} \circ \mathcal{I}_{-3}$.

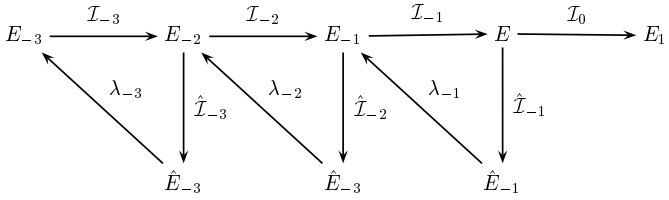
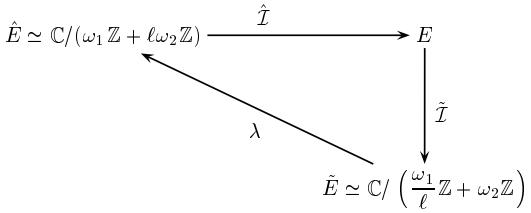


FIG. 3.3 – Cycle d'isogénies obtenu "à reculons".

Contrairement aux corps finis de caractéristique 2 et 3, une petite difficulté en caractéristique plus grande que 5 provient du fait que si nous calculons avec la méthode d'Elkies une courbe E isogène de degré ℓ à une courbe $\hat{E}$ et qu'ensuite, avec ce même algorithme, nous calculons une courbe isogène à E de même invariant que $\hat{E}$, nous obtenons, certes une courbe $\tilde{E}$ isomorphe à E^*, mais distincte.



Cependant, l'isomorphisme λ entre les deux courbes n'est autre que

$$\begin{array}{rccc} \lambda\ : & \tilde{E}(\mathbb{F}_q) & \to & \hat{E}(\mathbb{F}_q), \\ & (X,Y) & \mapsto & \left(\dfrac{X}{\ell^2}, \dfrac{Y}{\ell^3}\right), \end{array}$$

comme le fait justement remarquer Dewaghe. Ainsi, si nous supposons avoir, dans un premier temps, calculé avec l'algorithme d'Elkies à partir d'une racine F de $\Phi_\ell^{c,*}(X, j_E)$ une courbe $\tilde{E}$ isogène de degré ℓ à E donnée par

$$\tilde{E}\ : Y^2 = X^3 + \tilde{a}_4 X + \tilde{a}_6$$

ainsi que p_1, alors $\hat{E}$ est donnée par

$$\hat{E}\ : Y^2 = X^3 + \hat{a}_4 X + \hat{a}_6,$$

et E serait obtenue avec l'algorithme d'Elkies avec la racine $W_\ell(F)$ de $\Phi_\ell^{c,*}(X, j_{\hat{E}})$. Nous avons de plus $\hat{p}_1 = -\ell p_1$.

Ainsi armés, nous obtenons la généralisation suivante de l'algorithme `ComputeTModLnBackwards` [33].

1. Nous cherchons $f_1(X)\cdots f_s(X)$ la factorisation de $\Phi_\ell^{c,*}(X, j_E)$ en polynômes irréductibles de degrés croissants. Nous posons alors, en faisant référence à la proposition 35,

$$\mathbb{K} = \begin{cases} \mathbb{F}_q[t]/(f_2(t)) & \text{dans le cas } (i), \\ \mathbb{F}_q & \text{dans le cas } (ii), \\ \mathbb{F}_q[t]/(f_1(t)) & \text{dans le cas } (iii). \end{cases}$$

   Nous appelons ρ, le degré de l'extension $\mathbb{K}$ par rapport à $\mathbb{F}_q$ et nous prenons pour F, l'une des racines dans $\mathbb{K}$ de $\Phi_\ell^{c,*}(X, j_E)$.
2. Nous calculons à partir de F une courbe E_1 isogène à E de degré ℓ et $g_0(X)/h_0^2(X)$, l'abscisse de l'isogénie $\mathcal{I}_0$ associée.

3. Nous posons $\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6)$ avec $\mathbb{X} = \mathbb{K}[X]/(h_0(X))$. Le point $P = (X, Y)$ est alors un point de ℓ-torsion.
4. Si $\rho = 1$, alors
   (a) nous trouvons le plus petit entier κ tel que $\phi_E(P) = [\kappa]_E(P)$. et nous posons $\theta = \kappa + q/\kappa \bmod \ell$. Alors nous avons $c = \theta \bmod \ell$ et nous posons δ le plus petit des semi ordres de κ et $q/\kappa \bmod \ell$;
   (b) si le semi ordre de κ est strictement plus grand que δ, nous choisissons pour F l'autre racine $\hat{F}$ de $\Phi_\ell^{c,*}(X, j_E)$ dans $\mathbb{F}_q$ et nous recalculons E_1 et $\mathcal{I}_0$ en conséquence.
   (c) nous calculons un facteur $h(X)$ de degré δ de $h_0(X)$.

   Sinon,
   (a) nous énumérons comme pour l'algorithme original de Schoof les entiers θ de $\mathbb{F}_\ell$ jusqu'à obtenir $\phi_E^2(P) + [q \bmod \ell]_E(P) = [\theta]_E(\phi_E(P))$, et alors $c = \theta \bmod \ell$;
   (b) nous posons $h(X) = h_0(X)$.
5. Posons $E_0 = E$. Si $\rho = 1$, alors pour i variant de -1 à $1 - k$,
   (a) nous calculons la courbe $\hat{E}_i$ isogène de degré ℓ à E_{i+1} à partir d'une racine $\hat{F}$ dans $\mathbb{K}$ de $\Phi_\ell^{c,*}(X, j_{E_{i+1}})$ distincte de F ;
   (b) nous calculons une courbe elliptique E_i isogène à E_{i+1} à partir de $\hat{E}_i$ via l'isomorphisme λ_i en posant $F = W_\ell(\hat{F})$ ainsi que l'abscisse $g_i(X)/h_i^2(X)$ de l'isogénie $\mathcal{I}_i$ associée ;
   (c) nous remplaçons $h(X)$ par le numérateur de $h(g_i(X)/h_i^2(X))$ et posons

$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6) \text{ avec } \mathbb{X} = \mathbb{K}[X]/(h(X)).$$

   Le point $P = (X, Y)$ est alors un point de ℓ^{-i+1}-torsion dans $E_i(\mathbb{Y})$;
   (d) nous trouvons le plus petit entier μ (borné par ℓ) tel que

$$\phi_{E_i}(P) = [\kappa + \mu\ell^{-i}]_E(P)$$

   et nous posons $\kappa = \kappa + \mu\ell^{-i}$, $\theta = \kappa + q/\kappa \bmod \ell^{-i+1}$. Alors $c = \theta \bmod \ell^{-i+1}$.

Complexité

La complexité asymptotique de ces deux algorithmes étant identique concentrons-nous sur le second. Pour calculer $c \bmod \ell$, la première chose à faire est de trouver un facteur de degré r de $\Phi_\ell^{c,*}(X, j_E)$, ce qui peut être réalisé en $O(\ell^2 \max(\log q, \ell))$ multiplications dans $\mathbb{F}_q$. Le coût du calcul de la courbe isogène étant clairement négligeable, reste ensuite celui du calcul d'une isogénie de degré ℓ sur une extension de degré r de $\mathbb{F}_q$ dont le coût est égal à $O(r^2\ell^2)$ (chapitre 4), $O(r^2\ell^3)$ (chapitre 5 ou 7) ou $O(r^2\ell^2 \log q)$ (chapitre 6). Le calcul de θ nécessite alors $O(\ell^2 r^2 \log q)$ multiplications dans $\mathbb{F}_q$ pour calculer $\phi_E(P)$ (éventuellement $\phi_E^2(P)$) puis $O(\ell^3 r^2)$ pour effectuer les $O(\ell)$ sommes de points nécessaires dans $E(\mathbb{Y})$. Ce qui nous fait un maximum de $O(\ell^2 r^2 \max(\log q, \ell))$ multiplications dans $\mathbb{F}_q$. Pour $c \bmod \ell^k$, un raisonnement analogue conduit à un maximum de $O(\ell^{2k} r^2 \max(\log q, \ell^k))$ multiplications dans $\mathbb{F}_q$.

Comme, d'une part, nous avons $r = 1$ pour la moitié des nombres premiers ℓ et que, d'autre part, $O(\log q)$ nombres premiers ℓ de l'ordre de $O(\log q)$ sont nécessaires dans l'algorithme `SEA`, nous avons le résultat suivant.

Proposition 13. *Le calcul de la cardinalité d'une courbe elliptique définie sur $\mathbb{F}_q$ avec l'algorithme `SEA` nécessite $O(\log^6 q)$ opérations élémentaires.*

3.4.4 Exemple

Considérons encore la courbe $E_{\overline{7}}$ dans $\mathbb{F}_{2^8}$, nous allons calculer la valeur de c modulo $\ell^k = 5^3$. Pour cela, nous substituons tout d'abord $j_E = \overline{209}$ à Y dans

$$\Phi_5^c(X, Y) = X^6 + 30\,X^5 + 315\,X^4 + 1300\,X^3 + 1575\,X^2 - (-Y + 750)\,X + 125.$$

Le polynôme obtenu a alors deux racines : $F = \overline{52}$, $\hat{F} = \overline{130}$; et nous notons $\mathbb{K} = \mathbb{F}_{2^8}$. Substituons X par $1/F$ dans $\Phi_5^c(X, Y)$, nous obtenons un polynôme de degré 1 dont la racine est $j_{E_1} = \overline{232} = 1/\overline{8}$. E est donc isogène de degré 5 à

$$E_1 \ : Y^2 + XY = X^3 + \overline{8}.$$

Une simple application du corollaire 6 (cf. chapitre 7) permet alors de trouver que l'abscisse de l'isogénie $\mathcal{I}_0$ associée est égale à

$$\frac{g_0(X)}{h_0^2(X)} = \frac{X^5 + \overline{15}\,X^3 + \overline{140}\,X}{(X^2 + \overline{57}\,X + \overline{74})^2}.$$

Posons alors

$$\mathbb{X} = \mathbb{F}_{2^8}[X]/(X^2 + \overline{57}\,X + \overline{74}), \text{ puis } \mathbb{Y} = \mathbb{X}[Y]/(Y^2 + XY + X^3 + \overline{7}) \ ;$$

$P = (X, Y)$ est un point de 5-torsion. Nous avons alors

$$\begin{aligned} &Y^q = \overline{57}X + Y(\overline{102}X + \overline{52}), \ (q \bmod 5)(X, Y) = O_{E_{\overline{7}}}, \\ &2(X, Y) = (X + \overline{57}, \overline{56}X + \overline{57} + Y(\overline{102}X + \overline{52})). \end{aligned}$$

D'où $(X^q, Y^q) = \kappa(X, Y)$ avec $\kappa = 3$ et finalement $c = \kappa + 2^8/\kappa = 0 \bmod 5$. Comme κ et $2^8/\kappa = 2$ sont d'ordre 4, $h_0(X)$ est irréductible et nous posons $h(X) = h_0(X)$.

Calculons maintenant E_{-1}. Pour cela nous substituons $1/\hat{F}$ à X dans $\Phi_5^c(X, Y)$ et nous trouvons que le polynôme obtenu a pour racine $j_{-1} = \overline{64} = 1/\overline{29}$. Ainsi E_{-1} est définie par

$$E_{-1} \ : Y^2 + XY = X^3 + \overline{29}$$

et est isogène de degré 5 à E via l'isogénie $\mathcal{I}_{-1}$ d'abscisse

$$\frac{g_{-1}(X)}{h_{-1}^2(X)} = \frac{X^5 + \overline{26}\,X^3 + \overline{252}\,X}{(X^2 + \overline{129}\,X + \overline{46})^2}.$$

Soit alors $h(X)$ le numérateur de $h(g_{-1}(X)/h_{-1}^2(X))$, c'est-à-dire

$$\begin{aligned} h(X) = X^{10} + \overline{57}\,X^9 + \overline{74}\,X^8 + \overline{166}\,X^7 + \overline{95}\,X^6 + \overline{36}\,X^5 \\ + \overline{20}\,X^4 + \overline{193}\,X^3 + \overline{22}\,X^2 + \overline{99}\,X + \overline{162}. \end{aligned}$$

Comme précédemment, nous construisons

$$\mathbb{X} = \mathbb{F}_{2^8}[X]/(h(X)), \text{ puis } \mathbb{Y} = \mathbb{X}[Y]/(Y^2 + XY + X^3 + \overline{29}) \ ;$$

$P = (X, Y)$ est un point de 25-torsion. Nous trouvons alors que

$$(X^q, Y^q) = [\kappa]_{E_{-1}}(X, Y) \text{ avec } \kappa = 18,$$

d'où $c = 10 \bmod 25$.

Rapidement, une nouvelle itération du procédé conduit à $\hat{F} = \overline{66}$, donc nous avons à partir de $F = \overline{55}$, E_{-2} qui est donnée par

$$E_{-2} \ : Y^2 + XY = X^3 + \overline{168},$$

et l'abscisse de l'isogénie $\mathcal{I}_{-2}$ est égale à

$$\frac{g_{-2}(X)}{h_{-2}^2(X)} = \frac{X^5 + \overline{181}\,X^3 + \overline{199}\,X}{(X^2 + \overline{102}\,X + \overline{179})^2}.$$

Ainsi le numérateur de $h(g_{-2}(X)/h^2_{-2}(X))$, et donc un facteur de degré 50 du polynôme de 125-division qui est quant à lui de degré 7812, est égal à

$$\begin{aligned}h(X) = {} & X^{50} + \overline{57}\,X^{49} + \overline{74}\,X^{48} + \overline{147}\,X^{47} + \overline{163}\,X^{46} + \overline{220}\,X^{45} + \overline{28}\,X^{44} + \overline{179}\,X^{43} + \overline{119}\,X^{42} \\ & + \overline{176}\,X^{41} + \overline{11}\,X^{40} + \overline{189}\,X^{39} + \overline{72}\,X^{38} + \overline{143}\,X^{37} + \overline{102}\,X^{36} + \overline{169}\,X^{35} + \overline{7}\,X^{34} + \overline{40}\,X^{33} \\ & + \overline{103}\,X^{32} + \overline{33}\,X^{31} + \overline{40}\,X^{30} + \overline{201}\,X^{29} + \overline{82}\,X^{28} + \overline{115}\,X^{27} + \overline{86}\,X^{26} + \overline{67}\,X^{25} + \overline{240}\,X^{24} \\ & + \overline{242}\,X^{23} + \overline{222}\,X^{22} + \overline{215}\,X^{21} + \overline{62}\,X^{20} + \overline{40}\,X^{19} + \overline{45}\,X^{18} + \overline{155}\,X^{17} + \overline{85}\,X^{16} \\ & + \overline{160}\,X^{15} + \overline{127}\,X^{14} + \overline{82}\,X^{13} + \overline{17}\,X^{12} + \overline{169}\,X^{11} + \overline{208}\,X^{10} + \overline{61}\,X^{9} + \overline{214}\,X^{8} \\ & + \overline{36}\,X^{7} + \overline{201}\,X^{6} + \overline{70}\,X^{5} + \overline{99}\,X^{4} + \overline{153}\,X^{3} + \overline{52}\,X^{2} + \overline{10}\,X + \overline{85}.\end{aligned}$$

Nous obtenons finalement $c = 110 \bmod 125$.

Deuxième partie

Calculs d'isogénie entre courbes elliptiques

Chapitre 4

Algorithmes en grande caractéristique

Le cœur de l'algorithme SEA décrit dans ses grandes lignes au chapitre 3 est le calcul pour de petits nombres premiers ℓ d'une isogénie $\mathcal{I}$ de degré ℓ.

Pour des courbes elliptiques définies dans un corps algébriquement clos $\mathbb{K}$, et en particulier dans le corps des complexes $\mathbb{C}$, il est classique de construire une isogénie $\mathcal{I}$ de noyau donné dans l'ensemble des points d'une courbe E_a et d'en déduire la courbe isogène E_b. Les formules de Vélu sont un de ces moyens et nous en rappelons la substance dans la section 4.1.

Pour SEA, nous avons le problème inverse ; nous connaissons E_b et souhaitons calculer $\mathcal{I}$ pour finalement en déduire son noyau. Sur $\mathbb{C}$, les courbes elliptiques sont aisées à manipuler une fois identifiées au quotient de $\mathbb{C}$ par un réseau (cf. chapitre 2). Dans ce cadre, la paramétrisation des points par la fonction de Weierstrass $\wp$ joue un rôle central et il est possible d'en dériver des algorithmes élégants comme nous le rappelons dans la section 4.2.

Dans un corps fini $\mathbb{F}_q$ de caractéristique p, tout se complique. Certes, on peut toujours continuer à appliquer les formules de Vélu dans $\bar{\mathbb{F}}_q$ (section 4.1). Par contre, il s'avère que la perte du développement en série de Laurent de la fonction de Weierstrass $\wp$ est irrémédiable, et seule la conservation dans $\mathbb{F}_q$ de certaines équations algébriques vérifiées sur $\mathbb{C}$ permettent de nous tirer d'affaire pour $p \geqslant 5$ et $\ell < p$ (section 4.2). Pour ces algorithmes nous donnons dans la section 4.3 des temps précis obtenus avec notre implantation, temps qui servent de référence aux algorithmes que nous décrivons dans les chapitres suivants pour résoudre le cas $\ell \geqslant p$.

4.1 Formules de Vélu

Les travaux de Vélu fournissent l'expression complète d'une isogénie une fois son noyau fixé. Rappelons à ce sujet que tout sous-groupe fini d'ordre ℓ de $E_a(\mathbb{K})$ est le noyau d'une isogénie d'après le théorème 13. Après en avoir rappelé la formulation aux corps algébriquement clos [113], nous montrons comment en étendre la portée aux isogénies de degrés impairs définies sur un corps fini [36] avant de finalement expliquer comment leurs utilisations conjointes à celle du polynôme modulaire $\Phi_2^c(F, J)$ permet le calcul complet des isogénies de degré 2 nécessaires à l'algorithme SEA [33].

4.1.1 Corps algébriquement clos

À partir d'un sous-groupe fini F de l'ensemble des points d'une courbe elliptique E_a définie sur un corps commutatif algébriquement clos $\mathbb{K}$, Vélu exhibe une isogénie $\mathcal{I}$ ayant pour noyau F. Cette isogénie

est donnée par les expressions suivantes :

$$\begin{array}{rcl} E_a(\mathbb{K}) & \to & E_b(\mathbb{K}) \\ P & \mapsto & \left\{ \begin{array}{ll} O_{E_b} & \text{si } P = O_{E_a}, \\ \left(X_P + \displaystyle\sum_{Q \in F - \{O_{E_a}\}} X_{P+Q} - X_Q \ , \right. & \\ \qquad \left. Y_P + \displaystyle\sum_{Q \in F - \{O_{E_a}\}} Y_{P+Q} - Y_Q \right) & \text{si } P = (X_P, Y_P). \end{array} \right. \end{array} \tag{4.1}$$

Par application des formules de la loi d'addition sur E_a du chapitre 2, il n'est alors pas difficile d'obtenir les expressions algébriques suivantes pour E_b et $\mathcal{I}$.

Théorème 39. *Avec les notations du théorème 5, soient E_a une courbe elliptique définie sur un corps commutatif algébriquement clos $\mathbb{K}$, F un sous-groupe fini de $E_a(\mathbb{K})$, R le sous-ensemble de F défini par $F - E_a[2] = R \cup (-R)$ avec $R \cap -R = \emptyset$ et $S = F \cap E_a[2] - \{O_{E_a}\}$. Soient de plus pour tout point $Q = (X_Q, Y_Q) \in F - \{O_{E_a}\}$,*

$$\begin{array}{rcll} g_Q^r & = & 3X_Q^2 + 2a_2X_Q + a_4 - a_1Y_Q, & \\ g_Q^y & = & -2Y_Q - a_1X_Q - a_3, & \\ t_Q & = & \left\{ \begin{array}{ll} g_Q^r & \text{si } Q \in S, \\ 2g_Q^r - a_1g_Q^y = 6X_Q^2 + d_2X_Q + d_4 & \text{sinon,} \end{array} \right. & \\ u_Q & = & (g_Q^y)^2 = 4X_Q^3 + d_2X_Q^2 + 2d_4X_Q + d_6 & \end{array}$$

et

$$t = \sum_{Q \in R \cup S} t_Q, \ w = \sum_{Q \in R \cup S} u_Q + X_Q t_Q.$$

Alors, la correspondance

$$\begin{array}{rcl} E_a(\mathbb{K}) & \to & E_b(\mathbb{K}) \\ P & \mapsto & \left\{ \begin{array}{ll} O_{E_b} & \text{si } P = O_{E_a}, \\ (X_{\mathcal{I}(P)}, Y_{\mathcal{I}(P)}) & \text{si } P = (X_P, Y_P), \end{array} \right. \end{array}$$

où E_b est définie sur $\mathbb{K}$ par

$$Y^2 + b_1XY + b_3Y = X^3 + b_2X^2 + b_4X + b_6,$$

avec $b_1 = a_1$, $b_2 = a_2$, $b_3 = a_3$, $b_4 = a_4 - 5t$ et $b_6 = a_6 - d_2t - 7w$ et $(X_{\mathcal{I}(P)}, Y_{\mathcal{I}(P)})$ est défini par

$$\left\{ \begin{array}{rcl} X_{\mathcal{I}(P)} & = & X_P + \displaystyle\sum_{Q \in R \cup S} \left(\frac{t_Q}{X_P - X_Q} + \frac{u_Q}{(X_P - X_Q)^2} \right), \\ Y_{\mathcal{I}(P)} & = & Y_P + \displaystyle\sum_{Q \in R \cup S} \left(u_Q \frac{2Y_P + a_1X_P + a_3}{(X_P - X_Q)^3} \right. \\ & & \left. + t_Q \dfrac{a_1(X_P - X_Q) + Y - Y_Q}{(X_P - X_Q)^2} + \dfrac{a_1u_Q - g_Q^r g_Q^y}{(X_P - X_Q)^2} \right), \end{array} \right. \tag{4.2}$$

est une isogénie de degré $\#F$.

4.1.2 Reformulation dans un corps fini

Le théorème 39 est difficilement exploitable pour une courbe E_a définie sur un corps fini $\mathbb{K}$. Lorsque $\ell = \#F$ est impair, $d = (\ell - 1)/2$ et le polynôme

$$h(X) = \prod_{Q \in R} (X - X_Q) = X^d - h_1X^{d-1} + h_2X^{d-2} - h_3X^{d-3} + \cdots + (-1)^d h_0$$

est à coefficients dans $\mathbb{K}$ bien que les X_Q soient seulement définis dans $\overline{\mathbb{K}}$, il est néanmoins possible de considérer la courbe comme définie sur $\overline{\mathbb{K}}$, d'y effectuer les calculs puis de ramener, grâce à des propriétés de conjugaisons, le résultat dans $\mathbb{K}$ afin d'obtenir une courbe E_b et des expressions rationnelles pour $\mathcal{I}$ définies dans $\mathbb{K}$ [36].

Le cœur en est l'évaluation dans $\mathbb{K}$ des quantités suivantes,

$$\begin{aligned}
&S_1 = \sum_{Q\in R} X_Q = h_1,\ S_2 = \sum_{Q\in R} X_Q^2 = h_1^2 - 2h_2, S_3 = \sum_{Q\in R} X_Q^3 = h_1^3 - 3h_1h_2 + 3h_3,\\
&\Sigma_1(X) = \sum_{Q\in R} \frac{1}{X - X_Q} = \frac{h'}{h}(X),\ \Sigma_2(X) = \sum_{Q\in R} \frac{1}{(X-X_Q)^2} = -\left(\frac{h'}{h}\right)'(X),\\
&\Sigma_3(X) = \sum_{Q\in R} \frac{1}{(X-X_Q)^3} = \frac{1}{2}\left(\frac{h'}{h}\right)''(X) = \frac{1}{h^3(X)}\left(h^2\frac{h'''}{2} - 3hh'\frac{h''}{2} + h'^3\right)(X).
\end{aligned}$$

Notons que ces formules sont aussi valides dans des corps finis de caractéristique deux, notamment celle de $\Sigma_3(X)$ en calculant formellement $h''(X)$ et $h'''(X)$ et en divisant par 2 avant de réduire modulo 2. Il est alors facile d'obtenir les coefficients de E_b à l'aide de

$$\left\{\begin{array}{rcl} t &=& 6S_2 + d_2S_1 + d_4d,\\ w &=& 10S_3 + 2d_2S_2 + 3d_4S_1 + d_6d.\end{array}\right.$$

Puis à partir des formules (4.2), nous avons

$$X_{\mathcal{I}(P)} = X_P + 2dX_P - 2h_1 - (6X_P^2 + d_2X_P + d_4)\Sigma_1(X_P) + (4X_P^3 + d_2X_P^2 + 2d_4X_P + d_6)\Sigma_2(X_P).$$

et

$$Y_{\mathcal{I}(P)} = Y\frac{\partial X_{\mathcal{I}(P)}}{\partial X_P} - a_3d - a_1S_1 + c_1(X_P)\Sigma_1(X_P) + c_2(X_P)\Sigma_2(X_P) + c_3(X_P)\Sigma_3(X_P),$$

avec

$$c_1 = Af'' - A'f', c_2 = -\left(A'(A^2 - 2f) + 3Af'\right), c_3 = A(4f + A^2),$$

où

$$f(X) = X^3 + a_2X^2 + a_4X + a_6 \text{ et } A(X) = a_1X + a_3.$$

4.1.3 Isogénies de degré 2 sur un corps fini

Les formules de Vélu permettent le calcul complet des isogénies de degré 2. Nous nous plaçons dans la suite sur l'une des courbes canoniques à j-invariant différent de 1728 du tableau 2.1. C'est pourquoi, nous séparons notre étude en trois parties ; les corps finis de caractéristique deux puis, en suivant [33], ceux de caractéristique trois et de caractéristique supérieure à cinq.

Corps finis de caractéristique deux

Pour une courbe canonique non supersingulière E_a de $\mathbb{F}_{2^n}$, nous avons $a_1 = 1$ et $a_3 = a_4 = 0$ et il n'est alors pas difficile de se rendre compte qu'une telle courbe a un unique point d'ordre 2, $Q = (0, \tilde{a}_6)$ à partir duquel il est possible de construire une isogénie de degré 2.

Tout d'abord, par simple application des formules de Vélu à $F = \{O_{E_a}, Q\}$, nous avons E_a qui est isogène de degré 2 à

$$Y^2 + XY = X^3 + a_2X^2 + \tilde{a}_6X + a_6 + \tilde{a}_6,$$

via l'isogénie

$$(X, Y) \mapsto \left(X + \frac{\tilde{a}_6}{X}, Y + \frac{1}{X} + \tilde{a}_6\frac{Y + \tilde{a}_6}{X^2}\right).$$

Pour se ramener à une courbe canonique E_b, il ne reste plus qu'à appliquer le changement de variable $Y \mapsto Y + \tilde{a}_6$ pour finalement obtenir que la courbe E_a est isogène de degré 2 à la courbe canonique E_b définie par

$$E_b \ : Y^2 + XY = X^3 + a_2X^2 + \tilde{a}_6,$$

via l'isogénie

$$\mathcal{I} \ : (X, Y) \mapsto \left(X + \frac{\tilde{a}_6}{X}, Y + \tilde{a}_6 + \frac{1}{X} + \tilde{a}_6 \frac{Y + \tilde{a}_6}{X^2} \right).$$

Corps finis de caractéristique trois

Pour une courbe canonique non supersingulière E_a de $\mathbb{F}_{3^n}$, nous avons $a_1 = a_3 = a_4 = 0$. Une isogénie de degré 2 est définie à partir de E_a si et seulement si cette courbe possède un point $Q = (x_0, 0)$ d'ordre 2. Avec les formules de Vélu, une isogénie de degré 2 est alors donnée par

$$(X, Y) \mapsto \left(\frac{X^2 - x_0X + 2a_2x_0}{X - x_0}, Y \left(1 - \frac{2a_2x_0}{(X - x_0)^2} \right) \right)$$

de la courbe E_a vers la courbe définie par

$$Y^2 = X^3 + a_2X^2 + 2a_2X + a_6 + a_2^2x_0 + a_2x_0^2.$$

Il ne reste plus qu'à appliquer le changement de variable $X \mapsto X + 2x_0$, pour finalement obtenir que la courbe E_a est isogène de degré 2 à la courbe canonique E_b définie par

$$E_b \ : Y^2 = X^3 + a_2X^2 + a_6 + a_2^2x_0 + 2x_0^3,$$

via l'isogénie

$$\mathcal{I} \ : (X, Y) \mapsto \left(\frac{X^2 - a_2x_0 - x_0^2}{X - x_0}, Y \left(1 - \frac{2a_2x_0}{(X - x_0)^2} \right) \right).$$

Pour déterminer x_0, il est bien sûr possible de factoriser le polynôme $X^3 + a_2X^2 + a_6$. Dans le cadre de l'algorithme `SEA`, une alternative particulièrement utile pour calculer la cardinalité d'une courbe modulo une puissance de 2 consiste à procéder comme pour les isogénies de degré impair, c'est-à-dire substituer j_{E_a} à J dans le polynôme modulaire canonique

$$\Phi_2^c(F, J) = F^3 + 48F^2 + (768 - J)F + 4096$$

et chercher si le polynôme de degré 3 obtenu a au moins une racine F (non nulle puisque $4096 \equiv 1 \bmod 3$). Lorsque cela est le cas, cela signifie que $X^3 + a_2X^2 + a_6$ a une racine x_0 que nous déterminons en fonction de F, en écrivant d'une part que l'invariant de E_b est racine de $\Phi_2^c(F, J)$ et d'autre part que $\mathcal{I}(P) \in E_b(\mathbb{K})$ pour tout point P de E_a. Quelques résultants entre les équations algébriques obtenues suffisent alors pour trouver

$$x_0 = \frac{a_6(F + 1)^2}{a_2^2F}.$$

Corps finis de caractéristique supérieure à cinq

Pour une courbe canonique E_a de $\mathbb{F}_{p^n}$ avec $p \geqslant 5$, nous avons $a_1 = a_3 = a_2 = 0$. Avec les mêmes notations que dans le cas des corps finis de caractéristique 3, une isogénie $\mathcal{I}$ de degré 2 de la courbe E_a vers la courbe E_b définie par

$$E_b \ : Y^2 = X^3 - (15x_0^2 + 4a_4)X + a_6 - 7(3x_0^3 + a_4x_0)$$

est alors donnée avec les formules de Vélu par

$$(X, Y) \mapsto \left(\frac{X^2 - x_0X + 3x_0^2 + a_4}{X - x_0}, Y \left(1 - \frac{3x_0^2 + a_4}{(X - x_0)^2} \right) \right).$$

Puis, avec la même méthode que celle utilisée pour les corps finis de caractéristique trois, nous trouvons

$$x_0 = -\frac{3a_6(F+16)}{2a_4(F-8)}$$

où F est une racine de $\Phi_2^c(F, j_{E_a})$. Remarquons que $F \neq 8$ dès que $j_{E_a} \neq 1728$.

4.2 Calcul d'isogénie en grande caractéristique

Dans le corps des complexes $\mathbb{C}$, calculer une isogénie de degré ℓ entre deux courbes E_a et E_b est un problème classique. Lorsque de plus, la demi somme p_1 des abscisses des points non nuls du noyau de $\mathcal{I}$ est comme dans l'algorithme `SEA` connue, il est même possible de n'obtenir que le polynôme dont les racines sont les abscisses des points du noyau de $\mathcal{I}$. Une fois donnés deux algorithmes pour cela, nous expliquons quels enseignements en tirer pour le cas des corps finis.

4.2.1 Cas de $\mathbb{C}$

Soit $E_a : Y^2 = X^3 + a_4X + a_6$ une courbe elliptique définie sur $\mathbb{C}$. Elle est alors isomorphe au quotient de $\mathbb{C}$ par un réseau $\omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ (cf. chapitre 2). Lorsqu'une isogénie $\mathcal{I}$ de degré ℓ existe à partir de E_a, nous obtenons alors avec les algorithmes du chapitre 3 une courbe isogène $E_b : Y^2 = X^3 + b_4X + b_6$ isomorphe au quotient de $\mathbb{C}$ par le réseau $\frac{\omega_1}{\ell}\mathbb{Z} + \omega_2\mathbb{Z}$ ainsi que p_1, la demi somme des abscisses des points non nuls du noyau de $\mathcal{I}$. Vu en terme de réseau, $\mathcal{I}$ est particulièrement simple puisqu'il s'agit de

$$\begin{array}{ccc} \mathbb{C}/(\omega_1\mathbb{Z} + \omega_2\mathbb{Z}) & \to & \mathbb{C}/(\frac{\omega_1}{\ell}\mathbb{Z} + \omega_2\mathbb{Z}), \\ z & \mapsto & z. \end{array}$$

Nous notons $\wp_a(z)$ et $\wp_b(z)$ les fonctions de Weierstrass respectivement associées à E_a et E_b. En posant $c_k = (2k+1)G_{2k}$ pour $k \in \mathbb{N}^*$, nous avons

$$\wp_a(z) = \frac{1}{z^2} + \sum_{k=1}^{\infty} c_{a_k} z^{2k} \text{ et } \wp_b(z) = \frac{1}{z^2} + \sum_{k=1}^{\infty} c_{b_k} z^{2k} \tag{4.3}$$

avec

$$c_{a_1} = -\frac{a_4}{5},\ c_{a_2} = -\frac{a_6}{7},\ c_{a_k} = \frac{3}{(k-2)(2k+3)} \sum_{j=1}^{k-2} c_{a_j} c_{a_{k-1-j}} \text{ pour } k \geqslant 3,$$

et de même

$$c_{b_1} = -\frac{b_4}{5},\ c_{b_2} = -\frac{b_6}{7},\ c_{b_k} = \frac{3}{(k-2)(2k+3)} \sum_{j=1}^{k-2} c_{b_j} c_{b_{k-1-j}} \text{ pour } k \geqslant 3.$$

Remarquons déjà qu'une relation analogue aux équations (4.1) relie $\wp_a(z)$ à $\wp_b(z)$.

Théorème 40. *Soient $\wp_a(z)$ et $\wp_b(z)$ deux fonctions de Weierstrass respectivement associées aux réseaux $\omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ et $\frac{\omega_1}{\ell}\mathbb{Z} + \omega_2\mathbb{Z}$. Alors*

$$\forall z \in \mathbb{C},\ \wp_b(z) = \sum_{i=0}^{l-1} \wp_a(z + i\frac{\omega_1}{\ell}) - \sum_{i=0}^{l-1} \wp_a(i\frac{\omega_1}{\ell}).$$

Notons $I(X) = g(X)/h^2(X)$ l'image par $\mathcal{I}$ d'un point (X, Y) de $E_a(\mathbb{C})$. Nous avons

$$\wp_b(z) = I(\wp_a(z)).$$

En remplaçant $\wp_a(z)$ et $\wp_b(z)$ par leurs développements en séries de Laurent, cette dernière relation suffit au calcul de I. Néanmoins, d'après les formules de Vélu, seul le calcul de $h(X)$ égal ici à

$$h(X) = \prod_{i=1}^{(\ell-1)/2} \left(X - \wp_a\left(i\frac{\omega_1}{\ell}\right)\right).$$

est nécessaire pour déterminer complètement $\mathcal{I}$. Plusieurs algorithmes sont connus pour calculer directement $h(X)$. Nous rappelons les deux principaux, l'algorithme `CCR` [26] et l'algorithme d'Atkin [5].

Algorithme `CCR`

Nous reprenons ici la présentation de Charlap, Coley et Robbins reprise par ailleurs par Morain [91]. Ainsi, nous procédons en deux étapes. Dans un premier temps, nous calculons les sommes

$$p_k = \sum_{i=1}^{(\ell-1)/2} \wp_a^k \left(i\frac{\omega_1}{\ell} \right), \ k \in \mathbb{N},$$

à partir desquelles il est facile dans un deuxième temps d'obtenir $h(X)$ avec les "formules de Newton" [27, p. 161]. À cet effet, nous dérivons une fois l'équation différentielle satisfaite par $\wp_a$ (cf. théorème 19),

$$\left(\frac{d\wp_a}{dz} \right)^2 = 4\wp_a^3 + 4a_4\wp_a + 4a_6,$$

pour obtenir

$$\frac{d^2\wp_a}{dz^2} = 6\wp_a^2 + 2a_4.$$

En dérivant par deux fois cette relation, nous trouvons

$$\frac{d^4\wp_a}{dz^4} = 120\wp_a^3 + 72a_4\wp_a + 48a_6$$

et plus généralement

$$\forall k \in \mathbb{N}^*, \forall z \in \mathbb{C}, \frac{d^{2k}\wp_a}{dz^{2k}}(z) = (2k+2)!\wp_a^{k+1}(z) + \cdots$$

Nous substituons alors $\frac{\omega_1}{\ell}$, $\ldots$, $\frac{(\ell-1)\omega_1}{\ell}$ à z dans cette équation et nous sommons les relations obtenues pour avoir,

$$\forall k \in \mathbb{N}^*, \sum_{i=1}^{l-1} \frac{d^{2k}\wp_a}{dz^{2k}} \left(i\frac{\omega_1}{\ell} \right) = 2\left((2k+2)!p_{k+1} + \cdots \right).$$

On démontre à partir du théorème 40 que

$$\sum_{i=1}^{l-1} \frac{d^{2k}\wp_a}{dz^{2k}} \left(i\frac{\omega_1}{\ell} \right) = (2k)!(c_{b_k} - c_{a_k})$$

et donc

$$(2k)!(c_{b_k} - c_{a_k}) = 2(2k+2)!\wp_a^{k+1}(z) + \cdots \tag{4.4}$$

En particulier,

$$\begin{aligned} a_4 - b_4 &= 5(6p_2 + 2a_4p_0), \\ a_6 - b_6 &= 7(10p_3 + 6a_4p_1 + 4a_6p_0). \end{aligned}$$

Ainsi, le calcul des p_k revient à inverser un système linéaire triangulaire.

Algorithme d'Atkin

L'idée de l'algorithme d'Atkin est de calculer la série $h(\wp_a(z))$ puisqu'à partir de celle-ci il est ensuite aisé d'obtenir $h(X)$. Pour cela, nous cherchons en suivant [103] à calculer la dérivée logarithmique de $h(\wp_a(z))$, c'est-à-dire

$$\sum_{i=1}^{(l-1)/2} \frac{\wp_a'(z)}{\wp_a(z) - \wp_a\left(i\frac{\omega_1}{\ell}\right)}.$$

Soient $\zeta_a(z)$ et $\zeta_b(z)$, deux primitives de $-\wp_a(z)$ et $-\wp_b(z)$ données en séries de Laurent par

$$\zeta_a(z) = \frac{1}{z} + \sum_{k=1}^{\infty} \frac{c_{a_k}}{2k+1} z^{2k+1} \text{ et } \zeta_b(z) = \frac{1}{z} + \sum_{k=1}^{\infty} \frac{c_{b_k}}{2k+1} z^{2k+1}.$$

Le lemme suivant permet de relier la dérivée logarithmique de $\wp_a$ à ζ_a.

Lemme 5. *Soient $\wp(z)$ et $\zeta(z)$, les fonctions ζ et $\wp$ de Weierstrass associées à un réseau $\omega_1\mathbb{Z} + \omega_2\mathbb{Z}$. Alors*

$$\forall (\delta, z) \in \mathbb{C}^2, \zeta(z+\delta) + \zeta(z-\delta) - 2\zeta(z) = \frac{\wp'(z)}{\wp(z) - \wp(\delta)}.$$

Et donc

$$\sum_{i=1}^{(l-1)/2} \frac{\wp_a'(z)}{\wp_a(z) - \wp_a\left(i\frac{\omega_1}{\ell}\right)} = -\ell\zeta_a(z) + \sum_{-\ell/2<i<\ell/2} \zeta_a\left(i\frac{\omega_1}{\ell}\right).$$

À partir du théorème 40, il est alors facile d'obtenir le résultat suivant.

Lemme 6. *Soient $\zeta_a(z)$ et $\zeta_b(z)$, deux fonctions ζ de Weierstrass respectivement associées aux réseaux $\omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ et $\frac{\omega_1}{\ell}\mathbb{Z} + \omega_2\mathbb{Z}$. Alors*

$$\zeta_b(z) = \sum_{-\ell/2<i<\ell/2} \zeta_a\left(z + i\frac{\omega_1}{\ell}\right) + z \sum_{-\ell/2<i<\ell/2} \zeta_a\left(i\frac{\omega_1}{\ell}\right).$$

En conséquence,

$$\sum_{i=1}^{(l-1)/2} \frac{\wp_a'(z)}{\wp_a(z) - \wp_a\left(i\frac{\omega_1}{\ell}\right)} = -\ell\zeta_a(z) + \zeta_b(z) - p_1 z,$$

et par intégration logarithmique, nous obtenons le théorème suivant.

Théorème 41. *Soient $\wp_a(z)$ et $\wp_b(z)$, deux fonctions de Weierstrass respectivement associées aux réseaux $\omega_1\mathbb{Z} + \omega_2\mathbb{Z}$ et $\frac{\omega_1}{\ell}\mathbb{Z} + \omega_2\mathbb{Z}$. Alors*

$$\prod_{i=1}^{(\ell-1)/2} \left(\wp_a(z) - \wp_a\left(i\frac{\omega_1}{\ell}\right)\right) = z^{1-l} \exp\left(-\frac{1}{2}p_1 z^2 - \sum_{k=1}^{\infty} \frac{c_{b_k} - l c_{a_k}}{(2k+1)(2k+2)} z^{2k+2}\right). \tag{4.5}$$

4.2.2 Application aux corps finis

Tout d'abord, notons que la paramétrisation des courbes elliptiques utilisées sur $\mathbb{C}$ nous restreint au cas des corps finis de caractéristique p plus grande que cinq. De plus, la décomposition en série de Laurent de $\wp_a(z)$ n'est plus valide dans un corps fini. Il suffit pour s'en convaincre de remarquer que les expressions de c_{a_k} et c_{b_k} nécessitent l'inversion modulo p de $(k-1)(2k+3)$ qui peut être nul dès que $k \geqslant (p-3)/2$.

Il est par contre possible de remplacer ces séries par les sommes partielles

$$\frac{1}{z^2} + \sum_{k=1}^{n} c_{a_k} z^{2k} \text{ et } \frac{1}{z^2} + \sum_{k=1}^{n} c_{b_k} z^{2k} \text{ pour } n < p/2.$$

Dans ce cas les relations algébriques des algorithmes précédents qui n'utilisent que des coefficients c_{a_k} et c_{b_k} définis restent utilisables. Un examen attentif des calculs mis en jeu dans ces algorithmes montre qu'en général il est ainsi possible d'obtenir au plus $(p-3)/2$ coefficients du polynôme $h(X)$, ce qui est suffisant dès que $\ell < p$. Dans le cadre de l'algorithme `SEA`, il se peut que nous ayons à utiliser ces algorithmes avec des courbes E_a et E_b non isogènes. Pour détecter ce cas, il est classique de vérifier, une fois les coefficients de $h(X)$ calculés, qu'une ou deux relations algébriques supplémentaires fournies par

les équations (4.4) ou (4.5) sont satisfaites. Finalement, ces algorithmes permettent donc le calcul d'une isogénie de degré ℓ impair si $\ell < p - 4$.

Si $\ell \geqslant p \geqslant 5$, nous n'obtenons qu'une partie du polynôme $h(X)$. Il est néanmoins possible en conjonction avec une variante de l'algorithme de Couveignes (cf. chapitre 5) d'en tirer parti comme nous le décrivons au chapitre 8. Pour les corps finis de caractéristique deux et trois, force nous est de constater que ces techniques ne nous sont d'aucune utilité.

Complexités : comme le coût majeur de l'algorithme `CCR` est l'inversion d'un système linéaire triangulaire, la complexité en est $O(\ell^2)$ multiplications dans $\mathbb{F}_q$.

Avec l'algorithme d'Atkin, il est par contre nécessaire de composer deux séries pour obtenir la partie droite de l'équation (4.5), ce qui nécessite $O(\ell^3)$ multiplications dans $\mathbb{F}_q$ avec des algorithmes classiques. Avec l'algorithme de composition de Brent et Kung [15] utilisé en conjonction avec un algorithme de multiplication de séries basé sur un schéma de transformée de Fourier discrète, il est néanmoins possible de réaliser cette composition en $O(\ell^{\frac{3}{2}+\epsilon})$. À ce moment là, le coût majeur devient le calcul des c_{b_k} et l'algorithme d'Atkin a donc aussi pour complexité $O(\ell^2)$ multiplications dans $\mathbb{F}_q$.

4.3 Résultats

Pour tout corps fini $\mathbb{F}_q$ de caractéristique p plus grande que cinq, nous avons repris à l'aide de la librairie `ZEN` (cf. chapitre 10) l'implantation de l'algorithme `CCR` réalisée par Morain pour $\mathbb{Z}/p\mathbb{Z}$ [91]. Vus les temps qui sont en pratique clairement négligeables vis-à-vis des autres calculs nécessaires dans l'algorithme `SEA`, nous n'avons pas jugé utile de programmer l'algorithme d'Atkin.

Pour illustrer ce phénomène mais aussi servir de référence aux algorithmes que nous décrivons dans les chapitres suivants, nous avons mesuré le temps nécessaire au calcul d'isogénies de degrés premiers et impairs ℓ croissants définies sur $\mathbb{F}_{4294967291}$. Pour un ℓ donné, la courbe utilisée est une courbe de la forme E_a : $Y^2 = X^3 + X + a$ où a est le plus petit entier tel qu'une isogénie de degré ℓ soit définie à partir de E_a. Les résultats sont donnés sur la courbe 4.1.

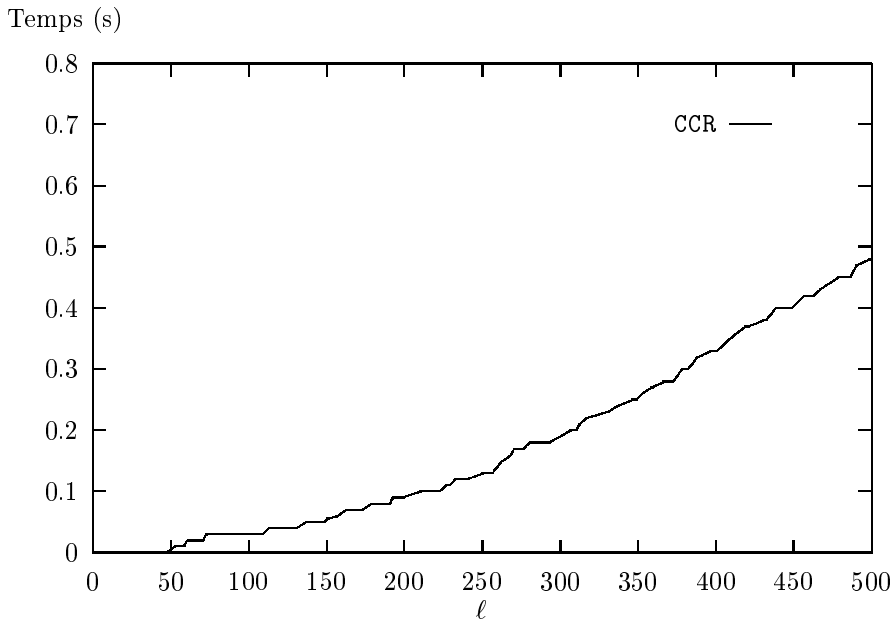


FIG. 4.1 – Temps de calcul d'isogénies de degré ℓ avec l'algorithme `CCR` (station DEC).

Chapitre 5

Premier algorithme de Couveignes en petite caractéristique

Après la découverte par Schoof en 1985 de son algorithme de calcul de la cardinalité d'une courbe elliptique, le corps de définition sous-jacent aux nombreuses améliorations développées jusqu'en 1994 était $\mathbb{Z}/p\mathbb{Z}$ pour de "grands" nombres premiers p. Au vu des résultats impressionnants obtenus en pratique pour ces derniers [34, 94], généraliser ces idées aux autres corps finis est une démarche naturelle. Comme déjà esquissé en 1992 par Menezes, Vanstone et Zuccherato [80] et comme nous l'expliquons dans la partie I, les techniques initialement développées pour $\mathbb{Z}/p\mathbb{Z}$ sont utilisables, à quelques complications purement techniques près, à l'ensemble des corps finis $\mathbb{F}_q$, excepté le calcul d'isogénie d'Atkin et d'Elkies du chapitre 4.

Pour calculer une isogénie, l'algorithme d'Atkin-Elkies explicite la correspondance entre les paramétrisations de Weierstrass associées classiquement aux points (X, Y) des deux courbes isogènes ; X développé en série de Fourier en un paramètre τ et $Y = \partial X/\partial\tau$. Cette paramétrisation n'est malheureusement pas valide pour des caractéristiques p "petites". Nous devons à Couveignes [30] le premier algorithme utilisable dans ce cas. Il consiste à remplacer la paramétrisation de Weierstrass par celle, utilisable pour tout corps fini, des groupes formels où X est développé en série de Laurent en un paramètre t et $Y = -X/t$.

Nous expliquons ici comment mettre en œuvre efficacement cet algorithme[1]. Une fois énoncés les prérequis nécessaires à la compréhension de l'ensemble (section 5.1), nous rappelons quelles sont les idées de Couveignes (section 5.2) et décrivons ensuite dans le détail les algorithmes utilisés (section 5.3) pour finalement donner des exemples d'utilisation et des statistiques obtenues avec notre implantation (section 5.4).

Notre contribution est d'avoir incorporé à l'algorithmique de Couveignes l'ensemble des techniques algorithmiques nécessaires à la réalisation d'une implantation efficace. Ce qui nous a permis d'effectuer pour la première fois en 1995 le calcul du nombre de points d'une courbe elliptique définie dans $\mathbb{F}_{2^{1009}}$.

Remarque : nous nous restreignons au cas des courbes non supersingulières.

5.1 Prérequis

L'une des propriétés utilisées par Couveignes est la conservation par les isogénies de la p-torsion. Nous en mettons donc en lumière quelques aspects qui nous seront utiles dans les sections 5.2 et 5.3. Puis nous rappelons ce qu'est un groupe formel associé à une courbe elliptique.

[1] Ce chapitre reprend un rapport de recherche [70] soumis par ailleurs pour publication [71].

5.1.1 Points de p-torsion

Les points de p-torsion forment un sous-groupe cyclique d'ordre p de la courbe elliptique considérée sur $\bar{\mathbb{F}}_{p^n}$ (cf. théorème 26). Il leur correspond donc $(p-1)/2$ abscisses distinctes. Or, d'après le théorème 11, les racines du polynôme $f_p(X)$ de p-division $f_p(X)$, qui est a priori de degré $(p^2-1)/2$, sont aussi les abscisses de points de p-division. L'explication de ce petit mystère est qu'en fait $f_p(X)$ est la puissance p-ième d'un polynôme $h_p(X)$ de degré $(p-1)/2$.

Pour le démontrer, nous pouvons par exemple utiliser un théorème de Kronecker [11] par lequel le polynôme modulaire $\Phi_p(X,Y)$ satisfait

$$\Phi_p(X,Y) \equiv (X^p - Y)(X - Y^p) \bmod p.$$

Ce qui montre immédiatement que si E_a et E_b sont p-isogènes, alors leurs j-invariants vérifient $j_{E_b} = j_{E_a}^p$ ou $j_{E_b} = j_{E_a}^{1/p}$. Dans le premier cas, les coefficients de la courbe E_b sont les puissances p-ième de ceux de E_a et nous notons $E_b = E_a^p$, sinon les coefficients de la courbe E_b sont les racines p-ième de ceux de E_a et nous notons $E_b = \tilde{E}_a$. Il existe alors des isogénies inséparables de degré p de $\tilde{E}_a$ vers E_a ou de E_a vers E_a^p données par $\phi_p : (X,Y) \mapsto (X^p, Y^p)$ dont la composition avec leurs isogénies duales (séparables quant à elles) permet de décomposer la multiplication par p sur E_a.

Proposition 14. *La multiplication par p sur une courbe elliptique E est donnée par*

$$[p]_E(X,Y) = (F_p(X)^p, G_p(X,Y)^p) \tag{5.1}$$

où $F_p(X)$ et $G_p(X,Y)$ sont deux fractions rationnelles.

Comme $f_p(X)$ apparaît au dénominateur des coordonnées de $[p]_{E_a}$, nous avons $f_p(X) = h_p^p(X)$ car $h_p(X)$ apparaît au dénominateur de $F_p(X)$ et $G_p(X,Y)$.

Lorsque p est égal à 2 ou 3, il n'est pas difficile d'obtenir $F_p(X)$ et $G_p(X,Y)$ à partir de $[2]_{E_a}$ ou $[3]_{E_a}$. Pour des caractéristiques p plus grandes, il est préférable de calculer directement $h_p(X)$ avec les idées de Gunji [45] plutôt que de calculer naïvement $f_p(X)$ pour ensuite en prendre la racine p-ième.

Idées de Gunji

Nous n'utiliserons ces résultats que pour des corps finis de caractéristique $p \geqslant 5$. Nous nous limitons donc exceptionnellement dans ce paragraphe à ce cas. Les idées de Gunji permettent d'obtenir l'abscisse d'un point de p-torsion d'une courbe $E : Y^2 = X^3 + a_4X + a_6$ dans l'extension de $\mathbb{F}_q$ qui contient une racine $(p-1)$-ième du coefficient de Hasse H_E de E défini au chapitre 2, c'est-à-dire $\mathbb{F}_q(\sqrt[p-1]{H_E})$ (si $\mathbb{F}_q$ contient $\sqrt[p-1]{H_E}$, l'abscisse du point de p-division obtenu est alors bien sûr définie dans $\mathbb{F}_q$) [45, p. 156].

Théorème 42. *Soient $r = (p-1)/2$ et $\gamma = \sqrt[p-1]{H_E}$. Alors un point non nul de p-torsion (x,y) est défini dans $\mathbb{F}_q(\gamma)$. Soient de plus $\alpha_\nu = \nu(\nu-1)a_6$, $\beta_\nu = \nu(\nu-1/2)a_4$, $\delta_\nu = \nu(\nu+1/2)$ pour $\nu = 1, 2, \ldots, r$. Si Δ_0 et Δ_1 sont les déterminants $r \times r$ et $(r-1)\times(r-1)$ définis par*

$$\Delta_0 = \begin{vmatrix} \beta_1 & \alpha_2 & 0 & 0 & \cdots & 0 \\ \delta_1 & -\gamma^2 & \beta_3 & \alpha_4 & \ddots & \vdots \\ 0 & \delta_2 & -\gamma^2 & \beta_4 & \ddots & \vdots \\ \vdots & \ddots & \delta_3 & -\gamma^2 & \ddots & \alpha_r \\ \vdots & & \ddots & \ddots & \ddots & \beta_r \\ 0 & \cdots & 0 & & \delta_{r-1} & -\gamma^2 \end{vmatrix} \text{ et } \Delta_1 = \begin{vmatrix} -\gamma^2 & \beta_3 & \alpha_4 & \cdots & 0 \\ \delta_2 & -\gamma^2 & \beta_4 & \ddots & \vdots \\ 0 & \delta_3 & -\gamma^2 & \ddots & \alpha_r \\ \vdots & \ddots & \ddots & \ddots & \beta_r \\ 0 & \cdots & 0 & \delta_{r-1} & -\gamma^2 \end{vmatrix},$$

alors

$$x^p = \frac{\Delta_0^2 - b\gamma^2\Delta_1^2}{4\gamma^2}.$$

Il est ensuite facile d'obtenir le polynôme $h_p(X)$ à partir de x comme nous l'expliquons dans le paragraphe qui suit.

Multiplication par p

Pour $p = 2$ et une courbe canonique $Y^2 + XY = X^3 + a$, nous obtenons aisément, par exemple à partir des formules d'addition sur la courbe,

$$F_2(X) = X + \frac{a}{X} \text{ et } G_2(X,Y) = Y + \frac{a}{X^2} + \tilde{a}\left(1 + \frac{Y}{X^2} + \frac{1}{X}\right).$$

De même, pour $p = 3$ et une courbe canonique $Y^2 = X^3 + a_2X^2 + a_6$, nous avons

$$F_3(X) = \frac{X^3 + 2\,X\tilde{a}_2^3\tilde{a}_6 + \tilde{a}_6^3}{\left(\tilde{a}_2\,X + \tilde{a}_2\,\tilde{a}_6\right)^2},$$

et

$$G_3(X,Y) = Y\frac{X^3 + X\tilde{a}_2^3\tilde{a}_6 + \tilde{a}_6^3 + 2\tilde{a}_2^3\tilde{a}_6^2}{\left(\tilde{a}_2\,X + \tilde{a}_2\,\tilde{a}_6\right)^3}.$$

Pour $p \geqslant 5$ et une courbe d'équation canonique $E : Y^2 = X^3 + a_4X + a_6$, nous appliquons tout d'abord le théorème 42. Nous avons ainsi l'abscisse X_P d'un point P de p-torsion en fonction de $\gamma = \sqrt[p-1]{H_E}$. Il suffit alors de calculer une racine carrée de $X_P^3 + a_4X_P + a_6$ pour obtenir l'ordonnée Y_P de P. Nous calculons ensuite les multiples de P dans $E\left(\mathbb{F}_q(\gamma)\right)$, c'est-à-dire $iP = (X_{iP}, Y_{iP})$ avec $i = 2, \ldots, (p-1)/2$ pour finalement obtenir $h_p(X)$ par

$$h_p(X) = \prod_{i=2}^{(p-1)/2} (X - X_{iP}).$$

Pour disposer au besoin des fractions $F_p(X)$ et $G_p(X,Y)$, il suffit alors d'appliquer les formules de Vélu à partir de $h_p(X)$ comme expliqué au chapitre 4.

Exemple : calculons le polynôme $h_5(X)$ dont les racines sont les abscisses des points de 5-torsion de la courbe $E : Y^2 = X^3 + X + t$ définie sur $\mathbb{F}_{5^3} = \mathbb{F}_5[t]/(t^3 + t + 1)$. Comme

$$(X^3 + X + t)^2 = X^6 + 2\,X^4 + 2\,tX^3 + X^2 + 2\,tX + t^2,$$

nous avons $H_E = 2$. Comme $X^4 - H_E$ est irréductible dans $\mathbb{F}_{5^3}$, posons $\mathbb{K} = \mathbb{F}_{5^3}[\gamma]/(\gamma^4 - 2)$. Nous avons alors dans $\mathbb{K}$,

$$\Delta_0 = \begin{vmatrix} 3 & 2\,t \\ 4 & 4\,\gamma^2 \end{vmatrix} = 2\,\gamma^2 + 2\,t \text{ et } \Delta_1 = \left|4\gamma^2\right| = 4\gamma^2.$$

D'où, $x = \left(3\,t^2 + t + 1\right)\gamma^2 + 4\,t^2 + 2\,t + 1$, et le point

$$P = \left(x, \left(t^2 + 2\,t + 2\right)\gamma^3 + \left(3\,t^2 + 4\,t + 4\right)\gamma\right)$$

est un point de 5-torsion. Donc

$$2P = \left(\left(2\,t^2 + 4\,t + 4\right)\gamma^2 + 4\,t^2 + 2\,t + 1, \left(3\,t^2 + t + 1\right)\gamma^3 + \left(t^2 + 3\,t + 3\right)\gamma\right),$$

et en conséquence,

$$h_5(X) = (X - X_P)(X - X_{2P}) = X^2 + \left(2\,t^2 + t + 3\right)X + 3\,t.$$

5.1.2 Groupe formel

Dans ce qui suit, nous nous référons essentiellement au chapitre IV du livre de Silverman [111][2].

[2] Quelques erreurs et omissions sont à déplorer dans [50, Chap. 12].

Définition

Soit

$$E \ : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

une courbe elliptique définie sur $\mathbb{F}_q$ et posons $T = -X/Y$, $S = -1/Y$ pour envoyer le point à l'infini O_E de E en $(0,0)$. Nous obtenons alors

$$\mathcal{E} \ : \mathcal{F}(T,S) = T^3 + a_1TS + a_2T^2S + a_3S^2 + a_4TS^2 + a_6S^3 - S = 0$$

et l'idée est, même si toute notion de convergence est ici exclue, de pouvoir considérer le comportement des points de $\mathcal{E}$ "au voisinage" de l'élément neutre $O_\mathcal{E} = (0,0)$. Ainsi, nous obtenons S comme une série formelle en T que nous noterons $\mathcal{S}_\mathcal{E}$. Plus généralement, pour toute série formelle T de $\mathbb{F}_q[[\tau]]$ de valuation strictement positive, nous pouvons calculer une série formelle $S \in \mathbb{F}_q[[\tau]]$ telle que $\mathcal{F}(T(\tau), S(\tau)) = 0$. De tels couples $(T(\tau), S(\tau))$ sont appelés des *points formels*. Comme $\mathcal{E}$ est aussi une cubique, l'ensemble des points formels forment un groupe dont nous donnons les formules de la loi d'addition dans le paragraphe suivant. Remarquons de plus qu'un point est complètement déterminé par son abscisse $T(\tau)$ puisque son ordonnée est systématiquement $\mathcal{S}_\mathcal{E}(T(\tau))$.

À partir de l'équation $\mathcal{F}(T, \mathcal{S}_\mathcal{E}) = 0$, il est facile de calculer les L premiers coefficients de $\mathcal{S}_\mathcal{E}$ en une série formelle de T par récurrence avec $O(L^2)$ opérations. Plus généralement pour une série formelle $T(\tau)$, nous calculons $\mathcal{S}_\mathcal{E}(T(\tau))$ par le même procédé. Ainsi

$$\mathcal{S}_\mathcal{E}(T) = \sum_{i=3}^{\infty} s_i T^i = T^3 + a_1\,T^4 + (a_1^2 + a_2)T^5 + O(T^6), \tag{5.2}$$

dont nous déduisons

$$\begin{aligned} Y &= -\frac{1}{S} = -T^{-3} + a_1T^{-2} + a_2T^{-1} + a_3 + (a_1a_3 + a_4)T + O(T^2), \\ X &= \frac{T}{S} = -TY = T^{-2} - a_1T^{-1} - a_2 - a_3T - (a_1a_3 + a_4)T^2 + O(T^3), \\ Z &= \frac{1}{X} = T^2 + a_1T^3 + (a_1^2 + a_2)T^4 + O(T^5). \end{aligned}$$

En utilisant des techniques couramment utilisées en combinatoire [29, 99], nous pouvons néanmoins faire mieux. En particulier $\mathcal{S}_\mathcal{E}$ vérifie une équation différentielle linéaire du second ordre à coefficients polynomiaux en T à partir de laquelle il est facile de déduire des relations de récurrence entre les coefficients de $\mathcal{S}_\mathcal{E}$, ce qui nous permet de calculer $\mathcal{S}_\mathcal{E}$ en $O(L)$ opérations au prix de quelques précalculs.

Le cas $p = 2$: pour une courbe canonique $Y^2 + XY = X^3 + a$ définie sur $\mathbb{F}_{2^n}$, nous avons après le changement de variable $T = -X/Y$, $S = -1/Y$ l'équation de $\mathcal{E}$ qui devient

$$S = T^3 + TS + aS^3. \tag{5.3}$$

Nous obtenons alors la série $\mathcal{S}_\mathcal{E}(\tau) = \sum_{i=1}^{\infty} s_i\tau^i$ du point formel $(\tau, \mathcal{S}_\mathcal{E}(\tau))$ par

$$s_{2i+1} = \begin{cases} 1 & \text{si } i = 1,2,3,5, \\ 1 + a & \text{si } i = 4, \\ s_{2i} + a\left(s_{2i-5} + s_{i-1}^2 + \sum_{j=4}^{i-2} s_j^2 s_{2i-2j+1}\right) & \text{sinon}\,; \end{cases} \tag{5.4}$$

et

$$s_{2i} = \begin{cases} 1 & \text{si } i = 1,2,3,4,5, \\ s_{2i-1} + a\left(s_{2i-6} + \sum_{j=4}^{i-2} s_j^2 s_{2i-2j}\right) & \text{sinon.} \end{cases} \tag{5.5}$$

Cependant, nous pouvons faire mieux à partir de l'équation différentielle satisfaite par $\mathcal{S}_\mathcal{E}$ sur $\mathbb{Z}[a]$,

$$\begin{aligned} \left(-243\,a^3T^9 + 486\,a^3T^8 - 36\,a^2T^6 + 180\,a^2T^5 - 324\,a^2T^4 + 252\,a^2T^3 - 72\,a^2T^2\right)y''(T) + \\ \left(486\,a^3T^7 + 54\,a^2T^5 - 252\,a^2T^4 + 486\,a^2T^3 - 432\,a^2T^2 + 144\,a^2T\right)y'(T) + \\ \left(-486\,a^3T^6 - 36\,a^2T^4 + 126\,a^2T^3 - 162\,a^2T^2 + 72\,a^2T\right)y(T) = 0, \end{aligned}$$

à partir de laquelle nous trouvons, à l'aide du logiciel `GFUN` [99], que les coefficients s_i satisfont la relation de récurrence

$$\begin{aligned}\left(-243\,a^3i^2+3645\,a^3i-13608\,a^3\right)s_{i-7}+\left(486\,a^3i^2-5832\,a^3i+17010\,a^3\right)s_{i-6}+\\ \left(-36\,a^2i^2+378\,a^2i-972\,a^2\right)s_{i-4}+\left(180\,a^2i^2-1512\,a^2i+3042\,a^2\right)s_{i-3}+\\ \left(-324\,a^2i^2+2106\,a^2i-3078\,a^2\right)s_{i-2}+\left(252\,a^2i^2-1188\,a^2i+1008\,a^2\right)s_{i-1}+\\ \left(-72\,a^2i^2+216\,a^2i\right)s_i=0.\end{aligned}$$

Il ne reste plus qu'à calculer s_i sur $\mathbb{Z}[a]$ à partir de ses valeurs initiales

$$s_3=s_4=s_5=s_6=s_7=s_8=1,\ \ s_9=1+a$$

et à réduire ensuite modulo 2.

Loi de groupe

Puisque l'équation de $\mathcal{E}$ est encore celle d'une cubique, nous pouvons définir une loi d'addition sur $\mathcal{E}$ que nous notons $\oplus$ et dont l'élément neutre est $O_{\mathcal{E}}=(0,0)$. Pour la déterminer nous partons de deux points $P_1=(T_1,S_1)$ et $P_2=(T_2,S_2)$ différents de $O_{\mathcal{E}}$ et nous voulons en calculer la somme $P_3=(T_3,S_3)=(T_1,S_1)\oplus(T_2,S_2)$. Géométriquement, cela correspond à tracer une droite passant par P_1 et P_2 (la tangente à la courbe si $P_1=P_2$). Cette dernière coupe alors $\mathcal{E}$ en un troisième point $P_i=(T_i,S_i)$ et la droite passant par P_i et $O_{\mathcal{E}}$ coupe la courbe en P_3.

Précisément, soit $S=\lambda T+\nu$ la droite passant par P_1 et P_2. Si $(T_1,S_1)\neq(T_2,S_2)$, alors

$$\lambda=\frac{S_2-S_1}{T_2-T_1}=T_1^2+T_1T_2+T_2^2+\cdots, \tag{5.6}$$

sinon

$$\lambda=-\frac{\dfrac{\partial\mathcal{F}}{\partial T}}{\dfrac{\partial\mathcal{F}}{\partial S}}=-\frac{a_1\,S_1+3\,T_1^2+2\,a_2\,T_1S_1+a_4\,S_1^2}{-1+a_1\,T_1+2\,a_3\,S_1+a_2\,T_1^2+2\,a_4\,T_1S_1+3\,a_6\,S_1^2}=3T_1^2+4a_1T_1^3+O(T_1^4). \tag{5.7}$$

Dans tous les cas, nous avons

$$\nu=S_1-\lambda T_1.$$

Comme (T_i,S_i) est le troisième point d'intersection de cette droite avec $\mathcal{E}$, T_i vérifie

$$\mathcal{F}(T_i,\lambda T_i+\nu)=0$$

ou

$$\left(a_4\,\lambda^2+a_2\,\lambda+1+a_6\,\lambda^3\right)T^3+\left(a_1\,\lambda+2\,a_4\,\nu\,\lambda+a_2\,\nu+3\,a_6\,\nu\,\lambda^2+a_3\,\lambda^2\right)T^2+\cdots=0.$$

D'où

$$T_1+T_2+T_i=-\frac{a_1\,\lambda+2\,a_4\,\nu\,\lambda+a_2\,\nu+3\,a_6\,\nu\,\lambda^2+a_3\,\lambda^2}{a_4\,\lambda^2+a_2\,\lambda+1+a_6\,\lambda^3}, \tag{5.8}$$

et $S_i=\lambda T_i+\nu$.

Si $T_i=0$, alors P_2 est l'opposé de P_1 et nous avons immédiatement $P_3=O_{\mathcal{E}}$. Sinon, nous devons calculer l'opposé de (T_i,S_i) pour obtenir (T_3,S_3). L'équation de la droite passant par P_i et $O_{\mathcal{E}}$ est $\lambda_i=S_i/T_i$ et $\nu_i=0$. Avec (5.8), nous obtenons donc

$$T_3+0+T_i=-\frac{S_i\ (a_1\,T_i+a_3\,S_i)\,T_i}{a_4\,{S_i}^2T_i+a_2\,S_i\,{T_i}^2+{T_i}^3+a_6\,{S_i}^3}.$$

Comme $\mathcal{F}(T_i,S_i)=0$, le dénominateur se simplifie en

$$S_i-a_1T_iS_i-a_3S_i^2,$$

et nous avons finalement

$$\left\{\begin{array}{rcl} T_3 & = & \dfrac{T_i}{-1 + a_1\, T_i + a_3\, S_i}. \\ S_3 & = & \dfrac{S_i}{T_i} T_3. \end{array}\right. \tag{5.9}$$

Les premiers termes de T_3 sont donc

$$T_1 \oplus T_2 = T_1 + T_2 - a_1 T_1 T_2 - a_2(T_1^2 T_2 + T_1 T_2^2) - (2a_3 T_1^3 T_2 - (a_1 a_2 - 3a_3) T_1^2 T_2^2 + 2a_3 T_1 T_2^3) + \cdots \tag{5.10}$$

et pour T_1=T_2, nous avons de même

$$[2]_{\mathcal{E}} \circ T_1 = 2\, T_1 - a_1\, T_1^2 - 2\, a_2\, T_1^3 + (a_1\, a_2 - 7\, a_3)\, T_1^4 + \cdots . \tag{5.11}$$

Notons enfin qu'il est maintenant facile de calculer l'opposé $-P = (T', S')$ d'un point P par

$$\left\{\begin{array}{rcl} T' & = & \dfrac{T}{-1 + a_1 T + a_3 S}. \\ S' & = & \dfrac{S}{T} T'. \end{array}\right. \tag{5.12}$$

Le cas $p = 2$: pour le cas particulier d'une courbe canonique $Y^2 + XY = X^3 + a$ définie sur $\mathbb{F}_{2^n}$, nous avons ordonné ces calculs pour en diminuer les coûts. Ainsi, si $T_1 \neq T_2$, $(T_1 \oplus T_2(\tau), \mathcal{S}_{\mathcal{E}}(T_1 \oplus T_2(\tau))) = (T_1(\tau), S_1(\tau)) \oplus (T_2(\tau), S_2(\tau))$ est calculé par

$$\left\{\begin{array}{l} \lambda(\tau) = \dfrac{S_1(\tau) + S_2(\tau)}{T_1(\tau) + T_2(\tau)},\ \nu(\tau) = S_1(\tau) + \lambda(\tau) T_1(\tau), \\ \\ T_1 \oplus T_2(\tau) = \dfrac{T_1(\tau) + T_2(\tau) + \lambda(\tau) + a_6 \lambda^2(\tau)(S_1(\tau) + S_2(\tau) + \nu(\tau))}{1 + T_1(\tau) + T_2(\tau) + \lambda(\tau) + a_6 \lambda^2(\tau)\,(S_1(\tau) + S_2(\tau) + \nu(\tau) + \lambda(\tau))}, \\ \\ \mathcal{S}_{\mathcal{E}}(T_1(\tau) \oplus T_2(\tau)) = \nu(\tau) + (\lambda(\tau) + \nu(\tau))(T_1 \oplus T_2(\tau)). \end{array}\right.$$

En stockant les séries intermédiaires nécessaires à ce calcul, nous n'avons donc besoin que de quatre multiplications et deux divisions de séries.

De même, pour $([2]_{\mathcal{E}} T(\tau), \mathcal{S}_{\mathcal{E}}([2]_{\mathcal{E}} T(\tau))) = [2]_{\mathcal{E}}(T(\tau), S(\tau))$, nous utilisons

$$\left\{\begin{array}{l} \delta(\tau) = T^3(\tau) + (1 + \tilde{a}_6 S(\tau))\, T^2(\tau) + \left(\tilde{a}_6 S^2(\tau) + S(\tau)\right) T(\tau) + \tilde{a}_6 S^2(\tau) + S(\tau), \\ \kappa(T) = \dfrac{T^3(\tau) + \tilde{a}_6 S^2(\tau) T(\tau)}{\delta(\tau)}, \\ [2]_{\mathcal{E}} T(\tau) = \kappa^2(\tau), \\ \mathcal{S}_{\mathcal{E}}([2]_{\mathcal{E}} T(\tau)) = \left(\dfrac{\delta(\tau)}{S(\tau) T(\tau)^2}\right)^2. \end{array}\right. \tag{5.13}$$

Le coût est donc aussi de quatre multiplications et deux divisions de séries.

Multiplication par p

La multiplication par tout nombre premier P dans le groupe formel $\mathcal{E}$ satisfait la relation suivante [111, pp. 120].

Théorème 43. *Pour tout nombre premier P, nous avons*

$$[P]_{\mathcal{E}}(T) = P f(T) + g(T^P)$$

où f et g sont des éléments de $\mathbb{F}_{p^n}[[T]]$.

Pour le cas particulier $P = p$,

$$[p]_{\mathcal{E}}(T) = \kappa_{\mathcal{E}}(T)^p = H_E T^p + O(T^{2p}). \tag{5.14}$$

où $\kappa_{\mathcal{E}} \in \mathbb{F}_{p^n}[[T]]$ avec $\kappa_{\mathcal{E}}(T) = \tilde{c}_E T + O(T^2)$ et H_E est l'invariant de Hasse de la courbe E défini au chapitre 2.

Pour accélérer l'algorithme de Couveignes, nous aurons besoin de calculer efficacement $[p]_{\mathcal{E}}$. Bien entendu, nous pouvons utiliser les formules d'addition dans le groupe formel, mais en utilisant la proposition 14, nous avons une façon élégante d'obtenir directement $\kappa_{\mathcal{E}}(T)$.

Proposition 15. *Il existe une fraction rationnelle $\mathcal{R}_E(T, S)$ telle que*

$$\kappa_{\mathcal{E}}(T) = \mathcal{R}_E(T, S(T)).$$

Démonstration. Il suffit d'écrire

$$[p]_{\mathcal{E}}(T) = -\left(\frac{F_p(T/S(T))}{G_p(T/S(T), -1/S(T))}\right)^p,$$

et donc

$$\mathcal{R}_E(T, S) = (-1)^p \frac{F_p(T/S)}{G_p(T/S, -1/S)}. \tag{5.15}$$

□

Exemple : pour $p = 2$ et une courbe canonique $E : Y^2 + XY = X^3 + a$ nous avons ainsi

$$\kappa_{\mathcal{E}}(T) = \frac{\left(T^2 + \tilde{a}S(T)^2\right) T}{T^3 + (1 + \tilde{a}S(T))\, T^2 + \left(\tilde{a}S(T)^2 + S(T)\right) T + \tilde{a}S(T)^2 + S(T)}.$$

5.2 Description théorique de l'algorithme

Soient E_a et E_b deux courbes elliptiques définies sur $\mathbb{F}_q$ par

$$\begin{array}{rrcl} E_a: & Y^2 + a_1XY + a_3Y & = & X^3 + a_2X^2 + a_4X + a_6, \\ E_b: & Y^2 + b_1XY + b_3Y & = & X^3 + b_2X^2 + b_4X + b_6, \end{array}$$

telles qu'il existe une isogénie $\mathcal{I}$ de degré ℓ entre elles donnée par

$$\begin{array}{rccc} \mathcal{I}: & E_a & \longrightarrow & E_b \\ & (X, Y) & \longmapsto & \left(\dfrac{g(X)}{h^2(X)}, \dfrac{l(X) + Yk(X)}{h^3(X)}\right), \end{array}$$

où $g(X)$, $h(X)$, $l(X)$ et $k(X)$ sont des polynômes de degrés respectifs au plus ℓ, $(\ell-1)/2$, $3(\ell+1)/2$ et $3(\ell-1)/2$. L'objectif de l'algorithme de Couveignes [30] est le calcul de $g(X)$ et $h(X)$. Nous nous préoccupons de l'abscisse I de $\mathcal{I}$ seulement. Cela équivaut à chercher g et h tels que

$$I : X \mapsto I(X) = \frac{g(X)}{h(X)^2}$$

ou encore $\hat{I}$ qui à $Z = 1/X$ associe $\hat{I}(Z)$ défini par

$$\hat{I} : Z \mapsto \hat{I}(Z) = Z\frac{\hat{h}^2(Z)}{\hat{g}(Z)}$$

avec $\hat{g}(Z) = Z^\ell g(1/Z)$ et $\hat{h}(Z) = Z^{(\ell-1)/2} h(Z)$. Il est bien connu que les coefficients du développement en série d'une fraction rationnelle $F(Z)$ avec un dénominateur de degré ℓ satisfont une relation de récurrence

de degré ℓ [56]. Réciproquement, étant donné les 2ℓ premiers coefficients, nous pouvons retrouver $F(Z)$. L'idée de Couveignes consiste à trouver une série semblable à celle provenant d'une isogénie et ensuite regarder s'il s'agit réellement d'une isogénie, c'est-à-dire s'il s'agit du développement d'une fraction rationnelle dont le dénominateur a degré ℓ. En fait, nous avons à en calculer $2\ell+2$ termes pour obtenir une fraction avec un dénominateur de degré *a priori* $\ell+1$. Si ce dénominateur se trouve avoir degré ℓ, nous sommes alors presque certain d'avoir la bonne isogénie.

L'énumération des isogénies possibles est rendue possible grâce à l'utilisation des groupes formels associés à E_a et E_b comme décrit ci-dessous.

5.2.1 Morphismes de groupe formel

Comme expliqué dans la section 5.1, deux groupes formels

$$\begin{array}{rrcl} & \mathcal{E}_a : & T^3 + a_1TS + a_2T^2S + a_3S^2 + a_4TS^2 + a_6S^3 - S & = & 0 \\ \text{et} & \mathcal{E}_b : & T^3 + b_1TS + b_2T^2S + b_3S^2 + b_4TS^2 + b_6S^3 - S & = & 0 \end{array}$$

sont associés à E_a et E_b. Un *morphisme* $\mathcal{M}$ de $\mathcal{E}_a$ vers $\mathcal{E}_b$ vérifie alors pour tout point $(T_1(\tau), S_1(\tau))$ et $(T_2(\tau), S_2(\tau))$ de $\mathcal{E}_a$

$$\mathcal{M}((T_1(\tau), S_1(\tau)) \oplus (T_2(\tau), S_2(\tau))) = \mathcal{M}((T_1(\tau), S_1(\tau))) \oplus \mathcal{M}((T_2(\tau), S_2(\tau))).$$

Associée à un morphisme $\mathcal{M}$, il existe une série

$$\mathcal{U}(T) = \sum_{i \geqslant 1} u_i T^i,$$

telle qu'un point $(T(\tau), S(\tau))$ de $\mathcal{E}_a$ est envoyé sur le point $\mathcal{M}(T(\tau), S(\tau))$ de $\mathcal{E}_b$ égal à $(\mathcal{U}(T(\tau)), \mathcal{S}_{\mathcal{E}_b}(\mathcal{U}(T(\tau))))$. *A fortiori*, les séries $\mathcal{U}(T)$ satisfont

$$\mathcal{U}(T_1 \oplus T_2(\tau)) = \mathcal{U}(T_1(\tau)) \oplus \mathcal{U}(T_2(\tau)) \tag{5.16}$$

ou encore $\mathcal{U} \circ [n]_{\mathcal{E}_a} = [n]_{\mathcal{E}_b} \circ \mathcal{U}$ pour tout entier n. Nous savons de plus que l'ensemble des morphismes de $\mathcal{E}_a$ vers $\mathcal{E}_b$ est un $\mathbb{Z}_p$-module de rang 1 (cf. [43]).

Pour en revenir à notre problème, $\hat{I}$ donne lieu à un morphisme $\mathfrak{I}$ de $\mathcal{E}_a$ vers $\mathcal{E}_b$, et donc à une série $\mathcal{W}$. Notre problème est maintenant le suivant : parmi tous les morphismes de $\mathcal{E}_a$ vers $\mathcal{E}_b$, comment déterminer celui qui provient de $\mathfrak{I}$, ou alternativement parmi les séries satisfaisant l'équation (5.16), comment déterminer celle qui est égale à $\mathcal{W}$.

Puisque $2\ell+2$ termes de $\hat{I}(Z)$ sont nécessaires à la détermination de I, et puisque $Z = 1/X = S/T = T^2 + O(T^3)$, cela signifie que $\mathcal{L} = 4\ell+2$ termes de la série $\mathcal{W}$ sont aussi nécessaires. De plus, nous devons considérer un nombre fini de séries afin de trouver la bonne. Nous calculerons le nombre exact de ces séries dans la section suivante.

5.2.2 Conditions nécessaires vérifiées par un morphisme

Exploitons maintenant les propriétés satisfaites par des morphismes de $\mathcal{E}_a$ vers $\mathcal{E}_b$ et les implications sur leurs séries associées. Nous allons calculer par récurrence les $\mathcal{L}$ premiers coefficients de la série

$$\mathcal{U}(T) = \sum_{i=1}^{\infty} u_i T^i.$$

Plus précisément, en supposant $u_1, \ldots, u_{i-1}$ connus, une exploitation ingénieuse de l'équation (5.16) nous permet de calculer u_i.

Spécialisons l'équation (5.16) avec $T_1(\tau) = \tau$ et $T_2(\tau) = A\tau$ où $A \in \mathbb{F}_q$. Ainsi

$$\mathcal{U}(\tau \oplus A\tau) = \mathcal{U}(\tau) \oplus \mathcal{U}(A\tau). \tag{5.17}$$

Extrayons les coefficients de τ^i dans l'équation (5.17). Nous savons par (5.10) que

$$\tau \oplus A\tau = (1+A)\tau + O(\tau^2)$$

et que donc

$$\mathcal{U}(\tau \oplus A\tau) = \sum_{k=1}^{\infty} u_k((1+A)\tau + O(\tau^2))^k$$

où u_i apparaît seul dans le coefficient de τ^i en $(1+A)^i u_i$ parmi des termes dépendants seulement de u_1, u_2, …, u_{i-1}. D'autre part,

$$\mathcal{U}(\tau) \oplus \mathcal{U}(A\tau) = \mathcal{U}(\tau) + \mathcal{U}(A\tau) + P(\mathcal{U}(\tau), \mathcal{U}(A\tau)),$$

où $P(\mathcal{U}(\tau), \mathcal{U}(A\tau))$ contient des monômes de degré total supérieur à un en $\mathcal{U}(\tau)$ et $\mathcal{U}(A\tau)$. Cela signifie que u_i apparaît dans le coefficient de τ^i en $(1+A^i)u_i$ parmi des termes dépendants seulement de u_1, u_2, …, u_{i-1}. De cela nous déduisons que

$$u_i \left((1+A)^i - 1 - A^i\right) + e_i(A, u_1, \ldots, u_{i-1}) = 0, \tag{5.18}$$

où e_i est un polynôme multivarié. Si $(1+A)^i \neq 1 + A^i$, cette relation nous fournit u_i. Cette condition n'est évidemment pas vérifiée quand i est une puissance de p, mais pour tout autre valeur de i, nous pouvons trouver A tel que ceci soit réalisé, au moins si $i < q$.

Supposons maintenant que $i = p^e$. Nous écrirons $\kappa_a(T)$ (resp. $\kappa_b(T)$) pour $\kappa_{\mathcal{E}_a}(T)$ (resp. $\kappa_{\mathcal{E}_b}(T)$). De même, nous posons $[p]_a = [p]_{\mathcal{E}_a}$, $[p]_b = [p]_{\mathcal{E}_b}$, $\mathcal{R}_a(T,S) = \mathcal{R}_{E_a}(T,S)$, $\mathcal{R}_b(T,S) = \mathcal{R}_{E_b}(T,S)$ et enfin $c = H_{E_a} = H_{E_b}$ puisque E_a et E_b ont par construction un invariant de Hasse identique (cf. chapitre 3). Tirons maintenant parti de

$$\mathcal{U}([p]_a\tau) = [p]_b(\mathcal{U}(\tau)),$$

obtenu de l'équation (5.16) et écrivons $\tilde{\mathcal{U}}(T) = \sum_{k=1}^{\infty} \tilde{u}_k T^k$. En utilisant le théorème 43, nous déduisons que

$$\tilde{\mathcal{U}}(\tau) \circ \kappa_a(\tau) = \kappa_b(\tau) \circ \mathcal{U}(\tau). \tag{5.19}$$

Ici, l'égalité des coefficients de X^i conduit à une équation non triviale de degré p en $\tilde{u}_i$,

$$c^{p^{e-1}} \tilde{u}_i - \tilde{c} u_i = f_i(\tilde{u}_1, \ldots, \tilde{u}_{i-1}), \tag{5.20}$$

avec f_i, un polynôme multivarié. Nous posons donc

$$\eta = c^{(p^e-1)/(p-1)}$$

et nous réécrivons (5.20) en

$$\left(\frac{u_i}{\eta}\right) - \left(\frac{u_i}{\eta}\right)^p = f_i'. \tag{5.21}$$

Nous verrons plus loin comment résoudre cette équation. A priori, elle a au plus p solutions.

Pour initialiser les récurrences précédentes, regardons le cas $i = 1 = p^0$. Les équations correspondantes sont

$$c u_1^p = u_1 c,$$

et donc $u_1 \in \mathbb{F}_p$. Comme $\mathcal{U}$ doit être de valuation 1, nous avons en fait $u_1 \in \mathbb{F}_p^*$.

Remarque :

Lors du calcul de la cardinalité de $E_a(\mathbb{F}_q)$ avec l'algorithme SEA, nous trouvons les courbes isogènes par leurs invariants j_{E_b} qui sont racines d'une équation modulaire. Il peut arriver qu'il y ait plusieurs racines mais que seulement l'une d'entre elles soit celle de l'invariant de E_b. Le procédé décrit pour trouver l'isogénie peut détecter rapidement des courbes E_b non isogènes, car dans ce cas l'une des équations (5.20) n'a pas de solution.

5.2.3 Énumérations

Nous pouvons résumer les résultats de la section précédente ainsi. Une fois u_{p^e} fixé, tous les coefficients u_j pour $p^e < j < p^{e+1}$ sont déterminés de façon unique. Pour des séries tronquées jusqu'à $\mathcal{L}$ avec $p^r < \mathcal{L} < p^{r+1}$, il existe au plus p^{r+1} séries distinctes. Pour chaque e, $1 \leq e \leq r$, il y a au plus p valeurs pour u_{p^e}. Si $e = 0$, ce nombre est au plus $p - 1$ puisque $u_1 = 0$ n'est pas valide. Il y a donc au plus $p^r(p-1)$ morphismes $\mathcal{U}$ à énumérer pour trouver celui qui coïncide avec une isogénie.

Première approche, "le backtracking"

Elle consiste simplement à essayer toutes les valeurs possibles de u_{p^e} pour chaque e en utilisant une procédure du type "backtracking". Ce qui découle immédiatement des explications précédentes (section 5.2.2).

Deuxième approche, "la multiplication p-adique"

Nous pouvons tirer parti du fait que l'ensemble des morphismes de $\mathcal{E}_a$ vers $\mathcal{E}_b$ est un $\mathbb{Z}_p$-module de dimension 1. Ainsi, un morphisme $\mathcal{U}$ non trivial comme celui de la section 5.2.2 en est un générateur et pour tout morphisme associé à une série $\mathcal{W}$, il existe un entier p-adique N tel que $\mathcal{W} = [N]_b \circ \mathcal{U}$. Écrivons

$$N = \sum_{i=0}^{\infty} n_i p^i.$$

En se rappelant que $p^r < \mathcal{L} < p^{r+1}$, nous avons

$$[N]_b \circ \mathcal{U} = \bigoplus_{i=0}^{r} \left([n_i]_b \circ ([p^i]_b \circ \mathcal{U})\right) \oplus \bigoplus_{i>r} \left([n_i]_b \circ ([p^i]_b \circ \mathcal{U})\right).$$

Mais la valuation de la série $[p^i]_b(T)$ est p^i, ce qui implique que, quand $i > r$, les termes venant de $[p^i]_b \circ \mathcal{U}$ n'apportent aucune contribution aux $\mathcal{L}$ premiers coefficients de $[N]_b \circ \mathcal{U}$. Il est donc suffisant de vérifier que l'une des séries $[N]_b \circ \mathcal{U}$ provient d'une isogénie pour $N < p^{r+1}$. Nous pouvons réduire le nombre de tentatives en utilisant, outre le fait que n_0 ne peut pas être nul, le résultat suivant.

Proposition 16. *Soient k, un entier strictement positif et N, un entier satisfaisant $0 \leq N < p^k$, nous avons alors dans un groupe formel $\mathcal{E}$ associé à une courbe elliptique E,*

$$[p^k - N]_{\mathcal{E}}(T) = -[N]_{\mathcal{E}}(T) + O(T^{p^k}). \tag{5.22}$$

Démonstration. Le résultat suit aisément de (5.14) puisque

$$[p^k - N]_{\mathcal{E}}(T) \oplus [N]_{\mathcal{E}}(T) = [p^k]_{\mathcal{E}}(T) = H_E^{p^{k-1}} T^{p^k} + O(T^{2p^k}).$$

□

Nous en déduisons que

$$([p^{r+1} - N]_b \circ \mathcal{U})(T) = ([-N]_b \circ \mathcal{U})(T) + O(T^{p^{r+1}}).$$

D'autre part, les séries $\mathcal{W}$ et $-\mathcal{W}$ mènent à la même isogénie $\hat{I}$. Donc, au moins un morphisme $[N]_b \circ \mathcal{U}$, pour $N < p^{r+1}/2$ et N premier avec p, est égal à $\mathcal{W}$ ou $-\mathcal{W}$ et est associé à $\hat{I}$. C'est-à-dire que nous avons à calculer au plus $p^r(p-1)/2$ morphismes $\mathcal{M}$. Cela revient à considérer dans la première approche seulement la moitié des u_1 potentiels, d'où $u_1 = 1, \ldots, (p-1)/2$ si p est impair et seulement $u_1 = 1$ si $p = 2$.

Comme point final de cette section, nous avons de plus :

Proposition 17. *Quand $p = 2$,*

$$\mathcal{U}(T) \oplus \mathcal{U}\left(\frac{T}{1+T}\right) = 0,$$

et quand p est impair,

$$\mathcal{U}(-T) = -\mathcal{U}(T).$$

Démonstration. Il s'agit d'une simple application de l'équation (5.12). □

5.3 Mise en œuvre pratique de l'algorithme

5.3.1 Calculs incrémentaux avec des séries tronquées

L'optimisation de l'algorithme de Couveignes nécessite l'utilisation d'algorithmes rapides pour manipuler des séries tronquées. Au travers de ce qui a été décrit dans la section 5.2, les algorithmes dont nous avons besoin nécessitent des calculs incrémentaux puisque nous trouvons les coefficients de séries, un par un, en utilisant des séries intermédiaires dont certaines sont aussi connues petit à petit.

Après avoir donné des algorithmes incrémentaux pour les quatre opérations usuelles, nous appliquerons ces idées à quelques algorithmes nécessaires dans les groupes formels. Enfin, nous donnons une version incrémentale de l'algorithme de Brent et Kung [15] utilisé pour la composition de séries.

Dans ces sections, nous notons pour toute série $\mathcal{A}(\tau) = \sum_{i \geqslant v}^{\infty} a_i \tau^i$ de valuation v dans $\mathbb{F}_q[[\tau]]$, $\mathcal{A}(\tau)_k$ la somme finie $\sum_{i=v}^{k} a_i \tau^i$. Nous faisons de plus l'hypothèse que la multiplication de deux séries avec m termes a pour complexité asymptotique $O(m^\mu)$ opérations dans $\mathbb{F}_q$ avec $1 < \mu \leq 2$.

Opérations usuelles

Nous rappelons tout d'abord quelques résultats immédiats à propos de l'addition, la multiplication et la division de deux séries $(\mathcal{A}(T), \mathcal{B}(T))$ de $\mathbb{F}_q[[T]]^2$, $\mathcal{C}(T) \in \mathbb{F}_q[[T]]$ étant le résultat de ces opérations. Nous notons

$$\mathcal{A}(T) = \sum_{i=0}^{\infty} a_i T^i,\ \mathcal{B}(T) = \sum_{i=0}^{\infty} b_i T^i \text{ et } \mathcal{C}(T) = \sum_{i=0}^{\infty} c_i T^i.$$

Les démonstrations des propositions 18, 19 et 20 peuvent être trouvées dans [56, chap. 4.7].

Proposition 18. *Soit $\mathcal{C}(T) = \mathcal{A}(T) \pm \mathcal{B}(T)$. Nous obtenons alors pour tout entier L, le coefficient c_L à partir de a_L et b_L par*

$$c_L = a_L \pm b_L$$

avec une addition/soustraction dans $\mathbb{F}_q$.

Proposition 19. *Soit $\mathcal{C}(T) = \mathcal{A}(T)\mathcal{B}(T)$. Nous obtenons alors, pour tout entier L, le coefficient c_L à partir de $(a_0, \ldots, a_L)$ et $(b_0, \ldots, b_L)$ par*

$$c_L = \sum_{i=0}^{L} a_i b_{L-i}$$

avec $L+1$ multiplications dans $\mathbb{F}_q$.

Proposition 20. *Soit $\mathcal{C}(T) = \mathcal{A}(T)/\mathcal{B}(T)$ avec $b_0 \neq 0$, nous obtenons alors pour tout entier L, le coefficient c_L à partir de de a_L, $(b_0, \ldots, b_L)$ et $(c_0, \ldots, c_{L-1})$ par*

$$c_L = \left(a_L - \sum_{i=0}^{L-1} c_i b_{L-i}\right) / b_0$$

avec L multiplications et une division dans $\mathbb{F}_q$.

Points formels

Soit $(\mathcal{V}(T), \mathcal{S}_{\mathcal{E}}(\mathcal{V}(T)))$ un point d'un groupe formel $\mathcal{E}$ associé à une courbe elliptique

$$E \ : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6.$$

Nous notons

$$\mathcal{V}(T) = \sum_{i=1}^{\infty} v_i T^i \text{ et } \mathcal{S}_{\mathcal{E}}(\mathcal{V}(T)) = \sum_{i=3}^{\infty} \varpi_i T^i.$$

Comme expliqué dans la section 5.2, le calcul des coefficients $s_1, \ldots, s_L$ de $\mathcal{S}_{\mathcal{E}}(T)$ peut être réalisé en $O(L^2)$ multiplications. La proposition 21 donne plus de détails à ce sujet.

Proposition 21. *Nous pouvons obtenir ϖ_L de $(\varpi_3, \ldots, \varpi_{L-1})$ et $(v_1, \ldots, v_{L-2})$ avec $O(L)$ multiplications dans $\mathbb{F}_q$.*

Démonstration. Puisque $(\mathcal{V}(T), \mathcal{S}_{\mathcal{E}}(\mathcal{V}(T)))$ est un point de $\mathcal{E}$, nous avons

$$\mathcal{V}^3 + a_1\mathcal{V}\mathcal{S}_{\mathcal{E}}(\mathcal{V}) + a_2\mathcal{V}^2\mathcal{S}_{\mathcal{E}}(\mathcal{V}) + a_3\mathcal{S}_{\mathcal{E}}(\mathcal{V})^2 + a_4\mathcal{V}\mathcal{S}_{\mathcal{E}}(\mathcal{V})^2 + a_6\mathcal{S}_{\mathcal{E}}(\mathcal{V})^3 = \mathcal{S}_{\mathcal{E}}(\mathcal{V}). \tag{5.23}$$

Comme montré dans les propositions 18 et 19 et étant donné que les valuations de $\mathcal{V}(T)$ et $\mathcal{S}_{\mathcal{E}}(\mathcal{V}(T))$ sont respectivement 1 et 3, le $L^{\text{ème}}$ coefficient de $\mathcal{V}^3$ dépend de $(v_1, \ldots, v_{L-2})$, le $L^{\text{ème}}$ coefficient de $\mathcal{V}\mathcal{S}_{\mathcal{E}}(\mathcal{V})$ dépend de $(v_1, \ldots, v_{L-3})$ et de $(\varpi_3, \ldots, \varpi_{L-1})$, le $L^{\text{ème}}$ coefficient de $\mathcal{V}^2\mathcal{S}_{\mathcal{E}}(\mathcal{V})$ dépend de $(v_1, \ldots, v_{L-4})$ et de $(\varpi_3, \ldots, \varpi_{L-2})$, le $L^{\text{ème}}$ coefficient de $\mathcal{S}_{\mathcal{E}}^2(\mathcal{V})$ dépend de $(\varpi_3, \ldots, \varpi_{L-3})$, le $L^{\text{ème}}$ coefficient de $\mathcal{V}\mathcal{S}_{\mathcal{E}}^2(\mathcal{V})$ dépend de $(v_1, \ldots, v_{L-6})$ et de $(\varpi_3, \ldots, \varpi_{L-4})$ et le $L^{\text{ème}}$ coefficient de $\mathcal{S}_{\mathcal{E}}(\mathcal{V})^3$ dépend de $(\varpi_3, \ldots, \varpi_{L-6})$. Nous déduisons de l'équation (5.23) que ϖ_L est un polynôme multivarié en $(v_1, \ldots, v_{L-2})$ et $(\varpi_3, \ldots, \varpi_{L-1})$.

Par ailleurs, nous avons vu que l'obtention du $L^{\text{ème}}$ coefficient d'un produit de séries peut être réalisé avec $O(L)$ multiplications. Le $L^{\text{ème}}$ coefficient de $\mathcal{S}_{\mathcal{E}}(\mathcal{V})$ peut par conséquent lui aussi être calculé avec $O(L)$ multiplications. □

Proposition 22. *Soient $\mathcal{A}(T) = \sum_{i=1}^{\infty} a_i T^i$ et $\mathcal{A}'(T) = \sum_{i=1}^{\infty} a'_i T^i$, deux séries formelles et posons $\mathcal{S}(T) = \mathcal{S}_{\mathcal{E}}(\mathcal{A}(T)) = \sum_{i=3}^{\infty} \varpi_i T^i$ (resp. $\mathcal{S}'(T) = \mathcal{S}_{\mathcal{E}}(\mathcal{A}'(T)) = \sum_{i=3}^{\infty} \varpi'_i T^i$). Nous pouvons obtenir le $L^{\text{ème}}$ coefficient de la série $(\mathcal{A}(T) \oplus \mathcal{A}'(T))_L$ à partir des points formels tronqués $(\mathcal{A}(T)_L, \mathcal{S}_{L+1}(T))$ et $(\mathcal{A}'(T)_L, \mathcal{S}'_{L+1}(T))$ avec $O(L)$ opérations.*

Démonstration. Appliquons les formules de la loi de groupe pour calculer $(\mathcal{A}(T) \oplus \mathcal{A}'(T))_L$. Nous obtenons des valuations des séries que le $L^{\text{ème}}$ coefficient de la série

$$\lambda(T) = \frac{\mathcal{S}(T) - \mathcal{S}'(T)}{\mathcal{A}(T) - \mathcal{A}'(T)},$$

dépend de $(a_1, \ldots, a_{L-1})$ et $(\varpi_3, \ldots, \varpi_{L+1})$ (resp. a'_i et ϖ'_i avec les mêmes indices). D'autre part, la valuation de $\lambda(T)$ est deux et donc le $L^{\text{ème}}$ coefficient de la série

$$\nu(T) = \mathcal{S}(T) - \lambda(T)\mathcal{A}(T)$$

dépend de $(a_1, \ldots, a_{L-2})$ et $(\varpi_3, \ldots, \varpi_L)$. La valuation de $\nu(T)$ est trois. Il suit alors que le $L^{\text{ème}}$ coefficient de la série

$$T_i(T) = -\mathcal{A}(T) - \mathcal{A}'(T) - \frac{a_1\,\lambda(T) + 2\,a_4\,\nu(T)\,\lambda(T) + a_2\,\nu(T) + 3\,a_6\,\nu(T)\,\lambda(T)^2 + a_3\,\lambda(T)^2}{1 + a_4\,\lambda(T)^2 + a_2\,\lambda(T) + a_6\,\lambda(T)^3}$$

dépend de $(a_1, \ldots, a_L)$ et $(\varpi_3, \ldots, \varpi_{L+1})$ (idem avec a'_i et ϖ'_i). La valuation de $T_i(T)$ est deux. Par conséquent, le $L^{\text{ème}}$ coefficient de la série

$$S_i(T) = \lambda(T)T_i(T) + \nu(T),$$

dépend de $a_1, \ldots, a_{L-2}$ et $\varpi_3, \ldots, \varpi_L$ (idem pour a'_i et ϖ'_i). La valuation de $S_i(T)$ est trois. D'où, de

$$\mathcal{A}(T) \oplus \mathcal{A}'(T) = \frac{T_i(T)}{-1 + a_1\, T_i(T) + a_3\, S_i(T)}$$

nous déduisons que le $L^{\text{ème}}$ coefficient de cette série dépend de $(a_1, \ldots, a_L)$ et $(\varpi_3, \ldots, \varpi_{L+1})$ (idem pour a'_i et ϖ'_i). □

Ce dernier résultat peut être prouvé de la même façon à partir des formules de la section 5.2

Proposition 23. *Nous pouvons obtenir le $L^{\text{ème}}$ terme de la série $\mathcal{R}_E(T, \mathcal{S}_{\mathcal{E}}(T)) \circ \mathcal{V}(T)$ à partir du point formel tronqué $(\mathcal{V}(T)_L, \mathcal{S}_{\mathcal{E}}(\mathcal{V}(T))_{L+1})$ avec $O(L)$ opérations.*

Composition incrémentale de série à la Brent et Kung

Soient $f(T) = \sum_{i=1}^{\infty} a_i T^i$ et $g(T) = \sum_{i=0}^{\infty} b_i T^i$, deux séries formelles de $\mathbb{F}_q[[T]]$, nous cherchons à calculer de façon incrémentale la série

$$h(T) = (g \circ f)(T) = \sum_{i=0}^{\infty} c_i T^i.$$

Précisément, nous supposons que f est connue jusqu'à l'ordre L et que nous avons besoin de calculer les coefficients h_0, h_1, …, h_L un par un, ou alternativement, étant donné les séries à l'ordre i, $i < L$, trouver h_i. La façon classique de faire est de précalculer les puissances de f, f^i pour $i = 2, \ldots, L$. Notre version incrémentale de l'algorithme de Brent et Kung [15] permet avec un même coût asymptotique de remplacer le stockage de ces L séries par celui d'environ $2\sqrt{L}$ séries.

Soit B, un entier $\leq L$ que nous déterminerons par la suite. Soit i un entier inférieur à L et supposons connu tous les coefficients de g (resp. h) d'indice $< i$. Nous cherchons alors c_i. Pour calculer $g_i \circ f$, nous écrivons

$$g_i(T) = \sum_{k=0}^{i} b_k T^k = \sum_{0 \leq j \leq i/B} G_j(T) T^{Bj}$$

où $G_j(T)$ est un polynôme de degré au plus $B-1$ en T. Alors

$$g_i \circ f = \sum_{0 \leq j \leq i/B} G_j(f) f^{Bj}.$$

Nous précalculons $f_j = f^j$ pour $0 \leq j \leq B$ et $F_j = f_B^j$ pour $0 \leq j \leq L/B$, jusqu'à l'ordre L. Posons $i = JB + I$ avec $0 \leq I < B$, nous obtenons

$$g_i \circ f = \sum_{0 \leq j < J} G_j(f_1) F_j + \left(\sum_{k=0}^{I-1} b_{JB+k} f_k \right) F_J + b_i f_I F_J = \Sigma_{1,i} + \Sigma_{2,i} F_J + b_i f_I F_J.$$

(Nous utilisons la convention que $\Sigma_{2,i} = 0$ pour $I = 0$.) Il est facile de voir que tous les termes de $\Sigma_{1,i}$ et $\Sigma_{2,i}$ d'ordre $\leqslant L$ (et non pas i) dépendent uniquement des i coefficients $b_1, \ldots, b_{i-1}$. Il est alors immédiat d'obtenir le $i^{\text{ème}}$ terme de $\Sigma_{2,i} F_J$ ou $f_I F_J$ en $O(i)$ opérations, ce qui nous permet de trouver le coefficient désiré c_i.

Une fois cela fait, nous avons à mettre à jour les séries. Notons que nous n'avons pas besoin des termes d'indice $\leq i$. Si $I < B-1$, alors $\Sigma_{1,i+1} = \Sigma_{1,i}$ et

$$\Sigma_{2,i+1} = \Sigma_{2,i} + b_i f_I.$$

Dans ce cas, mettre à jour la série coûte $O(L-i)$. Si $I = B-1$, alors

$$\Sigma_{1,i+1} = \Sigma_{1,i} + (\Sigma_{2,i} + c_i f_I) F_J$$

et $\Sigma_{2,i+1} = 0$. Puisque, nous avons seulement besoin des termes de degré $> i$, cela coûte $O((L-i)^\mu)$.

Precalculer les f_j coûte

$$\sum_{j=2}^{B}(L-j)^\mu,$$

et les F_j,

$$\sum_{j=2}^{L/B}(L-jB)^\mu,$$

pour un stockage de $O(B + L/B)$ séries avec L termes. Le coût des calculs de tous les $\Sigma_{1,i}$ et $\Sigma_{2,i}$ est aussi

$$\sum_{j=2}^{L/B}(L-jB)^\mu.$$

Nous devons donc minimiser

$$\sum_{j=2}^{B}(L-j)^\mu + 2\sum_{j=2}^{L/B}(L-jB)^\mu.$$

Nous approximons ces quantités avec les intégrales correspondantes et nous avons ainsi à minimiser

$$C(B) = (L-2)^{\mu+1} - (L-B)^{\mu+1} + 2(L-2B)^{\mu+1}/B.$$

Nous dérivons alors C par rapport à B et substituons $L\beta$ à B. Nous développons la dérivée comme une fonction en β et trouvons

$$C'(B) = L^{\mu+1}\left(-2 + ((\mu+1)L + 4\mu^2 + 4\mu)\beta^2 + O(\beta^3)\right).$$

Ce qui conduit au choix de

$$\beta = \left(\sqrt{\frac{\mu+1}{2}L}\right)^{-1}$$

ou

$$B = \sqrt{\frac{2}{\mu+1}}L^{1/2}.$$

Ainsi, le coût total de l'algorithme est approximativement $2BL^{\mu+1/2}$ avec un stockage en $O(L^{1/2})$.

Remarque : le rôle de la constante $\sqrt{\frac{2}{\mu+1}}$ n'est pas très important, puisqu'elle appartient à l'intervalle $[0.816, 1]$.

5.3.2 Programmation efficace de l'algorithme

Dans cette section, nous donnons les algorithmes nécessaires à l'implantation efficace des idées de Couveignes, et nous en déduisons la complexité de la méthode. Tout d'abord, nous décrivons les précalculs qui dépendent seulement de p, puis nous rappelons comment calculer les solutions de l'équation $X - X^p = \alpha$ dans $\mathbb{F}_{p^n}$. De plus, nous mettons en lumière quelques techniques utiles au calcul d'une fraction à partir de son développement en série. Nous terminons cette section en analysant les complexités complètes des deux approches précédemment décrites pour énumérer l'ensemble des morphismes.

Précalculs indépendant de p

Deux tels précalculs sont possibles, ceux nécessaires à $[p]_b$ et ceux pour résoudre efficacement des équations du type $X - X^p = \alpha$. Pour calculer efficacement la multiplication par p dans $\mathcal{E}_b$, le calcul de la fraction $\mathcal{R}_b(T, S)$ de la proposition 15 est nécessaire. Le coût en est $O(p^2 \log p)$ opérations élémentaires. Quant à résoudre $X - X^p = \alpha$, nous utilisons le résultat suivant dû à Hilbert (cf. par exemple [76]).

Proposition 24. *L'équation*

$$\beta - \beta^p = \alpha \tag{5.24}$$

a une solution dans $\mathbb{F}_{p^n}$ si et seulement si $\mathrm{Tr}_{\mathbb{F}_{p^n}/\mathbb{F}_p}(\alpha) = 0$. De plus, si θ est un élément de $\mathbb{F}_{p^n}$ de trace 1, alors une solution de cette équation est

$$\beta = \alpha\theta^p + (\alpha + \alpha^p)\theta^{p^2} + \cdots + (\alpha + \alpha^p + \cdots + \alpha^{p^{n-2}})\theta^{p^{n-1}}. \tag{5.25}$$

Remarquons que si (5.24) a une solution β, alors $\beta + k$ est aussi une solution pour tout élément k de $\mathbb{F}_p$. Il est aussi facile de voir que la correspondance $\alpha \mapsto \beta$ est linéaire. Ayant calculé la matrice de cette application, l'équation (5.21) peut être simplement résolue en appliquant cette matrice aux coefficients de cette équation.

Le calcul de cette matrice dépend uniquement de p et n, et non pas de ℓ. Cela signifie qu'il peut être réalisé seulement une fois avant tout calcul d'isogénie, ce coût peut donc être négligé. Notons que nous avons cependant besoin de stocker $O(n^2)$ éléments dans $\mathbb{F}_p$ mais qu'alors, le temps nécessaire pour appliquer cette matrice est $O(1)$ (multiplications dans $\mathbb{F}_q$).

Trouver un morphisme

Le cœur de l'algorithme est le calcul avec les équations (5.17) et (5.19) d'une série $\mathcal{U}$ associée à un morphisme. Nous distinguons deux étapes, une phase de précalculs et une phase principale.

Phase de précalculs : les séries indépendantes de $\mathcal{U}$ sont complètement calculées alors que seuls les premiers termes des autres séries peuvent être initialisés. Nous réalisons notamment quelques précalculs utiles à la composition de série sur la base de $\mathcal{L}$ termes de $\mathcal{U}$.

Précisément nous calculons :

1. $\mathcal{S}_a(\tau)_{\mathcal{L}+1}$ à partir de $(\tau)_{\mathcal{L}}$ avec la proposition 21 ;
2. A tels que $(1 + A)^i \neq 1 + A^i$ pour tout $i \leqslant \mathcal{L}$ (ce qui implique en particulier $q > \mathcal{L}$).
3. $\mathcal{S}_a(A\tau)_{\mathcal{L}+1}$ à partir de $\mathcal{S}_a(\tau)_{\mathcal{L}+1}$.
4. la série

$$((\tau \oplus A\tau(\tau))_{\mathcal{L}}, \mathcal{S}_a(\tau \oplus A\tau(\tau))_{\mathcal{L}+1}) = ((\tau)_{\mathcal{L}}, \mathcal{S}_a(\tau)_{\mathcal{L}+1}) \oplus ((A\tau)_{\mathcal{L}}, \mathcal{S}_a(A\tau)_{\mathcal{L}+1})$$

   à partir de la loi d'addition dans le groupe formel.
5. la série tronquée $\kappa_a(\tau)_{p^r} = \mathcal{R}_a(\tau, \mathcal{S}_a(\tau))_{p^r+1}$ et ces puissances jusqu'à l'ordre nécessaire pour l'algorithme incrémental de composition de séries.
6. toutes les séries intermédiaires de la proposition 23 nécessaires au calcul de $\mathcal{R}_b(\mathcal{U}(\tau), \mathcal{S}_b(\mathcal{U}(\tau)))$. Par exemple, en caractéristique 2,

$$\mathcal{R}_b(T, S) = \frac{T^3 + \tilde{b}_6 S^2 T}{T^2 + \tilde{b}_6 S^2 + \tilde{b}_6 T^2 s + s + Ts + T^3 + \tilde{b}_6 T S^2}$$

   et nous initialisons tous les monômes de cette fraction une fois substitué $(\mathcal{U}(\tau), \mathcal{S}_b(\mathcal{U}(\tau)))$ à (T, S) ; $\mathcal{U}(\tau)_1 = \tau$, $\mathcal{S}_b(\mathcal{U}(\tau))_3 = \tau^3$, $\mathcal{U}(\tau)\mathcal{S}_b(\mathcal{U}(\tau))_4 = \tau^4$, $\mathcal{U}^2(\tau)_2 = \tau^2$, $\mathcal{S}_b(\mathcal{U}(\tau))_6^2 = \tau^6$, ...
7. comme à l'étape 6, toutes les séries intermédiaires pour calculer $\mathcal{U}(\tau) \oplus \mathcal{U}(A\tau)$ comme indiqué dans la démonstration de la proposition 22 ; $\mathcal{U}(A\tau)_1 = A\tau$, $\mathcal{S}_b(\mathcal{U}(\tau))_3 = A^3\tau^3$, $\lambda(\tau)_2 = (A^2 + A + 1)\tau^2$, $\nu(\tau)_3 = (A^2 + A)\tau^3$, ...

Nous résumons les complexités en temps et espace de ces calculs dans le tableau 5.1.

La complexité totale de cette phase est au plus $O(\max(p\mathcal{L}^\mu, \mathcal{L}^2))$ avec un stockage en $O(\mathcal{L}^{3/2})$.

Étape	1	2	3	4	5	5	7
Temps	$O(\mathcal{L}^2)$	$O(\mathcal{L})$	$O(\mathcal{L})$	$O(\mathcal{L}^\mu)$	$O(p\mathcal{L}^\mu)$	$O(p)$	$O(p)$
Espace	$O(\mathcal{L})$	$O(1)$	$O(\mathcal{L})$	$O(1)$	$O(\mathcal{L}^{3/2})$	$O(p)$	$O(p)$

TAB. 5.1 – Complexité des précalculs.

Phase principale : au début de la $i^{\text{ème}}$ itération, $\mathcal{U}(\tau)_{i-1}$ est connu, ce qui se traduit d'après la proposition 21 par la connaissance des séries intermédiaires $\mathcal{S}_b(\mathcal{U}(\tau))_{i+1}$, $\mathcal{U}(A\tau)_{i-1}$, $\mathcal{S}_b(\mathcal{U}(\tau))_{i+1}$, $\lambda(\tau)_i$, $\nu(\tau)_{i+1}\ldots$. Alors, après un petit calcul formel, nous sommes capables de calculer u_i dont la connaissance nous permet de mettre à jour les séries intermédiaires afin d'être prêts pour la $(i+1)^{\text{ème}}$ itération. À cet effet, nous étudions séparément les cas $i \neq p^e$ et $i = p^e$ $(e \in \mathbb{N})$.

Le cas $i \neq p^e$: nous trouvons u_i à partir de

$$\mathcal{U}(\tau \oplus A\tau) = \mathcal{U}(\tau) \oplus \mathcal{U}(A\tau).$$

Les parties gauche et droite nous fournissent respectivement des expressions de la forme $(1+A)^i u_i + d$ et $(1+A^i)u_i + b$. Nous obtenons alors u_i par $u_i = (d-b)/(1+A^i-(1+A)^i)$.

Ce qui donne précisément :

Étape 1-a : nous devons calculer le $i^{\text{ème}}$ coefficient de $\mathcal{U}(\tau \oplus A\tau)$, ce que nous faisons en utilisant l'algorithme incrémental de Brent et Kung. Nous obtenons une expression du type $(1+A)^i u_i + d$.

Étape 1-b : nous devons calculer le $i^{\text{ème}}$ coefficient de $\mathcal{U}(\tau) \oplus \mathcal{U}(A\tau)$ en fonction de u_i. À cet effet, nous suivons pas à pas les calculs réalisés dans la démonstration de la proposition 22. Puisque chaque série intermédiaire nécessaire est connue jusqu'à l'ordre i, nous pouvons obtenir en fonction de u_i, le $(i+2)^{\text{ème}}$ coefficient de $\mathcal{S}_b(\mathcal{U}(\tau))$, le $i^{\text{ème}}$ coefficient de $\mathcal{U}(A\tau)$, le $(i+2)^{\text{ème}}$ coefficient de $\mathcal{S}_b(\mathcal{U}(\tau))$, le $(i+1)^{\text{ème}}$ coefficient de $\lambda(\tau)$ et ainsi de suite. Ainsi, le coefficient que nous cherchons est égal à $(1+A^i)u_i + b$.

La complexité de cette phase est $O(i)$.

Le cas $i = p^e$: nous trouvons u_i avec l'équation (5.19) que nous réécrivons en

$$\tilde{\mathcal{U}}(\kappa_a(\tau)) = \mathcal{R}_b(\mathcal{U}(\tau), \mathcal{S}_b(\mathcal{U}(\tau))).$$

Cela nous permet d'utiliser les mêmes techniques que celles décrites juste précédemment, c'est-à-dire appliquer $\tilde{\mathcal{U}}$ à une série connue en utilisant des précalculs puis calculer le développement en série d'une fraction rationnelle en $\mathcal{U}$ et $\mathcal{S}_b(\mathcal{U})$.

Ce qui donne précisément :

Étape 2-a : nous calculons en fonction de u_i le $i^{\text{ème}}$ coefficient de $\tilde{\mathcal{U}}(\tau) \circ \kappa_a(\tau)$. Ce que nous réalisons comme à l'étape 1-a. Ce coefficient est égal à $a\tilde{u}_i + d$.

Étape 2-b : nous calculons formellement le $i^{\text{ème}}$ coefficient de $\mathcal{R}_b(\mathcal{U}(\tau), \mathcal{S}_b(\mathcal{U}(\tau)))$. À cet effet, nous procédons comme à l'étape 1-b. Nous avons donc à obtenir en fonction de u_i le $(i+2)^{\text{ème}}$ coefficient de $\mathcal{S}_b(\mathcal{U}(\tau))$, le $(i+2)^{\text{ème}}$ coefficient de $\mathcal{S}_b(\mathcal{U}(\tau))$, ... Ce coefficient est égal à $u_i + b$.

Finalement $u_i^p - a^p u_i + b^p - d^p = 0$ et nous choisissons l'une des racines de cette équation pour u_i. Nous mettons ensuite à jour les séries intermédiaires, c'est-à-dire à partir de $\mathcal{U}(\tau)_i$, les séries intermédiaires tronquées $\mathcal{S}_b(\mathcal{U}(\tau))_{i+2}$, $\mathcal{U}(A\tau)_i$, $\mathcal{S}_b(\mathcal{U}(\tau))_{i+2}$, $\lambda(\tau)_{i+1}$, $\nu(\tau)_{i+2}\ldots$ comme expliqué dans les démonstrations des propositions 22 ou 23.

Nous remarquons que le calcul d'un morphisme est dominé par la composition de série. Le coût total en est donc $O(\mathcal{L}^{\mu+1/2}) = O(\ell^{\mu+1/2})$ et toutes les séries intermédiaires dont nous avons besoin ont $O(p\mathcal{L})$ termes.

Identifier morphisme et isogénie

Le problème est maintenant, étant donné un morphisme $\mathcal{M}(T)$ de $\mathcal{E}_a$ vers $\mathcal{E}_b$, de savoir si ce morphisme coïncide avec une isogénie. Il s'agit donc, en posant

$$Z_a(T) = \frac{\mathcal{S}_a(T)}{T} \text{ et } Z_b(T) = \frac{\mathcal{S}_b(T)}{T},$$

de trouver une série $\hat{M}(Z)$ telle que

$$Z_b(\mathcal{M}(T)) = \hat{M}(Z_a(T))$$

puis de calculer la fraction rationnelle dont le développement en série est celui de $\hat{M}$.

De $\mathcal{M}$ à $\hat{M}$: nous connaissons le développement en série de $Z_a(T) = T^2 + a_1T^3 + (a_1^2 + a_2)T^4 + O(T^5)$ et nous supposons que $Z_b(\mathcal{M}(T)) = m_2T^2 + \cdots + m_{4\ell+1}T^{4\ell+1} + O(T^{4\ell+2})$ pour calculer les coefficients de $\hat{M}(Z) = \hat{m}_1Z + \hat{m}_3Z^3 + \cdots + \hat{m}_{2\ell+1}Z^{2\ell+1} + O(Z^{2\ell+2})$. Nous trouvons ces coefficients un par un. Puisque nous aurons à calculer beaucoup de ces tests, il est rentable de précalculer les puissances impaires de $Z_a(T)$, c'est-à-dire $Z_a(T)^i$ pour $1 \leq i \leq 4\ell + 1$, i impair. Cela nécessite $O(\ell^2)$ éléments. Une description détaillée de cette procédure est donnée sur la figure 5.1

procedure `RecoverSeriesInZ`$(Z_b(\mathcal{M}(T)), Z_a(T))$
\# On écrit $Z_b(\mathcal{M}(T)) = m_2T^2 + \cdots + m_{4\ell+1}T^{4\ell+1} + O(T^{4\ell+2})$.
$\hat{m}_1 := m_2$
$W(T) := Z_b(\mathcal{M}(T)) - \hat{m}_1 Z_a(T)$
for $i := 1, \ldots, \ell$ **do**
  \# À ce stade, $W(T) = wT^{2i+1} + O(T^{2i+2})$
  $\hat{m}_{2i+1} := w$
  $W(T) := W(T) - \hat{m}_{2i+1} Z_a^{2i+1}(T)$
endfor
return$(\hat{m}_1Z + \hat{m}_3Z^3 + \cdots + \hat{m}_{2\ell+1}Z^{2\ell+1} + O(Z^{2\ell+2}))$
endprocedure

FIG. 5.1 – Obtention de $\hat{M}(Z)$ à partir de $\mathcal{M}(T)$.

La phase de précalcul nécessite $O(\ell^{\mu+1})$ opérations mais n'est réalisée qu'une fois. Le calcul principal coûte asymptotiquement $O(\ell^2)$.

Obtenir la fraction rationnelle : supposons $F(Z) = f_0 + f_1Z + \cdots + f_mZ^m$ et $G(Z) = g_0 + g_1Z + \cdots + g_mZ^m$, deux polynômes en Z. Alors

$$\frac{F(Z)}{G(Z)} = A(Z) = \sum_{k=0}^{\infty} a_kZ^k$$

où les a_k satisfont les relations de récurrence

$$\begin{cases} \sum_{i=0}^{k} g_i a_{k-i} &= f_k, \text{ si } 0 \leq k \leq m, \\ \sum_{i=0}^{m} g_i a_{k-i} &= 0 \text{ sinon.} \end{cases}$$

À l'inverse, étant donnée une série $A(Z)$ connue jusqu'à l'ordre $2m$, nous pouvons calculer sa (m, m)-approximation de Padé définie comme la fraction rationnelle $U(Z)/V(Z)$ où $\deg(U) \leq m$, $\deg(V) \leq m$ et où

$$A(Z)V(Z) - U(Z) = O(Z^{2m+1}).$$

Cette quantité peut-être calculée en $O(m^2)$ opérations [75] avec l'algorithme de Berlekamp-Massey ou en $O(m(\log m)^2)$ opérations [14] avec l'algorithme `EMGCD`. Notons que du point de vue pratique, les tailles des séries que nous manipulons sont telles que l'algorithme de Berlekamp est encore la méthode la plus rapide.

L'algorithme final et sa complexité : le test de l'isogénie peut être formalisé comme sur la figure 5.2.

procedure `IsogenyTest`$(\ell, \mathcal{M}(T), \mathcal{S}_a(T), \mathcal{S}_b(T))$
Étape 1
$Z_a(T) = \mathcal{S}_a(T)/T$
$Z_b(\mathcal{M}(T)) = \mathcal{S}_b(\mathcal{M}(T))/\mathcal{M}(T)$
Étape 2
$\hat{M}(Z) =$ `RecoverSeriesInZ`$(Z_b(\mathcal{M}(T)), Z_a(T))$
Étape 3
Retrouver la fraction $F(Z)/G(Z)$ qui est une $(\ell+1, \ell+1)$-approximation de Padé de $\hat{M}(Z)$.
Étape 4 : À ce stade, $F(Z)$ et $G(Z)$ ont des degrés $\leq \ell+1$ *a priori*.
if $\deg(F) = \deg(G) = \ell$ **and** $F(Z) = Z\hat{h}(Z)^2$**then**
  # $\hat{M}$ pourrait être l'isogénie que nous recherchons.
  Calculer le facteur $h_\ell(X)$ du $\ell^{\text{ème}}$ polynôme de ℓ division.
  if $[\ell]_{E_a}(X, Y) = O_{E_a}$ sur la courbe $\mathbb{F}_q[X, Y]/(E_a(X, Y), h_\ell(X))$ **then**
    return$(h_\ell(X))$
  endif
endif
# Ce morphisme ne coïncide pas avec l'isogénie.
return(0)
endprocedure

FIG. 5.2 – Test d'égalité d'un morphisme avec l'isogénie.

La première étape nécessite $O(\ell^\mu)$ opérations, la seconde $O(\ell^2)$, ce qui domine le coût de la troisième étape. Nous voyons donc que le coût du test d'isogénie (moins celui de la vérification ultime) est $O(\ell^2)$. Notons aussi que dans l'approche de "multiplication p-adique", nous avons déjà $\mathcal{S}_b(\mathcal{M}(T))$ à notre disposition.

Énumération des morphismes

Backtracking : il est facile de voir que le coût de cette approche est $O(\mathcal{L})$ fois le coût de la détermination d'un morphisme plus celui du test de l'isogénie. Le coût total est donc $O(\ell^{\max(\mu+3/2,3)})$.

Multiplication p-adique : nous n'avons pas réellement besoin de multiplier par un entier p-adique, mais simplement besoin de réaliser des additions dans le groupe formel jusqu'à trouver l'isogénie comme schématisé sur la figure 5.3. Ainsi nous obtenons $(\mathcal{M}(T), \mathcal{S}_b(\mathcal{M}(T)))$ avec une addition formelle entre un

procedure `ComputeIsogeny`$(\ell, \mathcal{E}_a, \mathcal{E}_b)$
Calculer un générateur $\mathcal{U}$ de l'ensemble des morphismes entre $\mathcal{E}$ et $\mathcal{E}_b$ en utilisant les algorithmes précédents.
for $N := 1, \ldots, p^{r+1}/2$, N premier avec p, **do**
  $(\mathcal{M}(T), \mathcal{S}_b(\mathcal{M}(T))) := [N]_b \circ (\mathcal{U}(T), \mathcal{S}_b(\mathcal{U}(T)))$
  $h_\ell(X) :=$ `IsogenyTest`$(\ell, \mathcal{M}(T), \mathcal{S}_a(T), \mathcal{S}_b(T))$
  if $h_\ell(X) \neq 0$ **then**
    return$(h_\ell(X))$
  endif
endfor
endprocedure

FIG. 5.3 – Calcul complet de l'isogénie.

point précédemment calculé et $(\mathcal{U}(T), \mathcal{S}_b(\mathcal{U}(T)))$ ou bien $[2]_b(\mathcal{U}(T), \mathcal{S}_b(\mathcal{U}(T)))$ selon $N \bmod p$. Le coût

de la seconde approche est celui de la détermination d'un morphisme ($O(\mathcal{L}^{\mu+1/2})$ multiplications) auquel s'ajoute $O(\mathcal{L})$ fois celui d'une addition dans le groupe formel ($O(\mathcal{L}^{\mu+1})$ multiplications), plus $O(\mathcal{L})$ fois celui du test d'isogénie ($O(\mathcal{L}^3)$). La complexité de cette seconde approche est donc $O(\ell^{\max(\mu+1,3)})$.

Asymptotiquement, si $\mu \leq 3/2$, les deux approches ont la même complexité $O(\ell^3)$. Si $\mu > 3/2$, la seconde est meilleure et la complexité est toujours $O(\ell^3)$. Quoiqu'il en soit, du point de vue pratique, la seconde approche est toujours meilleure, puisque mis à part le test de l'isogénie, nous remplaçons une substitution de série dont la complexité est $O(\mathcal{L}^{\mu+1/2})$ par une addition dans le groupe formel dont la complexité est seulement $O(\mathcal{L}^{\mu})$ (avec $1 \leq \mu \leq 2$).

Complexité asymptotique de l'ensemble

Nous résumons les complexités précédentes.

Proposition 25. *Après précalcul, le coût de l'algorithme de Couveignes est asymptotiquement $O(\ell^3)$ à p fixé. Le stockage est en $O(\ell^2)$.*

5.4 Résultats

5.4.1 Exemples

Selon la paramétrisation des courbes elliptiques utilisées, il y a trois cas à considérer : $p = 2$, $p = 3$ et $p > 3$, un représentant duquel sera $p = 5$. Le but est ici de donner des exemples d'utilisation de l'algorithme de Couveignes pour ces trois cas.

Corps finis de caractéristique 5

Soient $\mathbb{F}_{5^3} = \mathbb{F}_5[t]/(t^3 + t + 1)$, nous allons chercher une isogénie de degré $\ell = 3$ entre $E_a : Y^2 = X^3 + X + t$ et $E_b : Y^2 = X^3 + X + 3t^2 + 3t$.

Les développements en série de $[5]_a$ et $[5]_b$ sont :

$$\begin{aligned} [5]_a &= 2T^5 + O(T^7), \\ [5]_b &= (2t^2 + 2t + 4)\lambda^2 T^5 + O(T^7). \end{aligned}$$

Ici, nous avons $5 < 4 \times 3 < 5^2$ et nous devons calculer $\mathcal{U}$ jusqu'à l'ordre 12. Nous sélectionnons tout d'abord $A = t$ et nous posons $\mathcal{U}_1 = T$. Nous trouvons u_3 en écrivant l'égalité de $\mathcal{U}(T \oplus AT)$ et de $\mathcal{U}(T) \oplus \mathcal{U}(AT)$, ce qui conduit à

$$(3t^2 + 3t)u_3 = 0$$

ou $u_3 = 0$. Pour u_5, nous avons à égaliser $\mathcal{U} \circ [5]_a$ et $[5]_b \circ \mathcal{U}$, ce qui donne

$$t + 1 + 2u_5 + 3u_5^5 + 4t^2 = 0.$$

Les solutions de cette équation sont $t^2, t^2 + 1, t^2 + 2, t^2 + 3, t^2 + 4$. Nous choisissons $u_5 = t^2$. En réitérant ainsi de suite le procédé, nous obtenons finalement

$$\mathcal{U}(T) = T + t^2T^5 + (3t^2 + 2t)T^7 + 3t^2T^9 + (3t^2 + 4t + 4)T^{11} + (t^2 + t)T^{13} + O(T^{14}).$$

À ce stade, nous devons maintenant chercher un entier N, $1 \leq N \leq \lfloor p^{r+1}/2 \rfloor = 12$, $N \not\equiv 0 \bmod p$, tel que $[N]_b \circ \mathcal{U}$ soit la série associée à I. Nous trouvons tout d'abord

$$Z_a = \frac{\mathcal{S}_a(T)}{T} = T^2 + T^6 + tT^8 + 2T^{10} + O(T^{14}).$$

Avec $N = 1$,

$$[1]_b \circ \mathcal{U} = T + t^2T^5 + (3t^2 + 2t)T^7 + 3t^2T^9 + (3t^2 + 4t + 4)T^{11} + (t^2 + t)T^{13} + O(T^{14}),$$

et

$$Z_b([1]_b \circ \mathcal{U}) = T^2 + (2t^2+1)T^6 + (4t^2+2t)T^8 + (t^2+4t+2)T^{10} + (4t^2+2t)T^{12} + O(T^{14}),$$

qui peut être réécrit

$$Z_a(T) + 2t^2 Z_a(T)^3 + (4t^2+t)Z_a(T)^4 + 4tZ_a(T)^5 + (3t^2+4t+1)Z_a(T)^6 + O(Z_a(T)^7).$$

Puis, nous utilisons l'algorithme de Berlekamp-Massey pour retrouver la fraction, ce qui dans ce cas particulier donne

$$\frac{(2t^2+5t+5)Z^3 + (8t^2+6t+2)Z^2 + (5t^2+5t+1)Z}{(3t^2+3t+2)Z^4 + (3t^2+7t+7)Z^3 + 5t^2 + (3t^2+t+2)Z + 5t + 6}.$$

Cette fraction n'a pas le degré espéré, elle ne correspond donc pas à l'isogénie que nous cherchons. Nous devons donc tester les autres valeurs de N et finalement trouver, pour $N = 12$,

$$\begin{aligned} Z_b([12]_b \circ \mathcal{U}) &= 4T^2 + (2+3t^2)T^6 + (4t^2+t)T^8 + (2t^2+t)T^{10} + (t^2+t+1)T^{12} + O(T^{14}) \\ &= 4Z_a(T) + (3t^2+3)Z_a(T)^3 + (4t^2+2t)Z_a(T)^4 + (3t^2+t+3)Z_a(T)^5 + 3tZ_a(T)^6 + O(Z_a(T)^7) \end{aligned}$$

qui correspond à la fraction rationnelle

$$\frac{(3t^2+1)Z^3 + (2t^2+4t)Z^2 + Z}{(4t^2+t+3)Z^3 + (t^2+4)Z^2 + (2t^2+4t)Z + 1}$$

dont le numérateur est

$$Z(3t^2+1)(Z+4t+2)^2.$$

Le facteur du polynôme de ℓ division recherché est donc le polynôme réciproque de $Z+4\,t+2$, c'est-à-dire $h_3(X) = X + t^2 + 2t$. Il est maintenant facile de vérifier qu'il s'agit réellement d'un facteur, en vérifiant l'égalité $[3]_{E_a}(X,Y) = O_{E_a}$ dans $\mathbb{F}_{5^3}[X,Y]/(E_a(X,Y), h_3(X))$, ce qui est le cas ici.

Corps finis de caractéristique 3

Considérons $\mathbb{F}_{3^4} = \mathbb{F}_3[t]/(t^4+2t+2)$ et $E_a : Y^2 = X^3 + X^2 + t$ avec $\ell = 5$. La courbe isogène est ici $E_b : Y^2 = X^3 + X^2 + 2t^3 + 2t$. Nous calculons de la même façon que ci-dessus

$$\begin{aligned} \mathcal{U}(T) = T + tT^3 + 2T^5 t + 2t^2 T^7 + (2t^3+t^2+t)T^9 + (2t^3+2t)T^{11} + (2t^3+t^2+2t+2)T^{13} + \\ (t^2+2t+2)T^{15} + (t^2+2t+2)T^{17} + (2+t^3+t^2+2t)T^{19} + (2t^3+t^2+t+1)T^{21} + O(T^{23}). \end{aligned}$$

Puis pour $N = 2$, nous obtenons une fraction de degré attendu, dont le numérateur est

$$\left(t^3+t+2\right)\left(Z^2 + \left(2\,t^3+t^2+t+2\right)Z + 2\,t^2+1\right)^2$$

ce qui conduit à $g_5(X) = X^2 + (t+2)X + 2t^3 + 2t + 1$.

Corps finis de caractéristique 2

Soit $\mathbb{F}_{2^8} = \mathbb{F}_2[t]/(t^8+t^4+t^3+t+1)$. Chaque élément de $\mathbb{F}_{2^8}$ peut être réécrit en un polynôme en t. Nous écrirons un tel polynôme $a(t) = \sum_{i=0}^{n-1} a_i t^i$ en $\overline{a(2)}$. Par exemple, le polynôme t^2+t sera abrégé $\overline{6}$.

Calculons une isogénie de degré $\ell = 5$ entre $E_a : Y^2 + XY = X^3 + \overline{7}$ et $E_b : Y^2 + XY = X^3 + \overline{8}$. Nous trouvons tout d'abord que $A = \overline{2}$ est valide. L'équation (5.17) devient alors

$$0 \quad = \quad \left(\overline{6}\,u_2 + \overline{6}\,u_3\right)T^3 + \left(\overline{4}\,{u_2}^2 + \overline{4}\,u_2\right)T^4 + O(T^5)$$

et l'équation (5.19)

$$0 \quad = \quad \left(\sqrt{u_2}^2 + \sqrt{u_2}\right)T^2 + \left(\sqrt{u_3}^2 + \sqrt{u_3}\right)T^3 + \left(\sqrt{u_4}^2 + \sqrt{u_4} + \sqrt{\overline{15}}\right)T^4 + O(T^5).$$

Par conséquent,

$$\begin{array}{rcll}
\sqrt{u_2}^2+\sqrt{u_2}=0 & \Longrightarrow & u_2\in\{\overline{0},\overline{1}\}, & \text{nous utilisons } u_2=\overline{0},\\
\overline{6}\,u_2+\overline{6}\,u_3=0 & \Longrightarrow & u_3\in\{\overline{0}\}, & \text{nous prenons } u_3=\overline{0},\\
\sqrt{u_4}^2+\sqrt{u_4}+\sqrt{\overline{15}}=0 & \Longrightarrow & u_4\in\{\overline{56},\overline{57}\}, & \text{nous choisissons } u_4=\overline{56},
\end{array}$$

et une fois les calculs terminés, nous obtenons un morphisme $\mathcal{M}$ dont la série $\mathcal{U}$ associée est la suivante,

$$\begin{aligned}\mathcal{U}(T)=T&+\overline{56}T^4+\overline{56}T^5+\overline{15}T^7+\overline{16}T^8+\overline{31}T^9+\overline{219}T^{10}+\overline{124}T^{11}+\overline{5}T^{12}+\overline{44}T^{13}\\&+\overline{91}T^{14}+\overline{47}T^{15}+\overline{210}T^{16}+\overline{201}T^{17}+\overline{231}T^{18}+\overline{198}T^{19}+\overline{188}T^{20}+\overline{118}T^{21}+O(T^{22}).\end{aligned}$$

Utilisons tout d'abord l'approche "Backtracking". Le morphisme $\mathcal{M}_1$ obtenu à partir de $\mathcal{M}$ en posant $u_{16}=\overline{211}$ est

$$\begin{aligned}\mathcal{U}_1(T)=T&+\overline{56}T^4+\overline{56}T^5+\overline{15}T^7+\overline{16}T^8+\overline{31}T^9+\overline{219}T^{10}+\overline{124}T^{11}+\overline{5}T^{12}+\overline{44}T^{13}\\&+\overline{91}T^{14}+\overline{47}T^{15}+\overline{211}T^{16}+\overline{200}T^{17}+\overline{231}T^{18}+\overline{198}T^{19}+\overline{132}T^{20}+\overline{78}T^{21}+O(T^{22}).\end{aligned}$$

Alors le morphisme $\mathcal{M}_2$ obtenu à partir de $\mathcal{M}_1$ en posant $u_8=\overline{17}$ est

$$\begin{aligned}\mathcal{U}_2(T)=T&+\overline{56}T^4+\overline{56}T^5+\overline{15}T^7+\overline{17}T^8+\overline{30}T^9+\overline{219}T^{10}+\overline{124}T^{11}+\overline{61}T^{12}+\overline{20}T^{13}\\&+\overline{83}T^{14}+\overline{32}T^{15}+\overline{202}T^{16}+\overline{214}T^{17}+\overline{52}T^{18}+\overline{186}T^{19}+\overline{18}T^{20}+\overline{82}T^{21}+O(T^{22}).\end{aligned}$$

En continuant ainsi, nous obtenons

$$\begin{aligned}\mathcal{U}_3(T)=T&+\overline{56}T^4+\overline{56}T^5+\overline{15}T^7+\overline{17}T^8+\overline{30}T^9+\overline{219}T^{10}+\overline{124}T^{11}+\overline{61}T^{12}+\overline{20}T^{13}\\&+\overline{83}T^{14}+\overline{32}T^{15}+\overline{203}T^{16}+\overline{215}T^{17}+\overline{52}T^{18}+\overline{186}T^{19}+\overline{42}T^{20}+\overline{106}T^{21}+O(T^{22}),\end{aligned}$$

$$\begin{aligned}\mathcal{U}_4(T)=T&+\overline{57}T^4+\overline{57}T^5+\overline{15}T^7+\overline{40}T^8+\overline{39}T^9+\overline{211}T^{10}+\overline{115}T^{11}+\overline{29}T^{12}+\overline{59}T^{13}\\&+\overline{136}T^{14}+\overline{91}T^{15}+\overline{132}T^{16}+\overline{21}T^{17}+\overline{196}T^{18}+\overline{2}T^{19}+\overline{188}T^{20}+\overline{109}T^{21}+O(T^{22}),\end{aligned}$$

$$\begin{aligned}\mathcal{U}_5(T)=T&+\overline{57}T^4+\overline{57}T^5+\overline{15}T^7+\overline{40}T^8+\overline{39}T^9+\overline{211}T^{10}+\overline{115}T^{11}+\overline{29}T^{12}+\overline{59}T^{13}\\&+\overline{136}T^{14}+\overline{91}T^{15}+\overline{133}T^{16}+\overline{20}T^{17}+\overline{196}T^{18}+\overline{2}T^{19}+\overline{133}T^{20}+\overline{84}T^{21}+O(T^{22}),\end{aligned}$$

pour finalement trouver que

$$\begin{aligned}\mathcal{U}_6(T)=T&+\overline{57}T^4+\overline{57}T^5+\overline{15}T^7+\overline{41}T^8+\overline{38}T^9+\overline{211}T^{10}+\overline{115}T^{11}+\overline{36}T^{12}+\overline{2}T^{13}\\&+\overline{128}T^{14}+\overline{84}T^{15}+\overline{164}T^{16}+\overline{50}T^{17}+\overline{31}T^{18}+\overline{113}T^{19}+\overline{2}T^{20}+\overline{94}T^{21}+O(T^{22})\end{aligned}$$

coïncide avec la série $\mathcal{W}(T)$ de l'isogénie recherchée.

Avec l'approche p-adique, les additions dans le groupe formel conduisent à

$$\begin{array}{rcl}
[2]_b\circ\mathcal{U}(T) & = & T^2+\overline{63}T^8+\overline{63}T^{10}+\overline{29}T^{14}+\overline{184}T^{16}+\overline{165}T^{18}+\overline{227}T^{20}+O(T^{22})\\
[3]_b\circ\mathcal{U}(T) & = & T+T^2+T^3+\overline{56}T^4+\overline{56}T^5+\overline{56}T^6+\overline{55}T^7+\overline{39}T^8+\overline{39}T^9+\\
& & \overline{244}T^{10}+\overline{84}T^{11}+\overline{154}T^{12}+\overline{28}T^{13}+\overline{79}T^{14}+\overline{52}T^{15}+\overline{247}T^{16}+\\
& & \overline{51}T^{17}+\overline{44}T^{18}+\overline{66}T^{19}+\overline{102}T^{20}+\overline{84}T^{21}+O(T^{22})
\end{array}$$

Nous obtenons finalement à partir de $[3]_b\circ\mathcal{U}(T)$ ou à partir de $\mathcal{U}_6(T)$ avec l'algorithme de Berlekamp-Massey

$$I(X)=\frac{X^5+\overline{15}X^3+\overline{140}X}{(X^2+\overline{57}X+\overline{74})^2}.$$

Notons ici que $\mathcal{U}_6(T)$ est l'opposé de $[3]_b\circ\mathcal{U}(T)$, autrement dit $[3]_b\circ\mathcal{U}(T)=\mathcal{U}_6(T/(1+T))$.

5.4.2 Statistiques

Nous avons programmé les algorithmes des sections 5.2 et 5.3 avec la librairie `ZEN` (cf. chapitre 10) pour tout corps fini $\mathbb{F}_q$. Par soucis de simplification, nous ne donnons des temps précis que pour des corps finis de caractéristique 2.

Pour tout d'abord comparer les efficacités respectives des deux principales approches, le "backtracking" et la "multiplication p-adique", nous avons mesuré le temps nécessaire au calcul d'isogénies de degrés ℓ croissants définies sur $\mathbb{F}_{2^{10}}$ (temps moyens pour 50 courbes). Les résultats sont donnés sur les courbes de la figure 5.4.

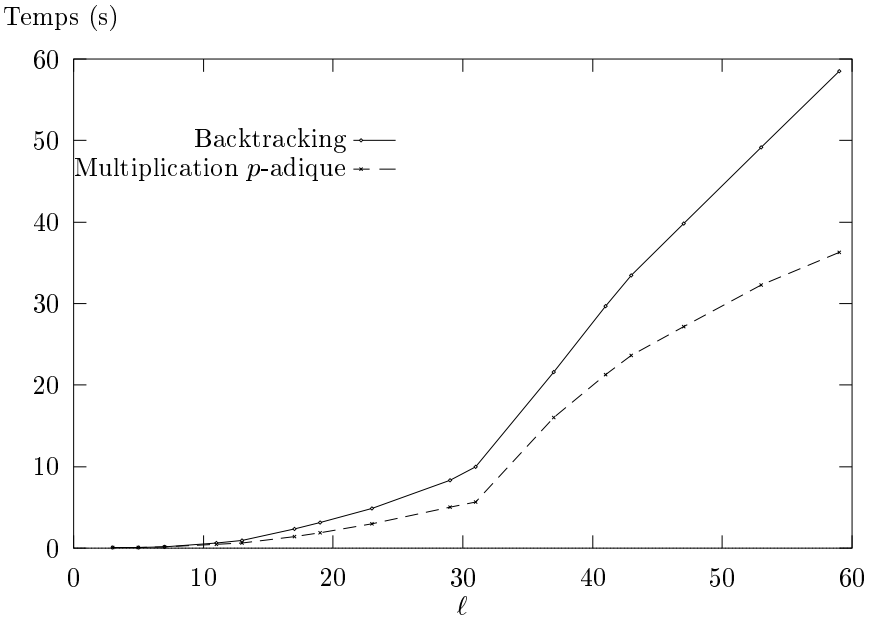


FIG. 5.4 – Temps moyens pour les deux approches (50 courbes sur station DEC).

Comme l'approche p-adique est la plus efficace en pratique, nous l'avons disséquée en sous-étapes dont nous avons aussi étudié le comportement moyen pour 50 courbes définies sur $\mathbb{F}_{2^{10}}$ et des degré ℓ croissants. Nous considérons notamment pour chaque nombre premier ℓ les temps nécessaires au calcul de "$\mathcal{U}$" (générateur des morphismes), "$\oplus$" (sommes de points formels), la phase de "Précalculs", "$Z_b(\mathcal{M}) = \hat{M}(Z_a)$" (appels à la routine `RecoverSeriesInZ`) et "Berlekamp-Massey" (appels à l'algorithme de Berlekamp-Massey). Les résultats sont tracés sur les graphes de la figure 5.5.

Enfin, pour $\mathbb{F}_{2^{300}}$, nous avons mesuré le temps nécessaire aux différentes parties de l'algorithme lors du calcul de la cardinalité de la courbe $Y^2 + XY = X^3 + \overline{91128}$. En plus des indications des graphes de la figure 5.5, nous trouvons dans le tableau 5.6, pour chaque nombre premier ℓ, le nombre de valeurs de J^* à essayer et l'entier 2-adique N tel que $[N]_b \circ \mathcal{U}$ coïncide avec l'isogénie. Les temps de $\oplus$, $Z_b(\mathcal{M}) = \hat{M}(Z_a)$ et BM (Berlekamp-Massey) sont ici des moyennes.

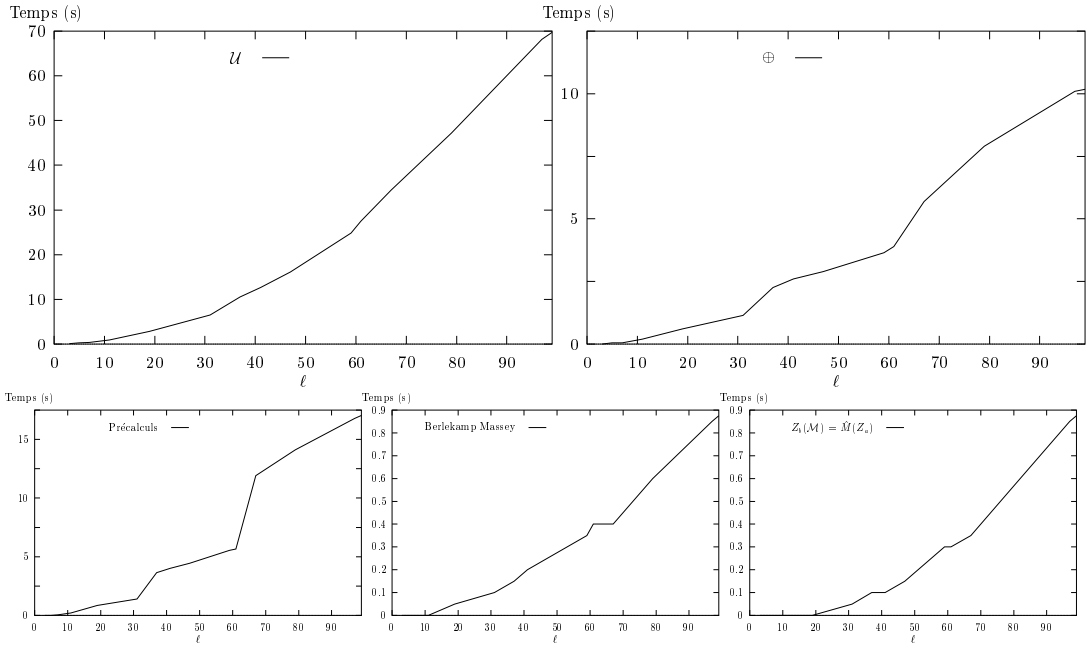


FIG. 5.5 – Sous-étapes de l'approche p-adique (station DEC).

ℓ	$\#J^*$	Prec	$\mathcal{U}$	$Z(T)^i$	$\oplus$	$Z_b(\mathcal{M}) = \hat{M}(Z_a)$	BM	N
3	1	0.0	0.1	0.0	0.0	0.0	0.0	3
5	1	0.0	0.2	0.0	0.0	0.0	0.0	7
7	1	0.0	0.4	0.0	0.0	0.0	0.0	7
11	2	0.2	0.9	0.0	0.2	0.0	0.0	7
19	3	2.6	2.8	0.3	0.6	0.0	0.0	47
31	1	1.4	6.6	1.1	1.1	0.0	0.1	63
37	3	3.6	10.5	1.8	2.2	0.1	0.1	127
41	1	4.0	12.6	2.5	2.6	0.1	0.2	81
47	1	8.4	16.1	3.7	2.9	0.1	0.2	111
59	1	5.0	24.8	6.8	3.9	0.3	0.3	103
61	1	5.6	27.5	7.3	3.9	0.3	0.4	3
67	3	23.8	34.4	9.2	5.2	0.3	0.4	93
79	3	14.1	47.2	14.9	7.9	0.5	0.6	133
97	1	16.8	68.1	26.3	10.1	0.8	0.8	79
101	1	17.3	71.2	29.3	10.2	0.9	0.9	161
103	1	17.5	73.7	31.0	10.3	0.9	0.9	37
107	5	72.05	79.5	34.1	10.8	1.0	1.0	251
Total	-	189.5	478	34.1	-	-	-	-

FIG. 5.6 – Statistiques pour $\mathbb{F}_{2^{300}}$ (station DEC).

Chapitre 6

Deuxième algorithme de Couveignes en petite caractéristique

Tout comme il est possible de calculer des isogénies entre courbes elliptiques définies sur $\mathbb{C}$ par deux algorithmes, l'algorithme `CCR` basé sur les dérivées de la fonction de Weierstrass et l'algorithme d'Atkin basé sur le comportement au voisinage de $z = 0$ de l'isogénie, les idées de Couveignes ont en quelque sorte aussi deux visages. Le premier est celui de son premier algorithme (chapitre 5) dans lequel il remplace la dérivée $\frac{d\wp(z)}{dz}$ par la "dérivée discrète" $\frac{X(t)}{t}$ (Pour un parallèle précis entre ces deux approches, voir [115]). Le deuxième visage est l'algorithme qui fait l'objet de ce chapitre et qui consiste à étudier le comportement de l'isogénie au voisinage de "l'équivalent de $z = 0$" pour une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ de caractéristique p, c'est-à-dire les points de p^k-torsion ($k \in \mathbb{N}^*$).

Au premier abord, cet algorithme semble assez simple puisqu'il s'agit ni plus ni moins d'une interpolation de l'isogénie sur les points de p^k-torsion des courbes isogènes [31]. Nous nous sommes pourtant rapidement rendu compte que les calculs qui seraient mis en œuvre par une implantation naïve de ces idées conduiraient à des coûts prohibitifs. Grâce notamment à une méthode ingénieuse de calcul des points de p^k-torsion sur une courbe elliptique définie sur $\mathbb{F}_q$ initiée par Hasse et Witt et explicitée par Voloch [114], ces coûts peuvent être notablement réduits. À tel point qu'en conjonction avec d'autres idées de Couveignes [32], la complexité asymptotique de cet algorithme est de $O(n\ell^{1+\epsilon})$ opérations dans $\mathbb{F}_q$ à p fixé.

Une fois donné le principe de l'algorithme (section 6.1), nous expliquons comment déterminer efficacement des points de p^k-torsion sur une courbe elliptique définie sur un corps fini (section 6.2) et nous décrivons ensuite précisément les parties "critiques" de l'algorithme de Couveignes (section 6.3) avant de donner dans une quatrième partie des exemples d'applications et les résultats que nous obtenons avec notre implantation (section 6.4). Faute de temps, nous n'avons pas encore programmé toutes les optimisations récemment proposées par Couveignes et prises en compte pour le calcul de la complexité asymptotique. Nous donnons néanmoins les gains que nous pourrions avoir après la mise en place de ces améliorations.

Remarque : nous nous restreignons par soucis de simplification au cas $p \geqslant 5$, mais ces résultats s'étendent sans problème aux cas $p = 2$ et $p = 3$. De plus, comme le seul point de p^k-torsion d'une courbe elliptique supersingulière E est O_E (cf. théorème 26), nous nous restreignons aussi aux courbes non supersingulières. Enfin, nous supposons le degré ℓ des isogénies impair puisque le cas $\ell = 2$ a été traité au chapitre 4.

6.1 Principe de l'algorithme

Les points de p^k-torsion ($k \in \mathbb{N}^*$) d'une courbe elliptique E_a définie sur un corps fini $\mathbb{F}_q$ de caractéristique p sont à la base du deuxième algorithme de Couveignes. En règle générale, ces points ne sont bien

sûr pas définis dans $\mathbb{F}_q$ mais dans des extensions algébriques de $\mathbb{F}_q$. Nous rappelons quelques propriétés de ces points qui nous serons utiles par la suite, puis nous décrivons l'algorithme.

6.1.1 Propriétés des points de p^k-torsion

D'après le chapitre 2, nous savons que $E_a[p^k]$ forme un sous-groupe cyclique d'ordre p^k de $E_a\left(\bar{\mathbb{F}}_q\right)$. Pour calculer $E_a[p^k]$, il suffit donc d'en construire un générateur $P_k \in E_a\left(\bar{\mathbb{F}}_q\right)$.

En pratique, nous ne pouvons pas travailler dans $\bar{\mathbb{F}}_q$ mais plutôt dans des extensions algébriques $\mathbb{K}_k$ de $\mathbb{F}_q$.

Définition 19. *Pour tout entier $k \in \mathbb{N}^*$, on appelle $\mathbb{K}_k$ l'extension algébrique de degré minimal de $\mathbb{F}_q$ telle que $E_a[p^k] \subset E_a(\mathbb{K}_k)$.*

Le degré de cette extension est un élément important pour la faisabilité des calculs. Elle est déterminée par le théorème suivant [31].

Théorème 44. *Soit E_a une courbe elliptique non supersingulière définie dans un corps fini $\mathbb{F}_q$ de cardinalité $q+1-c$. Alors le degré de $\mathbb{K}_k$ par rapport à $\mathbb{F}_q$ est égal à l'ordre de c dans le sous-groupe multiplicatif de $\mathbb{Z}/p^k\mathbb{Z}$.*

Pour $E_a[p]$ et plus particulièrement $\mathbb{K}_1$, les travaux de Gunji et notamment le théorème 42 spécifient que le degré de $\mathbb{K}_1$ par rapport à $\mathbb{F}_q$ est le plus petit diviseur r de $p-1$ tel que $\gamma^r \in \mathbb{F}_q$ où $\gamma \in \bar{\mathbb{F}}_q$, est une racine $(p-1)$-ième de l'invariant de Hasse H_{E_a} de E_a.

Pour $E_a[p^k]$, $k>1$, le comportement de $\mathbb{K}_k$ est résumé par le résultat suivant.

Proposition 26. *Soit $k \in \mathbb{N}^*$, on a $[\mathbb{K}_k : \mathbb{K}_{k-1}] = p$ ou $\mathbb{K}_k = \mathbb{K}_{k-1} = \cdots = \mathbb{K}_1$.*

La démonstration de ce phénomène découle du théorème 44 et des petits lemmes suivants où nous notons $\mathrm{ord}_k(c)$ l'ordre d'un entier c premier avec p dans $(\mathbb{Z}/p^k\mathbb{Z})^*$.

Lemme 7. *Soient p un nombre premier et c un entier premier avec p. Alors, pour tout entier strictement positif k, $\mathrm{ord}_k(c)$ divise $\mathrm{ord}_{k+1}(c)$.*

Démonstration : de $c^{\mathrm{ord}_{k+1}(c)} = 1 \bmod p^{k+1}$, nous déduisons que $c^{\mathrm{ord}_{k+1}(c)} = 1 \bmod p^k$. □

Lemme 8. *Soient p un nombre premier et c un entier premier avec p. Alors, pour tout entier strictement positif k, $\mathrm{ord}_{k+1}(c)$ est égal à $\mathrm{ord}_k(c)$ ou à $p \times \mathrm{ord}_k(c)$.*

Démonstration : en posant $c^{\mathrm{ord}_k(c)} = 1 + \alpha p^k$ avec $\alpha \in \mathbb{N}^*$, nous avons $c^{p\,\mathrm{ord}_k(c)} = (1+\alpha p^k)^p \equiv 1 \bmod p^{k+1}$. Donc $\mathrm{ord}_{k+1}(c)$ divise $p\,\mathrm{ord}_k(c)$. Comme d'autre part $\mathrm{ord}_k(c)$ divise $\mathrm{ord}_{k+1}(c)$, on a $\mathrm{ord}_{k+1}(c) = \lambda\,\mathrm{ord}_k(c)$ avec $\lambda = 1$ ou $\lambda = p$. □

Corollaire 4. *Soient p un nombre premier et c un entier premier avec p. Alors, pour tout entier k strictement supérieur à 1, si $\mathrm{ord}_{k+1}(c) = \mathrm{ord}_k(c)$, alors $\mathrm{ord}_k(c) = \mathrm{ord}_{k-1}(c)$.*

Démonstration : supposons $\mathrm{ord}_k(c) \neq \mathrm{ord}_{k-1}(c)$. Nous déduisons du lemme 8 que $\mathrm{ord}_{k+1}(c) = \mathrm{ord}_k(c) = p\,\mathrm{ord}_{k-1}(c)$. En posant $c^{\mathrm{ord}_{k-1}(c)} = 1+\alpha p^{k-1}$ avec $\alpha \in \mathbb{N}^*$, nous avons alors $c^{p\,\mathrm{ord}_{k-1}(c)} = (1+\alpha p^k)^p \equiv 1+\alpha p^k \bmod p^{k+1}$. Comme d'autre part $\mathrm{ord}_{k+1}(c) = p\,\mathrm{ord}_{k-1}(c)$, nous avons aussi $c^{p\,\mathrm{ord}_{k-1}(c)} \equiv 1 \bmod p^{k+1}$. D'où $\alpha = 0 \bmod p$ et donc $c^{\mathrm{ord}_{k-1}(c)} \equiv 1 \bmod p^k$, ce qui est en contradiction avec notre hypothèse $\mathrm{ord}_k(c) = p\,\mathrm{ord}_{k-1}(c)$. □

6.1.2 Algorithme

Soient E_a et E_b deux courbes isogènes de degré égal à un nombre premier ℓ impair distinct de p. Le deuxième algorithme de Couveignes pour calculer une isogénie $\mathcal{I}_\ell$ entre ces deux courbes utilise le fait qu'une isogénie envoie $E_a[p^k]$ dans $E_b[p^k]$.

L'idée de l'algorithme est simple, il s'agit à partir d'un générateur P_k de $E_a[p^k]$ et Q_k de $E_b[p^k]$ de trouver l'entier λ premier avec p tel que $\mathcal{I}_\ell(P_k) = [\lambda]_{E_b}(Q_k)$ (cf. [31]). Lorsque cela est le cas, nous avons

alors $\forall i = 0, \ldots, p^k - 1$, $\mathcal{I}_\ell([i]_{E_a}(P_k)) = [i]_{E_b} \circ [\lambda]_{E_b}(Q_k)$ et si k est suffisamment grand nous sommes en mesure de calculer $\mathcal{I}_\ell$. En particulier, il est possible de vérifier que notre hypothèse sur λ était correcte.

Plus précisément, supposons l'isogénie $\mathcal{I}_\ell$ que nous recherchons s'écrive

$$\begin{array}{rccl} \mathcal{I}_\ell \ : & E_a(\mathbb{F}_q) & \to & E_b(\mathbb{F}_q) \\ & P & \mapsto & \left\{ \begin{array}{ll} O_{E_b} & \text{si } P = O_{E_a}, \\ \left(\dfrac{g_\ell(X)}{h_\ell^2(X)}, Y \dfrac{k_\ell(X)}{h_\ell^3(X)} \right) & \text{si } P = (X, Y) \end{array} \right. \end{array}$$

avec $\deg(g_\ell) = \ell$ et $\deg(h_\ell) = (\ell - 1)/2$. Une fois fixé un générateur P_k de $E_a[k]$, nous allons tester successivement les multiples de Q_k, c'est-à-dire $[\lambda]_{E_b}(Q_k)$ pour $\lambda = 0, \ldots, p^k - 1$ avec $\lambda \neq 0$ modulo p. Pour cela, nous construisons par interpolation le polynôme $A(X)$ qui prend pour valeur $X_{[i\lambda]Q_k}$ en $X_{[i]P_k}$ pour $i = 1, \ldots, (p^k - 3)/2$. Ainsi, les formules de Lagrange conduisent à

$$A(X) = \sum_{i=1}^{(p^k-3)/2} X_{[i\lambda]Q_k} \prod_{j \neq i} \frac{X - X_{[j]P_k}}{X_{[i]P_k} - X_{[j]P_k}}.$$

Ce polynôme a la propriété suivante :

$$\frac{g_\ell(X)}{h_\ell(X)^2} \equiv A(X) \bmod H(X) \text{ où } H(X) = \prod_{i=1}^{(p^k-3)/2} (X - X_{[i]P_k}).$$

Pour retrouver h_ℓ et g_ℓ à partir de cette identité, notons que nous devons avoir $p^k > 4\ell + 1$. Dans ce cas, nous obtenons g_ℓ à l'aide d'algorithme d'Euclide étendu :

$$\begin{array}{rcl} H(X) & = & Q_1(X)A(X) + R_1(X), \ \deg(R_1) < \deg(A), \\ A(X) & = & Q_2(X)R_1(X) + R_2(X), \ \deg(R_2) < \deg(R_1) \\ \cdots & = & \cdots \\ R_i(X) & = & Q_{i+2}(X)R_{i+1}(X) + R_{i+2}(X), \ \deg(R_{i+2}) < \deg(R_{i+1}) \end{array}$$

Pour $i \geqslant 1$, introduisons

$$U_{-1} = 1, V_{-1} = 0, U_0 = 0, V_0 = 1,$$

et

$$\begin{array}{rcl} U_{i+2}(X) & = & U_i(X) - Q_{i+2}(X)U_{i+1}(X), \\ V_{i+2}(X) & = & V_i(X) - Q_{i+2}(X)V_{i+1}(X). \end{array}$$

Il est alors bien connu que

$$R_i(X) = H(X)U_i(X) + A(X)V_i(X).$$

Lorsque $\deg(V_i) = \ell - 1$, nous avons

$$A(X)V_i(X) \equiv R_i(X) \bmod H(X)$$

et, si V_i est un carré, nous trouvons h_ℓ par le simple calcul de sa racine carrée.

6.2 Points de p^k-torsion

Nous expliquons maintenant comment construire les points de p^k-torsion d'une courbe elliptique afin d'en dériver des algorithmes efficaces.

6.2.1 Constructions de $E_a[p^k]$

Nous proposons ici deux constructions mathématiques de l'extension $\mathbb{K}_k$ sur laquelle est définie $E_a[p^k]$.
- La première construction est la construction naturelle. Elle conduit à une famille d'extensions algébriques définies par des polynômes irréductibles de $\mathbb{F}_q[X]$ à degrés croissants dont nous donnons une construction algorithmique dans la section 6.2.2.
- La deuxième construction est basée sur des travaux de Voloch [114]. Elle conduit à une famille d'extensions d'Artin-Schreier $\mathbb{U}_k$ définies par la donnée d'un polynôme $U^p - U - v$ où $v \in \mathbb{U}_{k-1}$. Nous en donnons aussi une construction algorithmique dans la section 6.2.3.

Notons déjà que les performances d'applications faisant usage de la p^k-torsion calculée par l'un de ces deux algorithmes, comme par exemple les calculs de la section 6.3, dépendent fortement de la structure choisie pour les extensions $\mathbb{K}_k$.

Approche naturelle

Une première méthode de construction des extensions algébriques $\mathbb{K}_k$ est l'analogue de la méthode utilisée au chapitre 3 pour calculer les points de ℓ^k-torsion. Elle consiste à construire les extensions $\mathbb{X}_k$ et $\mathbb{Y}_k$ définies comme suit.

Définition 20. *Pour tout entier $k \in \mathbb{N}^*$, on appelle $\mathbb{X}_k$ (resp. $\mathbb{Y}_k$) l'extension algébrique de degré minimal de $\mathbb{F}_q$ contenant les abscisses (resp. les ordonnées) des points de $E_a[p^k]$.*

Grâce aux travaux de Gunji, nous sommes déjà en mesure de calculer un générateur P_1 de $E_a[p]$ dans l'extension $\mathbb{U}_1$ définie par

$$\mathbb{U}_1 = \begin{cases} \mathbb{F}_q & \text{si } r = 1, \\ \mathbb{F}_q[U_1]/(U_1^r - \gamma^r) & \text{sinon,} \end{cases}$$

où $\gamma \in \bar{\mathbb{F}}_q$ est une racine $(p-1)$-ième de l'invariant de Hasse H_{E_a} de E_a et où r est le plus petit diviseur de $(p-1)$ tel que $\gamma^r \in \mathbb{F}_q$. Il suffit alors de calculer le polynôme minimal $m_1(X) \in \mathbb{F}_q[X]$ de X_1 pour obtenir $\mathbb{X}_1$ par $\mathbb{X}_1 = \mathbb{F}_q/(m_1(X))$. Notons que le degré d de $\mathbb{X}_1$ est généralement deux fois plus petit que celui de $\mathbb{U}_1$ et qu'il est alors nécessaire d'obtenir l'ordonnée de P_1 dans une extension $\mathbb{Y}_1$ de degré 2 sur $\mathbb{X}_1$.

Ensuite, soit $E_{\tilde{a}}$ la courbe dont les coefficients sont les racines p-ième $\tilde{a}_i$ des coefficients a_i de la courbe E_a. Il est alors possible avec les idées de Gunji (cf. chapitre 5) de construire une isogénie séparable $\mathcal{I}_p$ de degré p donnée par

$$\begin{array}{rccl} \mathcal{I}_p : & E_a(\bar{\mathbb{F}}_q) & \to & E_{\tilde{a}}(\bar{\mathbb{F}}_q) \\ & P & \mapsto & \begin{cases} O_{E_{\tilde{a}}} & \text{si } P = O_{E_a}, \\ \left(\dfrac{g_p(X)}{h_p^2(X)}, Y\dfrac{k_p(X)}{h_p^3(X)}\right) & \text{si } P = (X,Y). \end{cases} \end{array}$$

Rappelons que $\mathcal{I}_p$ est l'isogénie duale de l'isogénie inséparable ϕ_p définie de $E_{\tilde{a}}$ vers E_a par $(X,Y) \mapsto (X^p, Y^p)$. L'idée principale découle de la proposition suivante.

Proposition 27. *Avec les notations précédentes, l'image par $\mathcal{I}_p$ d'un générateur $P_k = (X_k,\ Y_k)$ du groupe des points de p^k-torsion de E_a est un point $(\tilde{X}_{k-1}, \tilde{Y}_{k-1})$ de p^{k-1}-torsion de $E_{\tilde{a}}$ dont l'image par ϕ_p est un générateur $P_{k-1} = (X_{k-1}, Y_{k-1})$ du groupe des points de p^{k-1}-torsion de E_a.*

Nous illustrons cette construction sur la figure 6.1.

À partir de $\mathbb{X}_1$ et de $P_1 = (X_1, Y_1)$, il suffit d'écrire $\mathcal{I}(P_2) = (\tilde{X}_1, \tilde{Y}_1)$ pour trouver X_2 comme racine du polynôme $g_p(X) - \tilde{X}_1 h_p^2(X)$. D'après le lemme 8, deux cas sont possibles. Si ce polynôme est irréductible, il définit une extension algébrique $\mathbb{X}_2$ de degré p de $\mathbb{X}_1$. Sinon, il se factorise complètement dans $\mathbb{X}_1$ et nous avons $\mathbb{X}_2 = \mathbb{X}_1$. Dans tous les cas, $Y_2 = \tilde{Y}_1 h_p^3(X_2)/k_p(X_2)$, ce qui définit $\mathbb{Y}_2$. Un raisonnement identique nous permet de construire $\mathbb{X}_k$ une fois connu $\mathbb{X}_{k-1}$. Nous schématisons cette construction sur la figure 6.2.

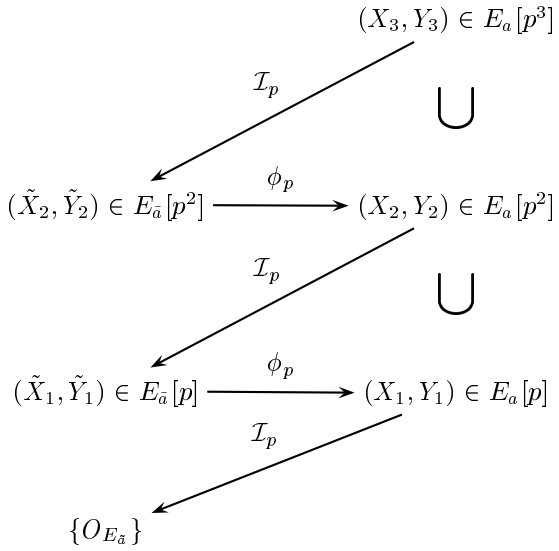


FIG. 6.1 – Points de p^k-torsion d'une courbe elliptique E_a.

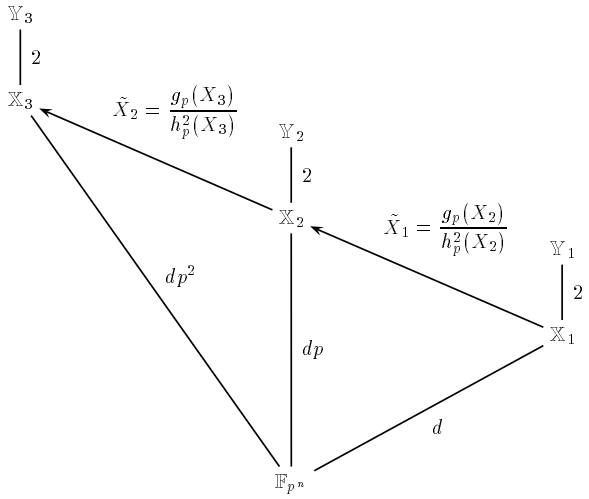


FIG. 6.2 – Extensions utilisées pour le calcul de la p^k-torsion avec des cycles d'isogénies.

Extensions d'Artin Schreier

Une construction moins classique des extensions $\mathbb{K}_k$ provient des travaux de Voloch [114] qui explicite pour le cas des courbes elliptiques une construction prônée par Hasse et Witt. Elle permet de représenter $\mathbb{K}_k$ par une tour d'extensions d'Artin-Schreier $\mathbb{U}_k$. Ceci est rendu possible par le théorème suivant.

Théorème 45. *Soient E_a une courbe elliptique définie sur un corps fini $\mathbb{F}_q$, $k \in \mathbb{N}^*$ et (X_k, Y_k) les coordonnées dans une extension algébrique $\mathbb{U}_k$ de $\mathbb{F}_q$ d'un générateur P_k de $E_a[p^k]$. Soient de plus $\mathcal{I}_p$ l'isogénie de degré p définie de E_a dans $E_{\tilde{a}}$ et duale du Frobenius ϕ_p et P_1 un point non nul de p-torsion sur E_a. On appelle $V_a(X)$ le quotient du polynôme $(X^3 + a_4X + a_6)^{\frac{p-1}{2}}$ par X^p et v_k la constante de Voloch définie dans $\mathbb{U}_k$ par*

$$v_k^p = \frac{Y_k V_a(X_k)}{H_{E_a}\gamma},$$

où H_{E_a} est l'invariant de Hasse de E_a et γ est une racine $(p-1)$-ième de H_{E_a} dans $\mathbb{U}_k$. Alors un générateur $P_{k+1} = (X_{k+1}, Y_{k+1})$ de $E_a[p^{k+1}]$ est défini dans l'extension $\mathbb{U}_{k+1}$ de $\mathbb{U}_k$ égale à

$$\mathbb{U}_{k+1} = \begin{cases} \mathbb{U}_k & \text{si } \mathrm{Tr}_{\mathbb{U}_k/\mathbb{F}_p}(v_k) = 0 \\ \mathbb{U}_k[U_{k+1}]/(U_{k+1}^p - U_{k+1} - v_k) & \text{sinon.} \end{cases}$$

Dans tous les cas, on pose u_{k+1} une racine de $X^p - X - v_k$ dans $\mathbb{U}_{k+1}$. Le point P_{k+1} est alors déterminé par le système

$$\begin{cases} (\tilde{X}_k, \tilde{Y}_k) &= \mathcal{I}_p(X_{k+1}, Y_{k+1}), \\ u_{k+1} &= -2\dfrac{Y_{k+1}}{\tilde{c}_{E_a}\tilde{\gamma}} \displaystyle\sum_{i=1}^{(p-1)/2} \frac{1}{X_{k+1} - X_{iP_1}}, \end{cases} \tag{6.1}$$

où X_{iP_1} est l'abscisse de iP_1.

Ainsi, une fois construit $\mathbb{U}_1$ et P_1 à l'aide du théorème 42, il est possible d'obtenir une extension algébrique $\mathbb{U}_2$ de $\mathbb{U}_1$ sur laquelle est définie un générateur P_2 de $E_a[p^2]$ à l'aide du théorème 45. Un raisonnement identique s'applique plus généralement pour obtenir $\mathbb{U}_{k+1}$ et P_{k+1} à partir de $\mathbb{U}_k$ et P_k comme nous l'illustrons sur la figure 6.3.

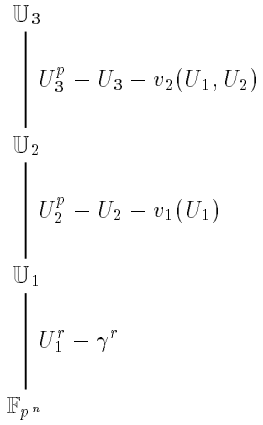


FIG. 6.3 – Extensions utilisées pour le calcul de la p^k-torsion avec les idées de Voloch.

6.2.2 Algorithme "naturel"

L'algorithme "naturel" découle en droite ligne de la proposition 27. On y construit des extensions algébriques $\mathbb{Y}_k$ généralement définies comme des extensions de degré 2 sur des extensions algébriques $\mathbb{X}_k$ de $\mathbb{F}_q$ elles-même définies à l'aide du polynôme minimal $m_k(X) \in \mathbb{F}_q[X]$ de X_k.

Autrement dit, nous construisons une suite d'extensions algébriques $\mathbb{X}_1, \ldots, \mathbb{X}_k$ de degrés croissants sur $\mathbb{F}_q$ et contenant respectivement les abscisses $X_1, \ldots, X_k$. De plus, le corps $\mathbb{X}_k$ est isomorphe à un sous-corps de $\mathbb{X}_{k+1}$. Sauf pour P_1 où si $Y_1 \in \mathbb{X}_1$, nous factorisons le polynôme $Y^2 - X_1^3 - a_4X_1 - a_6$ et nous posons $\mathbb{Y}_1 = \mathbb{X}_1$, nous évitons systématiquement, dès que $\mathbb{X}_k$ est distinct de $\mathbb{X}_1$, cette factorisation pour trouver Y_k et préférons poser

$$\mathbb{Y}_k = \mathbb{X}_k[Y_k]/(Y_k^2 - X_k^3 - a_4X_k - a_6).$$

Quand le degré de $\mathbb{X}_k$ est égal à l'ordre de c modulo p^k, $\mathbb{Y}_k$ n'est pas un corps fini, mais simplement un anneau. Pour éviter une factorisation coûteuse, nous utilisons la technique dite "des calculs paresseux à la Duval". C'est-à-dire que nous travaillons dans $\mathbb{Y}_k$ comme si c'était un corps mais il nous faut garder à l'esprit que lorsque nous utilisons un générateur P_k obtenu dans cette extension, quelques rares inversions peuvent être éventuellement impossibles. Quand cela se produit, nous obtenons l'ordonnée Y_k dans $\mathbb{X}_k$ par un simple pgcd et posons en cours de calcul $\mathbb{Y}_k = \mathbb{X}_k$.

Pour initier la récurrence, nous commençons par $E_a[p]$. Nous calculons donc :

1. $\mathbb{U}_1$, une extension de $\mathbb{F}_q$ de plus petit degré r contenant $\gamma = \sqrt[p-1]{H_{E_a}}$, c'est-à-dire
$$\mathbb{U}_1 = \begin{cases} \mathbb{F}_q & \text{si } r = 1, \\ \mathbb{F}_q[U]/(U^r - \gamma^r) & \text{sinon.} \end{cases}$$
2. x_1, l'abscisse dans $\mathbb{U}_1$ d'un générateur P_1 de $E_a[p]$ avec le théorème 42.
3. $m_1(X) \in \mathbb{F}_q[X]$, le polynôme minimal de x_1 de degré généralement égal à $r/2$.
4. $\mathbb{X}_1$, la plus petite extension algébrique de $\mathbb{F}_q$ contenant x_1, c'est-à-dire
$$\mathbb{X}_1 = \begin{cases} \mathbb{F}_q & \text{si } \deg(m_1) = 1, \\ \mathbb{F}_q[X_1]/(m_1(X_1)) & \text{sinon.} \end{cases}$$
5. $\mathbb{Y}_1$, la plus petite extension algébrique de $\mathbb{F}_q$ contenant l'ordonnée de P_1, c'est-à-dire
$$\mathbb{Y}_1 = \begin{cases} \mathbb{X}_1 & \text{si } \deg(m_1) = r, \\ \mathbb{X}_1[Y_1]/(Y_1^2 - X_1^3 - a_4X_1 - a_6) & \text{sinon.} \end{cases}$$
On a alors $P_1 = (X_1, Y_1)$.
6. c modulo p en cherchant comme au chapitre 3 l'entier c tel que $\phi_p(P_1) = [c]_{E_a}(P_1)$ dans $E_a(\mathbb{Y}_1)$.

Pour $E_a[p^k]$, $k > 1$, nous calculons tout d'abord $\mathcal{I}_p$. Dans un premier temps nous obtenons le polynôme $h_p(X)$ à partir des abscisses définies dans $\mathbb{X}_1$ des multiples de P_1. Nous appliquons les formules de Vélu pour trouver ensuite $g_p(X)$ et $k_p(X)$. Pour calculer P_k, il suffit alors d'écrire, une fois connu P_{k-1}, que $\phi_p^{-1}(P_{k-1}) = \mathcal{I}_p(P_k)$ ou encore

$$\tilde{X}_{k-1} = \frac{g_p(X_k)}{h_p^2(X_k)} \text{ avec } \tilde{m}_{k-1}\left(\tilde{X}_{k-1}\right) = 0.$$

Précisément, nous calculons :

1. $m(X) \in \mathbb{F}_q[X]$, le numérateur de $\tilde{m}_{k-1}\left(\dfrac{g_p(X)}{h_p^2(X)}\right)$.
2. $\mathbb{X}$, l'extension algébrique de $\mathbb{F}_q$ définie par $\mathbb{X} = \mathbb{F}_q[X_k]/(m(X_k))$.
3. $\mathbb{Y}$, une extension algébrique de $\mathbb{X}$ définie à l'aide de l'équation de la courbe par
$$\mathbb{Y} = \mathbb{X}[Y_k]/(Y_k^2 - X_k^3 - a_4X_k - a_6).$$
Alors $P = (X_k, Y_k)$ est un point de p^k-division.
4. c modulo p^k en cherchant comme au chapitre 3 l'entier c tel que $\phi_p(P) = [c]_{E_a}(P)$ dans $E_a(\mathbb{Y})$.
5. deux possibilités :
   - si l'ordre de c dans $\mathbb{Z}/p^k\mathbb{Z}$ est égal à l'ordre de c dans $\mathbb{Z}/p\mathbb{Z}$, X_k est en fait défini dans $\mathbb{X}_1$ et nous y factorisons le polynôme $m(X)$ pour en déduire X_k en fonction de X_1. Puis nous prenons pour Y_k l'une des racines carrées de $X_k^3 + a_4X_k + a_6$ dans $\mathbb{Y}_1$. Nous posons ensuite $\mathbb{X}_k = \mathbb{X}_1$, $\mathbb{Y}_k = \mathbb{Y}_1$, $P_k = (X_k, Y_k)$ et nous posons $m_k(X) = X - X_k$;
   - sinon, nous posons $\mathbb{X}_k = \mathbb{X}$, $\mathbb{Y}_k = \mathbb{Y}$, $P_k = P$ et $m_k(X) = m(X)$.

Exemple : nous allons calculer la 5^3-torsion de la courbe E_a : $Y^2 = X^3 + \overline{241917}X + \overline{128207}$ dans $\mathbb{F}_{5^8} = \mathbb{F}_5[t]/(t^8+2)$ avec la notation $\tau_0 + \tau_1 5 + \cdots + \tau_7 5^7 = \tau_0 + \tau_1 t + \cdots + \tau_7 t^7$. Nous avons déjà $H_{E_a} = \overline{93184}$ et on vérifie que $r = 2$ et $\gamma^2 = \overline{23986}$. Nous posons donc

$$\mathbb{U}_1 = \mathbb{F}_q[U_1]/(U_1^2 - \gamma^2).$$

Une application du théorème 42 donne alors $X_1 = \overline{387840}$. Cela signifie que $\mathbb{X}_1 = \mathbb{F}_q$. Comme par ailleurs $\mathbb{U}_1$ est de degré 2, le polynôme $Y^2 - X_1^3 - \overline{241917}X_1 - \overline{128207}$ est irréductible et $\mathbb{Y}_1 = \mathbb{X}_1[Y_1]/(Y_1^2 + \overline{59969})$. Ainsi, $P_1 = (\overline{387840}, Y_1)$ est un point de 5-torsion. Nous trouvons alors $(X_1^q, Y_1^q) = -(X_1, Y_1)$ dans $E_a(\mathbb{Y}_1)$, donc $c = 4 \bmod 5$ et l'ordre de c modulo 5 est égal à 2.

Nous avons ensuite besoin de l'isogénie

$$\mathcal{I}_5 \ : (X, Y) \mapsto \left(\frac{g_5(X)}{h_5^2(X)}, Y \frac{k_5(X)}{h_5^3(X)} \right),$$

où nous calculons $h_5(X)$ à partir des abscisses des multiples de P_1,

$$h_5(X) = X^2 + \overline{263252}\, X + \overline{203861},$$

ainsi que $g_5(X)$ et $k_5(X)$ avec les formules de Vélu,

$$\begin{array}{rcl} g_5(X) & = & \overline{239118}\, X^5 + \overline{94276}\, X^4 + \overline{18634}\, X^3 + \overline{377379}\, X^2 + \overline{297583}\, X + \overline{241544}, \\ k_5(X) & = & \overline{275333}\, X^6 + \overline{71501}\, X^5 + \overline{32491}\, X^4 + \overline{370558}\, X^3 + \overline{179694}\, X^2 + \overline{93568}\, X \\ & & + \overline{194678}. \end{array}$$

Nous obtenons alors $\tilde{X}_1 = \overline{200420}$, d'où

$$m(X) = X^5 + \overline{362872}\, X^4 + \overline{140334}\, X^3 + \overline{366866}\, X^2 + \overline{296079}\, X + \overline{389630}.$$

Ainsi, en posant

$$\mathbb{X} = \mathbb{F}_q[X_2]/(m(X_2)) \text{ et } \mathbb{Y} = \mathbb{X}[Y_2]/(Y_2^2 - X_2^3 - \overline{241917}X_2 - \overline{128207}),$$

$P = (X_2, Y_2)$ est un point de 25-torsion à partir duquel nous trouvons $(X_2^q, Y_2^q) = -(X_2, Y_2)$. Donc $c = 24 \bmod 5^2$ et l'ordre de c modulo 5^2 est *aussi* égal à 2. Cela signifie que nous avons un "effondrement" et que $m(X)$ se factorise complètement dans $\mathbb{F}_q$, en l'occurrence

$$m(X) = \left(X + \overline{36319}\right)\left(X + \overline{1696}\right)\left(X + \overline{259349}\right)\left(X + \overline{314290}\right)\left(X + \overline{168278}\right).$$

Posant $\mathbb{X}_2 = \mathbb{X}_1$, $\mathbb{Y}_2 = \mathbb{Y}_1$, le point $P_2 = (\overline{60711}, \overline{386227}Y_1)$ est un point de 25-torsion.

L'abscisse d'un point de 125-torsion est la racine de $m(X) = g_5(X) - \tilde{X}_2 h_5^2(X)$, c'est-à-dire

$$m(X) = X^5 + \overline{224816}\, X^4 + \overline{381048}\, X^3 + \overline{240666}\, X^2 + \overline{333053}\, X + \overline{87754}.$$

Ainsi on a

$$\mathbb{X} = \mathbb{F}_q[X_3]/(m(X_3)) \text{ et } \mathbb{Y} = \mathbb{X}[Y_3]/(Y_3^2 - X_3^3 - \overline{241917}X_3 - \overline{128207}).$$

Par conséquent $P = (X_3, Y_3)$ est un point de 125-torsion à partir duquel nous trouvons $(X_3^q, Y_3^q) = -[51]_{E_a}(X_3, Y_3)$ et nous tirons $c = 74 \bmod 5^2$. Comme l'ordre de c modulo 5^3 est égal à 10, $m(X)$ est irréductible et nous posons $\mathbb{X}_3 = \mathbb{X}$, $\mathbb{Y}_3 = \mathbb{Y}$, $P_3 = P$ et $m_3(X) = m(X)$.

6.2.3 Deuxième algorithme

L'un des inconvénients, certes mineur pour le calcul des points de p^k-torsion mais qui devient prohibitif pour l'algorithme de Couveignes, est la factorisation à l'étape 5 des polynômes $m(X)$ dans $\mathbb{X}_1$ lorsque l'ordre de c dans $\mathbb{Z}/p^k\mathbb{Z}$ est égal à l'ordre de c dans $\mathbb{Z}/p\mathbb{Z}$. Le théorème 45 remplace la factorisation de ce polynôme par celui d'un polynôme de la forme $X^p - X - v_{k-1}$ pour lequel il est possible d'appliquer des techniques particulières de factorisation (cf. par exemple la proposition 24). Le théorème 45 nous permet ainsi de calculer des générateurs P_k dans une tour d'extensions d'Artin-Schreier $\mathbb{F}_q \subset \mathbb{U}_1 \subset \cdots \subset \mathbb{U}_k$.

Pour démarrer la récurrence, le corps fini $\mathbb{U}_1$ est obtenu comme précédemment avec les idées de Gunji. Nous calculons donc :

1. $\mathbb{U}_1$, une extension de plus petit degré r contenant $\gamma = \sqrt[p-1]{H_{E_a}}$, c'est-à-dire
$$\mathbb{U}_1 = \begin{cases} \mathbb{F}_q & \text{si } r = 1, \\ \mathbb{F}_q[U]/(U^r - \gamma^r) & \text{sinon.} \end{cases}$$
2. X_1, l'abscisse dans $\mathbb{U}_1$ d'un générateur P_1 de $E_a[p]$ avec le théorème 42.
3. Y_1, l'ordonnée de P_1 en factorisant dans $\mathbb{U}_1$ le polynôme $Y^2 - X_1^3 - a_4X_1 - a_6$.
4. c modulo p en cherchant comme au chapitre 3 l'entier c tel que $\phi_p(P_1) = [c]_{E_a}(P_1)$ dans $E_a(\mathbb{U}_1)$.

Avant de réaliser le calcul de la p^2-torsion, nous avons besoin pour exploiter le théorème 45 de :

1. $V_a(X)$, le quotient de $(X^3 + a_4X + a_6)^{\frac{p-1}{2}}$ par X^p.
2. $\mathcal{I}_p = (g_p/h_p^2, \epsilon Y k_p/h_p^3)$, la p-isogénie de E_a dans $E_{\tilde{a}}$ où le signe ϵ de l'ordonnée est déterminé par $\phi_p \circ \mathcal{I}_p = [p]_{E_a}$. Pour cela, nous prenons un point aléatoire P de $E_a(\mathbb{F}_q)$ et nous choisissons ϵ tel que $Y^p_{\mathcal{I}_p(P)} = Y_{[p]_{E_a}(P)}$.

Enfin nous obtenons $E_a[p^k]$ à partir de $E_a[p^{k-1}]$ ($k \in \mathbb{N}^*$) en calculant :

1. $v_{k-1} \in \mathbb{U}_{k-1}$, la constante de Voloch donnée par
$$v_{k-1}^p = \frac{Y_{k-1}V_a\left(X_{k-1}\right)}{H_{E_a}\gamma}. \tag{6.2}$$
2. $\mathbb{U}$, l'extension de Voloch, c'est-à-dire
$$\mathbb{U} = \mathbb{U}_{k-1}[U_k]/(U_k^p - U_k - v_{k-1}).$$
3. X_k, l'abscisse d'un point de p^k-torsion. Pour cela, nous savons que
$$\tilde{X}_{k-1} = \frac{g_p(X_k)}{h_p^2(X_k)}, \tilde{Y}_{k-1} = Y_k\frac{k_p(X_k)}{h_p^3(X_k)}, U_k = -2\frac{Y_k}{\tilde{c}_{E_a}\tilde{\gamma}}\frac{h_p'\left(X_k\right)}{h_p\left(X_k\right)}.$$
D'où nous déduisons X_k d'après le théorème 45 de l'égalité
$$X - X_k = \operatorname{pgcd}\left(\tilde{X}_{k-1}h_p^2(X) - g_p(X), U_k\tilde{c}_{E_a}\tilde{\gamma}k_p\left(X\right) + 2\tilde{Y}_{k-1}h_p^2(X)h_p'(X)\right), \tag{6.3}$$
puis Y_k de l'égalité
$$Y_k = -U_k\frac{\tilde{c}_{E_a}\tilde{\gamma}}{2}\frac{h_p\left(X_k\right)}{h_p'\left(X_k\right)}. \tag{6.4}$$
Le point $P = (X_k, Y_k)$ est alors un point de p^k-torsion.
4. c modulo p^k en cherchant comme au chapitre 3 l'entier c tel que $\phi_p(P) = [c]_{E_a}(P)$ dans $E_a(\mathbb{U})$.
5. deux possibilités :
   - si l'ordre de c dans $\mathbb{Z}/p^k\mathbb{Z}$ est égal à l'ordre de c dans $\mathbb{Z}/p\mathbb{Z}$, X_k est en fait défini dans $\mathbb{U}_1$ et nous posons $\mathbb{U}_k = \mathbb{U}_1$. Nous obtenons alors explicitement X_k et Y_k dans $\mathbb{U}_1$ en remplaçant U_k par l'une des racines du polynôme $X^p - X - v_{k-1}$ dans leurs expressions obtenues à l'aide des équations (6.3) et (6.4). Le générateur P_k recherché est alors égal à (X_k, Y_k) ;
   - sinon, nous posons $\mathbb{U}_k = \mathbb{U}$ et $P_k = P$.

Ici encore, nous ne savons que $\mathbb{U}_k = \mathbb{U}_1$ qu'après avoir calculé c modulo p^k à l'étape 5, ceci pour éviter le lourd calcul de $\operatorname{Tr}_{\mathbb{U}/\mathbb{F}_p}(v_k)$. Nous pouvons éventuellement le détecter auparavant lors du calcul d'un inverse nécessaire au calcul du pgcd ou de c modulo p^k. Dans ce cas, nous obtenons immédiatement un facteur de $U_k^p - U_k - v_{k-1}$ qu'il ne reste plus qu'à factoriser pour finalement obtenir X_k et Y_k dans $\mathbb{U}_1$.

Un exemple d'application de cet algorithme est donné dans la section 6.4.

Légère amélioration : à ce stade, notons qu'il est possible d'utiliser ces idées pour améliorer légèrement l'algorithme basé sur les isogénies en remplaçant la factorisation éventuelle des polynômes $m_k(X)$ de l'étape 5 par les polynômes $X^p - X - v_{k-1}$. Nous donnons ici les quelques modifications qui sont pour cela nécessaires.

Pour calculer $E_a[p]$, nous procédons comme précédemment. Nous obtenons ainsi deux extensions $\mathbb{X}_1$ et $\mathbb{Y}_1$ contenant respectivement l'abscisse X_1 et l'ordonnée Y_1 d'un générateur P_1 de $E_a[p]$. Puis nous terminons en recalculant γ, la racine $(p-1)$-ième de H_{E_a} dans $\mathbb{Y}_1$.

Les calculs du polynôme $V_a(X)$ et de l'isogénie $\mathcal{I}_p$ sont alors réalisés comme pour le deuxième algorithme. En supposant $E_a[p^{k-1}]$ connu, nous calculons $E_a[p^k]$ comme dans le premier algorithme à l'exception près du point 5 que nous remplaçons par :

5. deux possibilités :
   - si l'ordre de c dans $\mathbb{Z}/p^k\mathbb{Z}$ est égal à l'ordre de c dans $\mathbb{Z}/p\mathbb{Z}$, X_k est en fait défini dans $\mathbb{X}_1$ et nous déterminons la constante de Voloch v_{k-1} par la formule (6.2). Il ne reste plus qu'à obtenir l'une des racines U_k dans $\mathbb{U}_1$ du polynôme $X^p - X - v_{k-1}$ pour ensuite déduire X_k et Y_k par simple application des relations (6.3) et (6.4). Nous posons ensuite $\mathbb{X}_k = \mathbb{X}_1$, $\mathbb{Y}_k = \mathbb{Y}_1$, $P_k = (X_k, Y_k)$ et posons $m_k(X) = X - X_k$;
   - sinon, nous posons $\mathbb{X}_k = \mathbb{X}$, $\mathbb{Y}_k = \mathbb{Y}$, $P_k = P$ et $m_k(X) = m(X)$.

6.2.4 Efficacité des deux approches

Les calculs réalisés dans les extensions $\mathbb{X}_k$ sont plus rapides que ceux réalisés dans les extensions $\mathbb{U}_k$ avec des méthodes classiques de multiplication polynomiale rapide comme l'algorithme de Karatsuba ou celui par transformée de Fourier discrète (FFT, cf. [20]). Cependant une méthode de multiplication polynomiale adaptée aux extensions d'Artin-Schreier et de même complexité asymptotique que celle par FFT existe [19], ce qui rendrait finalement le deuxième algorithme plus attractif pour de grandes p^k-torsions.

6.3 Isomorphismes entre $E_a[p^k]$ et $E_b[p^k]$

Nous expliquons ici comment le théorème 45 nous permet de calculer aisément un isomorphisme entre $E_a[p^k]$ et $E_b[p^k]$ nécessaire à l'implantation de l'algorithme de Couveignes. Il apparaîtra que le coût majeur de l'algorithme est la factorisation de polynômes de la forme $X^p - X - v$ dans des extensions d'Artin-Schreier dont nous discutons ensuite la résolution.

6.3.1 Isomorphismes entre extensions d'Artin-Shreier

Une condition préalable à l'application de l'algorithme de Couveignes est de disposer d'un isomorphisme explicite entre $E_a[p^k]$ et $E_b[p^k]$. Autrement dit, il s'agit de trouver les coordonnées (X_{Q_k}, Y_{Q_k}) d'un générateur Q_k de $E_b[k]$ dans les extensions sur lesquelles sont définies $E_a[p^k]$: $\mathbb{X}_k$ et $\mathbb{Y}_k$ en ce qui concerne le premier algorithme et $\mathbb{U}_k$ pour le deuxième algorithme.

Une solution naïve, mais trop coûteuse, est bien entendu de factoriser le polynôme minimal de X_{Q_k} dans ces extensions. Il est préférable d'utiliser le théorème 45 puisque dans ce cas nous avons à factoriser un polynôme de la forme $X^p - X - v$. Nous décrivons l'algorithme obtenu seulement pour une torsion $E_a[p^k]$ définie dans les extensions $\mathbb{X}_k$ et $\mathbb{Y}_k$ du premier algorithme. Pour les extensions $\mathbb{U}_k$, l'algorithme est en fait plus simple et se dérive sans problème de celui que nous donnons.

Une petite difficulté est ici le calcul dans $\mathbb{Y}_k$ de l'équivalent sur E_b du coefficient de Voloch v_k, puisqu'il nécessite la connaissance de γ, $X_{Q_{k-1}}$ et $Y_{Q_{k-1}}$ dans $\mathbb{Y}_k$. Pour la lever, il suffit, en supposant ces valeurs connues dans $\mathbb{Y}_{k-1}$, de remplacer X_{k-1} et Y_{k-1} par leurs expressions en fonction de X_k et Y_k déduites des relations

$$\tilde{X}_{k-1} = \frac{g_p(X_k)}{h_p^2(X_k)} \text{ et } \tilde{Y}_{k-1} = Y_k \frac{k_p(X_k)}{h_p^3(X_k)}. \tag{6.5}$$

Dans le détail, l'algorithme procède comme suit pour déterminer $E_b[p^k]$. Pour $k = 1$, nous avons :

1. X_{Q_1}, l'abscisse dans $\mathbb{X}_1$ de Q_1 par le théorème 42.
2. Y_{Q_1}, l'ordonnée dans $\mathbb{Y}_1$ de Q_1 comme racine carrée de $X_{Q_1}^3 + b_4 X_{Q_1} + b_6$.
3. γ, la racine $(p-1)$-ième de $H_{E_b} = H_{E_a}$ dans $\mathbb{Y}_1$.

Les précalculs nécessaires à l'application du théorème 45 sont alors :

1. $V_b(X)$, le quotient de $(X^3 + b_4 X + b_6)^{\frac{p-1}{2}}$ par X^p.
2. $\mathcal{I}_{b,p} = (g_{b,p}/h_{b,p}^2, \epsilon Y k_{b,p}/h_{b,p}^3)$, la p-isogénie de E_b dans $E_{\tilde{b}}$ où le signe ϵ de l'ordonnée est déterminé par $\phi_p \circ \mathcal{I}_{b,p} = [p]_{E_b}$. Pour cela, nous prenons un point aléatoire Q de $E_b(\mathbb{F}_{p^n})$ et nous choisissons ϵ tel que $Y^p_{\mathcal{I}_{b,p}(Q)} = Y_{[p]_{E_b}(Q)}$.

Pour obtenir $E_b[p^k]$, nous calculons alors :

1. γ et $X_{Q_{k-1}}$, $Y_{Q_{k-1}}$ dans $\mathbb{Y}_k$ en remplaçant dans leurs expressions connues dans $\mathbb{Y}_{k-1}$, les variables X_{k-1} et Y_{k-1} par les relations (6.5) fonctions de X_k et Y_k.
2. $v_{b,k-1}$, la constante de Voloch déduite de l'égalité

$$v_{b,k-1}^p = \frac{Y_{Q_{k-1}} V_b\left(X_{Q_{k-1}}\right)}{H_{E_b}\gamma}.$$

3. u_k, une racine de $X^p - X - v_{b,k-1}$ dans $\mathbb{Y}_k$.
4. X_{Q_k}, l'abscisse d'un point de p^k-torsion. Pour cela, nous savons que

$$\tilde{X}_{Q_{k-1}} = \frac{g_{b,p}(X_{Q_k})}{h_{b,p}^2(X_{Q_k})}, \tilde{Y}_{Q_{k-1}} = Y_{Q_k}\frac{k_{b,p}(X_{Q_k})}{h_{b,p}^3(X_{Q_k})}, u_k = -2\frac{Y_{Q_k}}{\tilde{c}_{E_b}\tilde{\gamma}}\frac{h'\left(X_{Q_k}\right)}{h\left(X_{Q_k}\right)},$$

   d'où nous déduisons X_{Q_k} de l'expression

$$\operatorname{pgcd}\left(\tilde{X}_{Q_{k-1}} h_{b,p}^2(X_{Q_k}) - g_{b,p}(X_{Q_k}), u_k \tilde{c}_{E_b}\tilde{\gamma} k_{b,p}\left(X_{Q_k}\right) + 2\tilde{Y}_{Q_{k-1}} h_{b,p}^2(X_{Q_k}) h'_{b,p}(X_{Q_k})\right)$$

   puis Y_{Q_k} de la relation

$$Y_{Q_k} = -u_k \frac{\tilde{c}_{E_b}\tilde{\gamma}}{2}\frac{h_{b,p}\left(X_{Q_k}\right)}{h'_{b,p}\left(X_{Q_k}\right)}.$$

Un exemple d'application de cet algorithme est donné dans la section 6.4.

6.3.2 Résoudre $X^p - X = \delta$ dans une tour d'extensions d'Artin-Schreier

Le coût majeur du deuxième algorithme de Couveignes pour calculer une isogénie de degré ℓ entre deux courbes définies sur $\mathbb{F}_{p^n}$ est le calcul à l'étape 3 de l'algorithme d'une racine u_k du polynôme $X^p - X - v_{b,k-1}$ dans les extensions $\mathbb{Y}_k$ ou $\mathbb{U}_k$. Couveignes propose deux méthodes pour cela, desquelles nous déduisons le coût asymptotique de l'algorithme en fonction de ℓ, p et n. L'unité de mesure est la multiplication de deux éléments dans $\mathbb{F}_{p^n}$ supposée réalisée par une méthode classique, c'est-à-dire en $O(n^2 \log^2 p)$ opérations arithmétiques.

Méthode classique

La première méthode [31] découle immédiatement des méthodes classiques décrites au chapitre 5.

Principe : nous procédons en deux étapes.

1. Nous calculons l'inverse de la matrice à $nrp^{k-1} \times nrp^{k-1}$ coefficients dans $\mathbb{F}_p$ de l'application $\mathbb{F}_p$-linéaire $X \mapsto X^p - X$ définie à partir de $\mathbb{U}_k$ ou $\mathbb{Y}_k$ (rappelons que $r = [\mathbb{U}_1 : \mathbb{F}_q]$).
2. Nous appliquons cet inverse au coefficient $v_{b,k-1}$ considéré comme un vecteur à composantes dans $\mathbb{F}_p$ pour obtenir la racine u_k de $X^p - X = v_{b,k-1}$.

Complexité : en supposant r de l'ordre de grandeur de p, la complexité de l'étape 1 est asymptotiquement de $O((np^k)^3)$ multiplications dans $\mathbb{F}_p$ pour chaque entier $k \leqslant K \simeq \log_p \ell$, soit un total de $O\left(n\ell^3\right)$ multiplications dans $\mathbb{F}_{p^n}$ et un stockage en $O(n^2\ell^2 \log p)$ bits. De même, nous déduisons que le coût asymptotique de l'étape 2 est $O\left(\ell^2\right)$ multiplications dans $\mathbb{F}_{p^n}$.

Dans l'algorithme `SEA`, les isogénies que nous avons à calculer sont de degré au plus $O(n \log p)$. Une astuce classique consiste à calculer au préalable et stocker les inverses des matrices nécessaires (elles ne dépendent que de E_a) au prix de $O(n^4)$ multiplications dans $\mathbb{F}_{p^n}$ et d'un stockage en $O(n^4)$. Le calcul des isogénies de degré ℓ nécessaires requiert alors un coût de $O(n^3)$ à p fixé.

En pratique cette méthode est inapplicable puisque, par exemple, pour calculer une isogénie de degré 500 dans $\mathbb{F}_{2^{1000}}$ elle nécessiterait un stockage de l'ordre du gigaoctet.

Tirer parti des extensions d'Artin-Schreier

Pour $E_a[k]$ calculé dans des extensions d'Artin-Schreier $\mathbb{U}_k$, Couveignes a récemment proposé une alternative de complexité en $O(n\ell^2)$ ou en $O(n\ell \log \ell)$ avec une multiplication polynomiale par FFT [32]. Couveignes exhibe pour cela un procédé astucieux pour remplacer la résolution d'une équation du type $X^p - X = \delta_k$ dans $\mathbb{U}_k$ où $\delta_k \in \mathbb{U}_k$, par celle de $X^p - X = \delta_{k-1}$ dans $\mathbb{U}_{k-1}$ avec δ_{k-1} défini dans $\mathbb{U}_{k-1}$.

L'algorithme final consiste à appliquer $(K-1)$ fois ce procédé à partir de $X^p - X - v_{b,K-1}$ pour ne plus avoir qu'à résoudre une équation du type $X^p - X = \delta_1$ dans $\mathbb{U}_1$ avec $\delta_1 \in \mathbb{U}_1$. En inversant les changements de variables induits par les $K-1$ transformations utilisées, ce qui est immédiat, nous sommes alors en mesure de retrouver une racine de $X^p - X = v_{b,k-1}$ dans $\mathbb{U}_k$.

Nous décrivons maintenant précisément ce procédé et nous en déduisons la complexité de l'algorithme de Couveignes.

Principe : le but est de remplacer le calcul d'une racine dans $\mathbb{U}_k$ de l'équation $X^p - X = \delta_k$ où $\delta_k \in \mathbb{U}_k$ par celui d'une racine dans $\mathbb{U}_{k-1}$ de l'équation $X^p - X = \delta_{k-1}$ où $\delta_{k-1} \in \mathbb{U}_{k-1}$. Dans un premier temps, nous calculons la pseudo-trace

$$\beta_k = \mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{F}_p}(\delta_k) = \sum_{i=0}^{[\mathbb{U}_{k-1}:\mathbb{F}_p]-1} \delta_k^{p^i}.$$

Notons que, comme $\delta_k \in \mathbb{U}_k$, nous avons a priori $\beta_k \in \mathbb{U}_k$. Nous nous intéressons alors à l'équation $X^{p^{nrp^{k-1}}} - X = \beta_k$. Clairement, puisque

$$X^{p^{nrp^{k-1}}} - X = (X^p - X)^{p^{nrp^{k-1}-1}} + \cdots + (X^p - X) = \delta_k^{p^{nrp^{k-1}-1}} + \cdots + \delta_k = \beta_k,$$

une racine de $X^p - X = \delta_k$ est aussi racine de $X^{p^{nrp^{k-1}}} - X = \beta_k$.

Or $X^{p^{nrp^{k-1}}} - X$ est une application $\mathbb{U}_{k-1}$-linéaire, nous pouvons donc construire sa matrice M de taille $p \times p$ dans la base polynomiale $\left\{1, U_k, \ldots U_k^{p-1}\right\}$. Nous calculons alors l'inverse de cette matrice que nous appliquons à β_k considéré comme vecteur à p composantes dans $\mathbb{U}_{k-1}$. Soit γ_k la racine dans $\mathbb{U}_k$ de $X^{p^{nrp^{k-1}}} - X = \beta_k$ obtenue ainsi. Il ne reste plus qu'à écrire une solution de $X^p - X = \delta_k$ sous la forme $Z + \gamma_k$ pour avoir $Z^p - Z = \delta_k - \gamma_k^p + \gamma_k$. Nous posons $\delta_{k-1} = \delta_k - \gamma_k^p + \gamma_k$, nous avons alors $\delta_{k-1} \in \mathbb{U}_{k-1}$ puisque

$$(\delta_k - \gamma_k^p + \gamma_k)^{p^{nrp^{k-1}}} - (\delta_k - \gamma_k^p + \gamma_k) = \left(\delta_k^{p^{nrp^{k-1}}} - \delta_k\right) - \mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{F}_p}\left(\delta_k^p - \delta_k\right) = 0.$$

De plus $\mathrm{Tr}_{\mathbb{U}_{k-1}/\mathbb{F}_p}(\delta_{k-1}) = 0$ puisque

$$\mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{F}_p}(\delta_k - \gamma_k^p + \gamma_k) = \mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{F}_p}(\delta_k) - \left(\gamma_k^{p^{nrp^{k-1}}} - \gamma_k\right) = 0.$$

Complexité : le calcul de la complexité du procédé précédent est assez lourd (cf [32]) ; nous en rappelons simplement les grandes lignes.

Soit K l'entier défini par $p^{K-1} \leqslant 4\ell+5 < p^K$. Une première application du procédé précédent remplace le calcul d'une racine dans $\mathbb{U}_K$ de $X^p - X - v_{b,K-1}$ avec $v_{b,K-1} \in \mathbb{U}_{K-1}$ par celui d'une racine dans $\mathbb{U}_{K-1}$ de $X^p - X - \delta_{K-1}$ avec $\delta_{K-1} \in \mathbb{U}_{K-1}$. En nombre de multiplications sur $\mathbb{F}_{p^n}$, la complexité des calculs est :

- $O(np^3 \log^2 p)$ pour β_K. En effet, nous avons dans toute extension d'Artin-Schreier $\mathbb{U}_k$ $(k>1)$,

$$\forall (u_0, \cdots, u_{p-1}) \in \mathbb{U}_{k-1}^p, \mathrm{Tr}_{\mathbb{U}_k/\mathbb{U}_{k-1}}\left(\sum_{i=0}^{p-1} u_i U_k^i\right) = -u_{p-1}. \tag{6.6}$$

  Par composition des traces,

$$\mathrm{Tr}_{\mathbb{U}_{K-1}/\mathbb{F}_p}(v_{b,K-1}) = \mathrm{Tr}_{\mathbb{U}_1/\mathbb{F}_p} \circ \cdots \circ \mathrm{Tr}_{\mathbb{U}_{K-1}/\mathbb{U}_{K-2}}(v_{b,K-1}), \tag{6.7}$$

  et le seul coût à considérer est celui de $\mathrm{Tr}_{\mathbb{U}_1/\mathbb{F}_p}$.
- $O(p^2/n^2)$ pour construire la matrice M de $X \mapsto X^{p^{nrp^{K-1}}} - X$. En effet, nous avons dans toute extension d'Artin-Schreier $\mathbb{U}_k$,

$$\forall h \in \mathbb{N},\ U_k^{p^h} = U_k + \mathrm{T}_h(v_{k-1}) \text{ où } \mathrm{T}_h(v_{k-1}) = \sum_{i=0}^{h-1} v_{k-1}^{p^i}. \tag{6.8}$$

  En particulier,

$$U_K^{p^{nrp^{K-1}}} = U_K + \mathrm{Tr}_{\mathbb{U}_{K-1}/\mathbb{F}_p}(v_{K-1}). \tag{6.9}$$

  La matrice M est donc à coefficients dans $\mathbb{F}_p$ et pour la construire il suffit d'écrire

$$\forall i \in \{0, \cdots, p-1\}, U_K^{ip^{nrp^{K-1}}} = \left(U_K + \mathrm{Tr}_{\mathbb{U}_{K-1}/\mathbb{F}_p}(v_{K-1})\right)^i.$$

  Nous négligeons le temps d'évaluation de $\mathrm{Tr}_{\mathbb{U}_{K-1}/\mathbb{F}_p}(v_{K-1})$ que nous précalculons.
- $O(p^3/n^2)$ pour inverser la matrice M ($O(p^3)$ dans $\mathbb{F}_p$).
- $O(p^2\ell/n)$ pour appliquer la matrice M^{-1} à β_K et obtenir γ_K.
- $O(\ell^2)$ pour calculer γ_K^p et en déduire δ_{K-1}.

Pour $1 < k < K$, l'application du procédé précédent remplace le calcul d'une racine dans $\mathbb{U}_k$ de $X^p - X - \delta_k$ avec $\delta_k \in \mathbb{U}_k$ par celui d'une racine dans $\mathbb{U}_{k-1}$ de $X^p - X - \delta_{k-1}$ avec $\delta_{k-1} \in \mathbb{U}_{k-1}$. En nombre de multiplications sur $\mathbb{F}_{p^n}$, la complexité des calculs est :

- $O(np^{2k+\epsilon})$ pour β_k. L'obtention de cette complexité est le point délicat puisque, comme $\delta_k \in \mathbb{U}_k$, la relation (6.6) ne s'applique pas. Par contre, la composition des traces (6.7) demeure.
  Concentrons-nous sur le calcul de $\mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{U}_{k-2}}(\delta_k)$. Le coût en est p fois le coût d'une exponentiation à la puissance $p^{np^{k-2}}$ dans $\mathbb{U}_k$. Une fois précalculé $U_k^{ip^{np^{k-2}}}$ pour i variant de 0 à $p-1$ (ce qui peut être fait en $O(p^{2k})$ à l'aide de l'équation (6.8), stockage en $O(np^{k+2}\log p)$), il n'est pas difficile de se rendre compte que le coût d'une exponentiation à la puissance $p^{np^{k-2}}$ est à son tour p fois celui d'une exponentiation par $p^{np^{k-2}}$ dans $\mathbb{U}_{k-1}$ plus p^2 fois celui d'un produit dans $\mathbb{U}_{k-1}$. Par un raisonnement analogue pour l'exponentiation dans $\mathbb{U}_{k-1}$, nous déduisons finalement que le coût de $\mathrm{T}_{\mathbb{U}_{k-1}/\mathbb{U}_{k-2}}(\delta_k)$ est p^3 fois celui d'une exponentiation par $p^{np^{k-2}}$ dans $\mathbb{U}_{k-2}$ auquel s'ajoute p^4 fois celui d'un produit dans $\mathbb{U}_{k-4}$ et p^3 fois celui d'un produit dans $\mathbb{U}_{k-1}$. Comme $\mathbb{U}_{k-2}$ est de cardinalité $p^{np^{k-2}}$, le coût de cette dernière exponentiation est donc nul, ce qui conduit à une complexité de $O(p^{2k+1})$.
  Dans ces conditions, le calcul prépondérant est celui de $\mathrm{T}_{\mathbb{F}_{p^n}/\mathbb{F}_p}$ appliqué à un élément de $\mathbb{U}_k$. Ce calcul est de complexité $O(np^{2k})$ au prix du précalcul aussi en $O(np^{2k})$ de t^{ip} pour i variant de 0 à $n-1$, t, élément générateur de $\mathbb{F}_{p^n}$.
- $O(p^2/n^2)$ pour obtenir la matrice M de $X \mapsto X^{p^{nrp^{k-1}}} - X$ par un raisonnement analogue au cas $k = K$.

– $O(p^3/n^2)$ pour inverser M.
– $O(p^{k+1}/n)$ pour appliquer la matrice M^{-1} à β_k et obtenir γ_k.
– $O(p^{2k})$ multiplications dans $\mathbb{F}_{p^n}$ pour calculer γ_k^p et en déduire δ_{k-1}.

La complexité totale de l'algorithme pour calculer une isogénie de degré ℓ est donc à p fixé de $O(n\ell^2)$ pour un stockage en $O(\ell^2)$ puisque le coût majeur en est celui de β_{K-1}. En utilisant de plus une multiplication polynomiale par FFT, nous pouvons faire chuter cette complexité à $O(n\ell^{1+\epsilon})$.

6.4 Résultats

Pour illustrer les constructions précédentes, nous donnons en exemple le calcul d'une isogénie de degré 13 dans $\mathbb{F}_{5^3}$. Les données numériques que nous fournissons sont extraites de l'exécution du programme écrit en langage `C` que nous décrivons ensuite.

6.4.1 Exemple

Dans

$$\mathbb{F}_{5^3} = \mathbb{F}_5[t]/(t^3 + t + 1),$$

nous nous proposons de calculer l'abscisse $\dfrac{g_\ell(X)}{h_\ell^2(X)}$ d'une isogénie $\mathcal{I}_\ell$ de degré $\ell = 13$ entre les courbes

$$E_a \ : Y^2 = X^3 + X + \overline{6} \text{ et } E_b \ : Y^2 = X^3 + X + \overline{107}.$$

Tout d'abord $4\ell + 5 = 57$ impose de calculer des générateurs de $E_a[5^3]$ et $E_b[5^3]$ dans une extension d'Artin-Schreier $\mathbb{U}_3$ avant de trouver $\mathcal{I}_\ell$ par interpolation. Tout au long de cet exemple, nous utiliserons la notation $\overline{\tau_0 + \tau_1 5 + \tau_2 5^2} = \tau_0 + \tau_1 t + \tau_2 t^2$.

Calcul de $E_a[5^3]$

Ce calcul se décompose ici en trois phases correspondant au calcul des générateurs P_1, P_2 et P_3.

Calcul d'un générateur P_1 de $E_a[5]$: nous avons $H_{E_a} = 2$ qui n'admet pas de racine $(p-1)$-ième dans $\mathbb{F}_{5^3}$, nous posons donc

$$\mathbb{U}_1 = \mathbb{F}_{5^3}[U_1]/(U_1^4 - 2)$$

et $\gamma = U_1$. Une application du théorème 42 donne alors

$$X_1 = \overline{123}\, U_1^2 + \overline{110}.$$

Puis la factorisation de $Y^2 - X_1^3 - X_1 - \overline{6}$ dans $\mathbb{U}_1$ conduit à

$$Y_1 = \overline{91}\, U_1^3 + \overline{49}\, U_1.$$

Enfin, nous trouvons $(X_1^q, Y_1^q) = -[2]_{E_a}(X_1, Y_1)$ donc $c = 3 \bmod 5$ et l'ordre de c est 4 modulo 5.

Calcul d'un générateur P_2 de $E_a[5^2]$: nous effectuons tout d'abord les précalculs, à savoir

$$V_a(X) = X \text{ et } \mathcal{I}_5 \ : (X, Y) \mapsto \left(\frac{g_5(X)}{h_5^2(X)}, Y \frac{k_5(X)}{h_5^3(X)} \right),$$

où nous calculons $h_5(X)$ à partir des abscisses des multiples de P_1,

$$h_5(X) = X^2 + \overline{55}\, X + \overline{5},$$

et ensuite $g_5(X)$ et $k_5(X)$ avec les formules de Vélu,

$$\begin{aligned} g_5(X) &= \overline{4}\, X^5 + \overline{40}\, X^4 + \overline{34}\, X^3 + \overline{95}\, X^2 + \overline{102}\, X + \overline{82}, \\ k_5(X) &= \overline{2}\, X^6 + \overline{55}\, X^5 + \overline{91}\, X^4 + \overline{109}\, X^3 + \overline{88}\, X^2 + \overline{43}. \end{aligned}$$

Attaquons maintenant le calcul de $\mathbb{U}_2$. Pour cela nous avons

$$v_1 = \left(\frac{Y_1 V_a(X_1)}{H_{E_a}\gamma}\right)^{1/5} = \overline{19}\,U_1^2 + \overline{70}.$$

Nous posons

$$\mathbb{U}_2 = \mathbb{U}_1[U_2]/(U_2^p - U_2 - v_1).$$

Pour trouver X_2, nous calculons alors

$$\operatorname{pgcd}\left(\tilde{X}_1 h_5^2(X_2) - g_5(X_2), U_2 \tilde{c}_{E_a}\tilde{\gamma} k_5(X_2) + 2\tilde{Y}_1 h_5^2(X_2) h_5'(X_2)\right)$$

pour obtenir un polynôme de degré 1, $X - X_2$, avec

$$X_2 = \left(\overline{51}\,U_1^2 + \overline{8}\right)U_2^4 + \left(\overline{28}\,U_1^2 + \overline{5}\right)U_2^3 + \left(\overline{5}\,U_1^2 + \overline{36}\right)U_2^2 + \left(\overline{11}\,U_1^2 + \overline{48}\right)U_2 + \overline{20}\,U_1^2 + \overline{100}.$$

Nous trouvons ensuite Y_2 par

$$Y_2 = -U_2\frac{\tilde{c}_{E_a}\tilde{\gamma}}{2}\frac{h(X_2)}{h'(X_2)},$$

ce qui donne ici,

$$Y_2 = \left(\overline{28}\,U_1^3 + \overline{5}\,U_1\right)U_2^4 + \left(\overline{69}\,U_1^3 + \overline{36}\,U_1\right)U_2^3 + \left(\overline{11}\,U_1^3 + \overline{48}\,U_1\right)U_2^2 + \left(\overline{6}\,U_1^3 + \overline{99}\,U_1\right)U_2 + \overline{9}\,U_1^3 + \overline{53}\,U_1.$$

Enfin, nous avons $(X_2^q, Y_2^q) = [8]_{E_a}(X_2, Y_2)$, donc $c = 8 \bmod 5^2$ et l'ordre de c modulo 5^2 est égal à 20.

Calcul d'un générateur P_3 de $E_a[5^3]$: nous obtenons, de la même façon qu'avec $E_a[5^2]$, le coefficient de Voloch

$$v_2 = \left(\overline{31} + \overline{59}\,U_1^2\right)U_2^4 + \left(\overline{31} + \overline{112}\,U_1^2\right)U_2^3 + \left(\overline{76} + \overline{7}\,U_1^2\right)U_2^2 + \left(\overline{104} + \overline{43}\,U_1^2\right)U_2 + \overline{62}\,U_1^2 + \overline{89}.$$

D'où nous posons

$$\mathbb{U}_3 = \mathbb{U}_2[U_3]/(U_3^p - U_3 - v_2),$$

et nous déduisons des formules de Voloch X_3 qui est égal ici à

$$\begin{aligned}&\left(\left(\overline{84}\,U_1^2 + \overline{42}\right)U_2^4 + \left(\overline{83}\,U_1^2 + \overline{39}\right)U_2^3 + \left(\overline{75}\,U_1^2 + \overline{55}\right)U_2^2 + \left(\overline{92}\,U_1^2 + \overline{7}\right)U_2 + \overline{93}\,U_1^2 + \overline{101}\right)U_3^4\\ &\quad + \left(\left(\overline{117}\,U_1^2 + \overline{44}\right)U_2^4 + \left(\overline{93}\,U_1^2 + \overline{10}\right)U_2^3 + \left(\overline{103}\,U_1^2 + \overline{115}\right)U_2^2 + \left(\overline{87}\,U_1^2 + \overline{43}\right)U_2 + \overline{107}\,U_1^2 + \overline{20}\right)U_3^3\\ &\quad + \left(\left(\overline{112}\,U_1^2 + \overline{88}\right)U_2^4 + \left(\overline{110}\,U_1^2 + \overline{84}\right)U_2^3 + \left(\overline{111}\,U_1^2 + \overline{100}\right)U_2^2 + \left(\overline{90}\,U_1^2 + \overline{33}\right)U_2 + \overline{26}\,U_1^2 + \overline{34}\right)U_3^2\\ &\quad + \left(\left(\overline{107}\,U_1^2 + \overline{18}\right)U_2^4 + \left(\overline{81}\,U_1^2 + \overline{115}\right)U_2^3 + \left(\overline{96}\,U_1^2 + \overline{58}\right)U_2^2 + \left(\overline{17}\,U_1^2 + \overline{76}\right)U_2 + \overline{33}\,U_1^2 + \overline{55}\right)U_3\\ &\quad + \left(\overline{3}\,U_1^2 + \overline{3}\right)U_2^4 + \left(\overline{60}\,U_1^2 + \overline{21}\right)U_2^3 + \left(\overline{66}\,U_1^2 + \overline{53}\right)U_2^2 + \left(\overline{55}\,U_1^2 + \overline{72}\right)U_2 + \overline{61}\,U_1^2 + \overline{9}\end{aligned}$$

et Y_3 que nous omettons. Enfin, nous trouvons $(X_3^q, Y_3^q) = [108]_{E_a}(X_3, Y_3)$, donc $c = 108 \bmod 5^3$ et l'ordre de c modulo 5^3 est égal à 100.

Calcul de $E_b[5^3]$

Nous devons ensuite calculer $E_b[5^3]$ dans la tour d'extensions d'Artin-Schreier

$$\mathbb{F}_p \subset \mathbb{F}_{p^n} \subset \mathbb{U}_1 \subset \mathbb{U}_2 \subset \mathbb{U}_3.$$

Plus succinctement que pour $E_a[5^3]$, nous procédons comme suit.

Calcul d'un générateur Q_1 de $E_b[5]$: en appliquant les idées de Gunji dans $\mathbb{U}_1$, nous trouvons $X_{Q_1} = \overline{84}\,U_1^2 + \overline{24}$. Puis en factorisant $Y^2 - X_{Q_1}^3 - X_{Q_1} - \overline{107}$ dans $\mathbb{U}_1$, nous posons $Y_{Q_1} = \overline{38}\,U_1^3 + \overline{111}\,U_1$.

Calcul d'un générateur Q_2 de $E_b[5^2]$: tout d'abord, nous effectuons les précalculs, à savoir

$$V_b(X) = X \text{ et } \mathcal{I}_{b,5} \ : (X,Y) \mapsto \left(\frac{g_{b,5}(X)}{h_{b,5}^2(X)}, Y \frac{k_{b,5}(X)}{h_{b,5}^3(X)} \right),$$

où nous calculons $h_{b,5}(X)$ à partir des abscisses des multiples de P_1,

$$h_{b,5}(X) = X^2 + \overline{12}\,X + \overline{106},$$

et ensuite $g_{b,5}(X)$ et $k_{b,5}(X)$ avec les formules de Vélu,

$$\begin{aligned} g_{b,5}(X) &= \overline{4}\,X^5 + \overline{6}\,X^4 + \overline{73}\,X^3 + \overline{18}\,X^2 + \overline{64}\,X + \overline{77}, \\ k_{b,5}(X) &= \overline{2}\,X^6 + \overline{12}\,X^5 + \overline{38}\,X^4 + \overline{42}\,X^3 + \overline{59}\,X^2 + \overline{9}. \end{aligned}$$

Comme indiqué dans la section précédente, nous avons ensuite à trouver une racine u_2 dans $\mathbb{U}_2$ du polynôme $X^p - X = v_{b,1}$ avec ici $v_{b,1} = \overline{67}\,U_1^2 + \overline{50}$. Nous prenons, par exemple, $u_2 = U_2 + \overline{121}\,U_1^2 + \overline{80}$, et les relations (6.1) mènent alors à

$$\begin{aligned} X_{Q_2} = \left(\overline{32}\,U_1^2 + \overline{109}\right) U_2^4 + \left(\overline{9}\,U_1^2 + \overline{53}\right) U_2^3 + \left(\overline{93}\,U_1^2 + \overline{115}\right) U_2^2 \\ + \left(\overline{15}\,U_1^2 + \overline{99}\right) U_2 + \overline{27}\,U_1^2 + \overline{59}, \end{aligned}$$

puis à

$$\begin{aligned} Y_{Q_2} = \left(\overline{64}\,U_1^3 + \overline{49}\,U_1\right) U_2^4 + \left(\overline{42}\,U_1^3 + \overline{73}\,U_1\right) U_2^3 \\ + \left(\overline{101}\,U_1^3 + \overline{73}\,U_1\right) U_2^2 + \left(\overline{42}\,U_1^3 + \overline{32}\,U_1\right) U_2 + \overline{56}\,U_1^3 + \overline{27}\,U_1. \end{aligned}$$

Calcul d'un générateur Q_3 de $E_b[5^3]$: enfin nous avons à trouver une racine u_3 dans $\mathbb{U}_3$ du polynôme $X^p - X = v_{b,2}$ avec ici

$$\begin{aligned} v_{b,2} = \left(\overline{70} + \overline{11}\,U_1^2\right) U_2^4 + \left(\overline{11}\,U_1^2 + \overline{18}\right) U_2^3 \\ + \left(\overline{89}\,U_1^2 + \overline{2}\right) U_2^2 + \left(\overline{42}\,U_1^2 + \overline{62}\right) U_2 + \overline{30}\,U_1^2 + \overline{49}. \end{aligned}$$

Nous prenons, par exemple,

$$\begin{aligned} u_3 = U_3 + \left(\overline{77}\,U_1^2 + \overline{31}\right) U_2^4 + \left(\overline{28}\,U_1^2 + \overline{57}\right) U_2^3 + \left(\overline{53}\,U_1^2 + \overline{50}\right) U_2^2 \\ + \left(\overline{122}\,U_1^2 + \overline{36}\right) U_2 + \overline{64}\,U_1^2 + \overline{39}, \end{aligned}$$

et les relations (6.1) mènent alors à X_{Q_3} et Y_{Q_3} que nous omettons.

Calcul de $\mathcal{I}_\ell$

À partir par exemple de P_3, il n'est pas difficile d'obtenir le polynôme $\prod_{i=1}^{62}(X - X_{iP_3})$. Dans notre exemple, celui-ci est égal à

$$\begin{aligned} &X^{62} + \overline{12}\,X^{61} + \overline{71}\,X^{60} + \overline{51}\,X^{59} + \overline{13}\,X^{58} + \overline{8}\,X^{57} + \overline{97}\,X^{56} + \overline{64}\,X^{55} + \overline{76}\,X^{54} + \overline{17}\,X^{53} + \overline{63}\,X^{52} \\ &\quad + \overline{99}\,X^{51} + \overline{121}\,X^{50} + \overline{44}\,X^{49} + \overline{89}\,X^{48} + \overline{123}\,X^{47} + \overline{15}\,X^{46} + \overline{75}\,X^{45} + \overline{78}\,X^{44} + \overline{34}\,X^{43} + \overline{97}\,X^{42} \\ &\quad + \overline{113}\,X^{41} + \overline{74}\,X^{40} + \overline{17}\,X^{39} + \overline{26}\,X^{38} + \overline{107}\,X^{37} + \overline{46}\,X^{36} + \overline{101}\,X^{35} + \overline{13}\,X^{34} + \overline{2}\,X^{33} + \overline{119}\,X^{32} \\ &\quad + \overline{43}\,X^{31} + \overline{54}\,X^{30} + \overline{57}\,X^{29} + \overline{21}\,X^{28} + \overline{67}\,X^{27} + \overline{23}\,X^{26} + \overline{120}\,X^{25} + \overline{65}\,X^{24} + \overline{65}\,X^{23} + \overline{29}\,X^{22} \\ &\quad + \overline{105}\,X^{21} + \overline{46}\,X^{20} + \overline{29}\,X^{19} + \overline{123}\,X^{18} + \overline{70}\,X^{17} + \overline{23}\,X^{16} + \overline{96}\,X^{15} + \overline{104}\,X^{14} + X^{13} + \overline{24}\,X^{12} \\ &\quad + \overline{111}\,X^{11} + \overline{53}\,X^{10} + \overline{16}\,X^9 + \overline{48}\,X^8 + \overline{110}\,X^7 + \overline{11}\,X^6 + \overline{40}\,X^5 + \overline{123}\,X^4 + \overline{89}\,X^3 + \overline{92}\,X^2 + \overline{120}\,X + \overline{48}. \end{aligned}$$

Il ne reste plus alors qu'à trouver l'entier λ tel que le polynôme $A(X)$ donné par

$$\sum_{i=1}^{62} X_{i\lambda Q_3} \prod_{j\neq i} \frac{X - X_{jP_3}}{X_{iP_3} - X_{jP_3}}.$$

soit égal à $\dfrac{g_{13}(X)}{h_{13}^2(X)}$ modulo $\prod_{i=1}^{62}(X - X_{iP_3})$. Ceci se produit ici pour $\lambda = 7$ et conduit finalement à

$$\begin{aligned} h_{13}(X) &= X^6 + \overline{72}\,X^5 + \overline{84}\,X^4 + \overline{54}\,X^3 + \overline{92}\,X^2 + \overline{45}\,X + \overline{112}, \\ g_{13}(X) &= X^{13} + \overline{119}\,X^{12} + \overline{13}\,X^{11} + \overline{48}\,X^{10} + \overline{104}\,X^9 + \overline{10}\,X^8 + \overline{112}\,X^7 + \overline{57}\,X^6 \\ &\quad + \overline{124}\,X^5 + \overline{93}\,X^4 + \overline{39}\,X^3 + \overline{53}\,X^2 + \overline{117}\,X + \overline{86}. \end{aligned}$$

6.4.2 Quelques temps de calcul

Nous avons programmé le schéma général de l'algorithme de Couveignes pour des courbes elliptiques définies dans des corps finis de caractéristique supérieure à cinq. L'implantation a été réalisée en langage `C` grâce à la librairie `ZEN` (cf. chapitre 10).

Depuis les premières formulations de Couveignes jusqu'aux développements que nous décrivons ici, les routines que nous avons programmées ont fortement évolué, à l'image de nos tâtonnements pour appréhender dans leur ensemble les constructions mathématiques complexes utilisées. Ainsi, si dans une première implantation nous nous étions placés dans des extensions d'Artin-Schreier, nous avons, dans un deuxième temps, programmé les calculs dans des extensions de grands degrés sur $\mathbb{F}_{p^n}$, celles obtenues par cycles d'isogénies, puisque nous disposons dans la librairie `ZEN` d'une multiplication polynomiale par Karatsuba efficace.

Les nouvelles idées de Couveignes pour résoudre des équations de la forme $X^p - X = \delta$ dans des extensions d'Artin-Schreier rendent clairement caduque cette deuxième implantation. En ce qui concerne la première implantation, au moins deux aspects restent à optimiser :

- la résolution astucieuse de $X^p - X = \delta$ dans des extensions d'Artin-Schreier comme expliqué dans [32] et à la section 6.3.
- le remplacement dans la dernière phase de l'algorithme d'une interpolation naïve avec les formules de Lagrange par une utilisation du théorème chinois comme expliqué par Couveignes [31].

Dans ces conditions, notre implantation s'apparente plus à une "maquette" ou un "jouet" permettant de tester de nouvelles idées. En conséquence, les quelques temps que nous donnons sont à prendre avec la plus grande précaution et seront certainement sujets à de fortes diminutions après optimisation !

Précisément, nous donnons dans le tableau 6.1 : $\mathbb{F}_{p^n}$, le corps de définition des courbes isogènes, ℓ, le degré de l'isogénie recherchée, K, le plus grand entier pour lequel nous avons à calculer $E_a[p^K]$ et $E_b[p^K]$, v_{k-1}, le temps de calcul des coefficients de Voloch sur E_a par la relation (6.2), P_k le temps de calcul des générateurs P_k pour $1 \leqslant k \leqslant K$ avec les relations (6.3) et (6.4), c mod p^k le temps de détermination de c modulo p^k, $v_{b,k-1}$, le temps de calcul des coefficients de Voloch sur E_b par la relation (6.2), $X^p - X$, le temps pour trouver une racine dans $\mathbb{U}_k$ de $X^p - X = v_{b,k-1}$, Q_k le temps de calcul des générateurs Q_k pour $1 \leqslant k \leqslant K$ avec les relations (6.3) et (6.4), λ l'entier tel que $\mathcal{I}_\ell(P_k) = [\lambda]_{E_b}(Q_k)$, le temps nécessaire à la phase d'interpolation et enfin Tot, le temps total. Tous ces temps sont exprimés en secondes.

Comme prévu, il apparaît nettement dans ce tableau que notre façon naïve de réaliser l'interpolation ou le calcul d'une racine de $X^p - X = v_{b,k-1}$ est désastreuse (lorsque le symbole ∞ est utilisé dans le tableau, c'est que nous n'avons pas pu aboutir). Par contre, ce qui nous semble plus inquiétant est le temps mis en jeu par les relations (6.3) et (6.4) pour P_k et Q_k, surtout lorsque p augmente légèrement. Ce temps est important vis-à-vis de celui nécessaire au calcul d'isogénies dans des corps finis de caractéristique 2 (cf. chapitre 7) ou dans des corps finis de grande caractéristique (cf. chapitre 4). Or, si ce n'est l'introduction de routine de multiplication par FFT, il nous paraît difficile d'optimiser plus cette partie. La prudence reste donc de mise quant à la réelle efficacité de cet algorithme en pratique.

Corps fini	ℓ	K	$E_a[p^K]$			$E_b[p^K]$			Interpolation		Tot
			v_{k-1}	P_k	$c \bmod p^k$	$v_{b,k-1}$	$X^p - X$	Q_k	λ	temps	
$\mathbb{F}_{5^3}$	13	3	0	1	2	0	28	1	7	104	140
$\mathbb{F}_{5^6}$	71	4	5	17	25	6	1519	17	∞	∞	∞
$\mathbb{F}_{7^3}$	17	3	2	15	23	14	1150	15	67	10586	11805
$\mathbb{F}_{11^2}$	17	2	0	2	1	0	13	2	5	75	110
$\mathbb{F}_{23^2}$	29	2	0	46	22	0	395	46	172	45488	46630

TAB. 6.1 – Calcul non optimisé d'isogénies avec le deuxième algorithme de Couveignes (station DEC).

Chapitre 7

Algorithme en caractéristique deux

Parmi les corps finis de petite caractéristique, ceux de caractéristique 2 jouent un rôle particulier de par leurs applications industrielles [46]. Les deux algorithmes de Couveignes pour déterminer explicitement, comme décrit dans les chapitres 5 et 6, une isogénie entre deux courbes elliptiques sont bien sûr directement applicables à ce cas. Pourtant, contrairement aux corps finis de très grande caractéristique, il s'avère en pratique que le coût majeur lors de la détermination par notre implantation du nombre de points d'une courbe elliptique définie sur $\mathbb{F}_{2^n}$ est celui du calcul d'isogénie.

C'est pourquoi nous avons développé une autre méthode pour résoudre ce problème. Basée sur des identités algébriques satisfaites par les isogénies, par exemple la commutativité avec la multiplication par deux sur la courbe, sa complexité asymptotique heuristique est similaire à celle du premier algorithme de Couveignes quoique cette méthode soit nettement plus efficace en pratique.

Une fois exposées les propriétés spécifiques aux isogénies définies sur $\mathbb{F}_{2^n}$ dans la première partie de ce chapitre (section 7.1), nous expliquons comment en tirer avantageusement parti pour obtenir l'algorithme de la section 7.2 et enfin, nous donnons dans la section 7.3 des statistiques et des temps obtenus avec notre implantation[1].

7.1 Isogénies définies sur $\mathbb{F}_{2^n}$

La 2-torsion d'une courbe elliptique E_a est donnée, comme expliqué au chapitre 2, par l'équation

$$E_a : Y^2 + XY = X^3 + a,\ a \in \mathbb{F}_{2^n}^*,$$

et joue un rôle central dans notre étude des isogénies. Nous en rappelons donc quelques aspects avant de mettre en lumière diverses propriétés des isogénies.

7.1.1 Point de 2-torsion

Une courbe elliptique E_a ne comporte qu'un point P_a d'ordre 2 puisqu'un point $P = (X, Y)$ est égal à son opposé $-P = (X, Y + X)$ si et seulement si $X = 0$. Ce point est donné par

$$P_a = (0, \sqrt{a}).$$

La translation par P_a vaut alors

$$T_{P_a} : P = (X, Y) \mapsto P + P_a = \left(\frac{\sqrt{a}}{X}, \sqrt{a} + \frac{\sqrt{a}}{X} + \frac{a}{X^2} + \sqrt{a}\frac{Y}{X^2} \right), \tag{7.1}$$

et la multiplication par 2 s'écrit

$$[2]_a :\ P = (X, Y) \mapsto 2P = \left(X^2 + \frac{a}{X^2}, \left(X + \frac{Y}{X} \right) \left(X^2 + \frac{a}{X^2} \right) + \frac{a}{X^2} \right). \tag{7.2}$$

[1] Ce chapitre a fait l'objet d'une publication [67].

7.1.2 Caractérisation des isogénies

Ici encore, nous supposons données deux courbes elliptiques E_a et E_b isogènes de degré ℓ, obtenues comme expliqué au chapitre 3 à l'aide d'équations modulaires, et cherchons à déterminer explicitement une isogénie $\mathcal{I}$ les reliant. Dans un premier temps, les formules de Vélu (7.4) nous permettent d'exhiber une telle isogénie.

Théorème 46. *Soit ℓ un entier impair et $d = (\ell-1)/2$. Soient E_a et E_b, deux courbes elliptiques définies sur $\mathbb{F}_{2^n}$ telles qu'il existe une isogénie de degré ℓ entre elles. Il existe alors un polynôme $h(X)$ de degré d, facteur du polynôme de ℓ-division $f_\ell(X)$, tel que l'une de ces isogénies envoie (X,Y) vers*

$$\left(\frac{Xp^2(X)}{h^2(X)}\,,(Y+X^2)\frac{p^2(X)}{h^2(X)}+X^2+\frac{X^2h_I^2(X)}{h^2(X)}\right.$$
$$\left.+X^3\left(\frac{h_I^3(X)}{h^3(X)}+\frac{h_{II}(X)h_I(X)+h_{III}(X)h(X)}{h^2(X)}\right)\right), \quad (7.3)$$

où, en posant $h(X)=h_E^2(X)+Xh_O^2(X)$, nous avons

$$p(X)=h(X)+h_O(X)h_E(X), \quad (7.4)$$

et $h_I(X)=h'(X)=h_O^2(X)$, $h_{II}(X)={h'_E}^2(X)+Xh'_O{}^2(X)$, $h_{III}(X)=h'_{II}(X)={h'_O}^2(X)$.

Démonstration. Par hypothèse, il existe une isogénie $\mathcal{I}'$ de degré ℓ de $E_a(\bar{\mathbb{F}}_{2^n})$ vers $E_b(\bar{\mathbb{F}}_{2^n})$. $\mathrm{Ker}(\mathcal{I}')$ étant un sous-groupe de $E_a(\bar{\mathbb{F}}_{2^n})$, il existe alors une isogénie $\mathcal{I}$ donnée par les formules de Vélu (7.4) ayant pour noyau ce sous-groupe. Le point P_a d'ordre 2 n'est pas dans $\mathrm{Ker}(\mathcal{I}')$ parce que ℓ est impair et que $\mathrm{Ker}(\mathcal{I}')$ est inclus dans le noyau de la multiplication par ℓ de $E_a(\bar{\mathbb{F}}_{2^n})$. On peut par conséquent écrire

$$\mathrm{Ker}(\mathcal{I}')=\{O_{E_a}\}\cup\mathfrak{S}\cup-\mathfrak{S}\text{ avec }\mathfrak{S}\cap-\mathfrak{S}=\{O_{E_a}\},$$

et $\mathcal{I}$ envoie alors un point (X,Y) vers

$$\mathcal{I}(X,Y)=\left(X\left(1+\sum_{P\in\mathfrak{S}}\frac{X_P}{(X-X_P)^2}\right),\right.$$
$$\left.Y+\sum_{P\in\mathfrak{S}}X_P\left(\frac{Y+X^2}{(X-X_P)^2}+\frac{X^2}{(X-X_P)^3}\right)\right). \quad (7.5)$$

Comme $\mathrm{Ker}(\mathcal{I})$ est aussi inclus dans le noyau de la multiplication par ℓ de $E_a(\bar{\mathbb{F}}_{2^n})$, le polynôme $h(X)$ défini par $\prod_{P\in\mathfrak{S}}(X-X_P)$ divise bien $f_\ell(X)$ et on vérifie aisément que les équations (7.5) et (7.3) sont identiques. □

À l'inverse, de telles isogénies doivent satisfaire les conditions nécessaires suivantes.

Théorème 47. *Soient E_a et E_b deux courbes elliptiques définies sur $\mathbb{F}_{2^n}$, ℓ un entier impair et $d=(\ell-1)/2$. Si $\mathcal{I}$ est une isogénie de degré ℓ entre E_a et E_b donnée par*

$$(X,Y)\mapsto\left(\frac{g(X)}{h^2(X)},\frac{l(X)+Yk(X)}{h^3(X)}\right)$$

où $h(X)$, $g(X)$, $l(X)$ et $k(X)$ sont des polynômes de $\mathbb{F}_{2^n}[X]$ de degrés respectifs au plus d, ℓ, $3d$ et $2d$, alors

1. *$g(X)=Xp^2(X)$ où $p(X)$ est un polynôme de degré au plus d tel que $\mathrm{pgcd}(p(X),h(X))=1$ et*

$$X^dh(\sqrt{a}/X)=\frac{\sqrt[8]{a}}{\sqrt[8]{b}}\left(\sqrt[4]{a}\right)^d p(X),$$

ou alternativement via $X\to\sqrt{a}/X$,

$$X^dp(\sqrt{a}/X)=\frac{\sqrt[8]{b}}{\sqrt[8]{a}}\left(\sqrt[4]{a}\right)^d h(X); \quad (7.6)$$

2. $k(X) = p^2(X)h(X)$;
3. $l(X) = Xr(X)p(X) + \sqrt{b}h^3(X) + \sqrt{a}p^2(X)h(X)$ *avec*

$$r(X) = X(ph)'(X) \text{ ou } r(X) = (Xph)'(X).$$

Démonstration. Puisque $[2]_b(\mathcal{I}(P_a)) = \mathcal{I}([2]_a(P_a)) = 0$ et que P_b est le seul point d'ordre 2 de E_b, $\mathcal{I}(P_a) = P_b$. Par conséquent

$$\forall P \in E_a(\mathbb{F}_{2^n}),\ \mathcal{I}(P + P_a) = \mathcal{I}(P) + P_b. \tag{7.7}$$

À partir des formules de la loi d'addition de E_a, nous obtenons pour $P = O_{E_a}$, $g(0)/h^2(0) = 0$, donc X divise $g(X)$. Soit $\Gamma(X) = g(X)/X$. Pour $P = (X, Y) \neq O_{E_a}$, l'équation (7.7) devient

$$\begin{aligned}&\left(\frac{\sqrt{a}\Gamma(\sqrt{a}/X)}{Xh^2(\sqrt{a}/X)}, \frac{l(\sqrt{a}/X) + \sqrt{a}(1 + 1/X + (Y + \sqrt{a})/X^2)k(\sqrt{a}/X)}{h^3(\sqrt{a}/X)}\right)\\ &\qquad = \\ &\left(\frac{\sqrt{b}h^2(X)}{X\Gamma(X)}, \sqrt{b} + \frac{\sqrt{b}h^2(X)}{X\Gamma(X)} + \frac{\sqrt{b}h^4(X)}{X^2\Gamma^2(X)}\left(\frac{l(X) + Yk(X)}{h^3(X)} + \sqrt{b}\right)\right).\end{aligned} \tag{7.8}$$

Après simplification, l'égalité des abscisses de l'équation (7.8) conduit à $\sqrt{a}\,\Gamma(\sqrt{a}/X)\,\Gamma(X) = \sqrt{b}\,h^2(\sqrt{a}/X)\,h^2(X)$. En notant $\tilde{\Gamma}(X) = X^{2d}\Gamma(\sqrt{a}/X)$ et $\tilde{h}(X) = X^d h(\sqrt{a}/X)$, l'équation précédente devient

$$\sqrt{a}\,\tilde{\Gamma}(X)\,\Gamma(X) = \sqrt{b}\,\tilde{h}^2(X)\,h^2(X). \tag{7.9}$$

Puisque $\mathrm{pgcd}(\Gamma(X), h^2(X)) = 1$, $\Gamma(X)$ divise $\tilde{h}^2(X)$. Mais $\Gamma(X)$ a le même degré que $\tilde{h}^2(X)$, il existe donc une constante $\gamma \in \mathbb{F}_{2^n}$ telle que $\Gamma(X) = \gamma^2\tilde{h}^2(X)$ et $\Gamma(X)$ est ainsi un carré. Soit $p(X) = \sqrt{\Gamma(X)}$, alors $p(X) = \gamma\tilde{h}(X)$ ou, alternativement $h(X) = \frac{\sqrt{a}^d}{\gamma}\tilde{h}(X)$. En substituant ces expressions dans (7.9) nous obtenons $\gamma = (\sqrt[4]{a})^d\,\frac{\sqrt[8]{b}}{\sqrt[8]{a}}$, ce qui prouve la relation (7.6).

Puisque $\forall P \in E_a(\mathbb{F}_{2^n}),\ \mathcal{I}(-P) = -\mathcal{I}(P)$, nous avons

$$\frac{l(X) + (Y + X)k(X)}{h^3(X)} = \frac{l(X) + Yk(X)}{h^3(X)} + X\frac{p^2(X)}{h^2(X)},$$

ce qui prouve aussi le point 2 du théorème.

D'autre part, à partir de $\mathcal{I}(P_a) = P_b$, nous obtenons $l(0) = \sqrt{a}p^2(0)h(0) + \sqrt{b}h^3(0)$. Nous pouvons donc écrire $l(X) = \sqrt{a}p^2(X)h(X) + \sqrt{b}h^3(X) + XL(X)$ où $L(X)$ est un polynôme de degré au plus $3d$. Ce qui conduit une fois substitué dans les ordonnées de l'équation (7.8) à

$$\begin{aligned}&\frac{\sqrt{a}L(\sqrt{a}/X)}{Xh^3(\sqrt{a}/X)} + \left(\frac{\sqrt{a}}{X} + \frac{a}{X^2} + \frac{\sqrt{a}Y}{X^2}\right)\frac{p^2(\sqrt{a}/X)}{h^2(\sqrt{a}/X)} = \\ &\qquad\qquad \frac{\sqrt{b}h^2(X)}{Xp^2(X)} + \frac{\sqrt{b}h^4(X)}{X^2p^4(X)}\left(\frac{XL(X)}{h^3(X)} + (Y + \sqrt{a})\frac{p^2(X)}{h^2(X)}\right).\end{aligned} \tag{7.10}$$

En utilisant l'égalité (7.6), cette équation peut être simplifiée en

$$\sqrt{a}X^{3d}L(\sqrt{a}/X)p(X) = \sqrt{b}\left(\frac{\sqrt[8]{a}}{\sqrt[8]{b}}\right)^3\left(\sqrt[4]{a}\right)^{3d}h(X)L(X). \tag{7.11}$$

Comme $\mathrm{pgcd}(p(X), h(X)) = 1$, $p(X)$ divise $L(X)$. Le polynôme $r(X) = L(X)/p(X)$ a un degré au plus égal à $2d$. Puisque pour tout $P \in E_a(\mathbb{F}_{2^n})$, $\mathcal{I}(P) \in E_b(\mathbb{F}_{2^n})$, nous avons

$$\begin{aligned}&\left(X\frac{r(X)p(X)}{h^3(X)} + \sqrt{b} + (Y + \sqrt{a})\frac{p^2(X)}{h^2(X)}\right)^2 + \left(\frac{Xp^2(X)}{h^2(X)}\right)^3 = \\ &\qquad\qquad \left(X\frac{r(X)p(X)}{h^3(X)} + \sqrt{b} + (Y + \sqrt{a})\frac{p^2(X)}{h^2(X)}\right)\frac{Xp^2(X)}{h^2(X)} + b.\end{aligned}$$

Par conséquent, nous avons

$$Xr(X)\left(r(X)+p(X)h(X)\right)=\left((X+\sqrt[4]{a})p(X)h(X)+\sqrt[4]{b}h^2(X)+Xp^2(X)\right)^2. \tag{7.12}$$

Soit $r_1(X)$ une solution polynomiale de l'équation (7.12) et soit $r_2(X)=r_1(X)+p(X)h(X)$. Puisque la partie gauche de l'égalité (7.12) est $Xr_1(X)r_2(X)$ et la partie droite est un carré, X doit diviser $r_1(X)$ ou $r_2(X)$. Comme $r_2(X)$ est aussi une solution de l'équation (7.12), $r_1(X)$ et $r_2(X)$ jouent un rôle symétrique et nous pouvons supposer que X divise $r_1(X)$.

Prouvons maintenant que $r_1(X)/X$ et $r_2(X)$ sont des carrés. Nous savons déjà que $r_1(X)\ r_2(X)/X$ est un carré. Supposons maintenant qu'un polynôme irréductible $\rho(X)$ divise à la fois $r_1(X)/X$ et $r_2(X)$ ($\rho(X)\neq X$). Le polynôme $\rho(X)$ divise alors $r_1(X)+r_2(X)=p(X)h(X)$, donc divise $p(X)$ ou $h(X)$ (pas simultanément car $\gcd(p(X),h(X))=1$). D'autre part, $\rho(X)$ divise la racine carrée de la partie droite de l'égalité (7.12), il divise donc $(X+\sqrt[4]{a})p(X)h(X)+\sqrt[4]{b}h^2(X)+Xp^2(X)$. Si nous supposons que $\rho(X)$ divise $p(X)$, $\rho(X)$ divise $h(X)$ et nous atteignons une contradiction. Comme nous arrivons à la même conclusion si nous supposons que $\rho(X)$ divise $h(X)$, nous avons $\gcd(r_1(X),r_2(X))=1$ et comme $r_1(X)r_2(X)/X$ est un carré, $r_1(X)/X$ et $r_2(X)$ sont aussi des carrés. Posons maintenant $r_1(X)=XO^2(X)$ et $r_2(X)=E^2(X)$, alors

$$XO^2(X)+E^2(X)=p(X)h(X). \tag{7.13}$$

La dérivation de (7.13) donne $r_1=X(p(X)h(X))'$ et celle de (7.13) multipliée par X donne $r_2=(Xp(X)h(X))'$, ce qui prouve le point 3 du théorème. □

Selon les théorèmes 46 ou 47, des isogénies sont complètement déterminées par leurs polynômes $p(X)$ ou $h(X)$. Ne reste plus qu'à trouver des "contraintes" satisfaites par ces derniers.

Corollaire 5. *Avec les notations du théorème 47, les polynômes $p(X)$ et $h(X)$ vérifient*

$$X^d\widehat{h}(X+\sqrt{a}/X)=h(X)p(X), \tag{7.14}$$

et

$$\left(X+\sqrt[4]{a}\right)X^d\widehat{p}\left(X+\sqrt{a}/X\right)=Xp^2(X)+\sqrt[4]{b}h^2(X), \tag{7.15}$$

où $\widehat{p}(X)=\sqrt{p(X^2)}$ et $\widehat{h}(X)=\sqrt{h(X^2)}$ (polynômes dont les coefficients sont les racines carrées des coefficients de $p(X)$ et $h(X)$).

Démonstration. En utilisant que $\forall P\in E_a(\mathbb{F}_{2^n})$, $\mathcal{I}([2]_a(P))=[2]_b(\mathcal{I}(P))$, nous obtenons

$$\left(\frac{Xp^2(X)}{h^2(X)}\right)\circ\left(X^2+\frac{a}{X^2}\right)=\left(X^2+\frac{b}{X^2}\right)\circ\left(\frac{Xp^2(X)}{h^2(X)}\right).$$

En prenant deux fois la racine carrée de cette équation, nous avons alors

$$\left(X+\sqrt[4]{a}\right)\frac{X^d\widehat{p}\left(X+\sqrt{a}/X\right)}{X^d\widehat{h}\left(X+\sqrt{a}/X\right)}=\frac{Xp^2(X)+\sqrt[4]{b}h^2(X)}{p(X)h(X)}. \tag{7.16}$$

Comme $\gcd(Xp^2(X),h^2(X))=1$, les deux fractions de cette égalité sont égales et en égalisant les numérateurs et les dénominateurs, on obtient les relations (7.14) et (7.15). □

À partir du théorème 47 et du corollaire 5, nous déduisons aisément les relations suivantes.

Corollaire 6. *Avec $p(X)=\sum_{i=0}^{d}p_i^2X^i$, $h(X)=X^d+\sum_{i=0}^{d-1}h_i^2X^i$, $\alpha=\sqrt[4]{a}$ et $\beta=\sqrt[4]{b}$, nous avons*

$$h_i=\frac{\sqrt[4]{\alpha}}{\sqrt[4]{\beta}}\sqrt{\alpha}^{d-2i}p_{d-i},\ \forall i\in\{0,\ldots,d\}, \tag{7.17}$$

et

$$p_0=\sqrt[4]{\alpha^{2d}+\alpha^{2d-1}p_{d-1}},\ p_d=1,\ p_{d-1}=\alpha+\beta,$$
$$p_{d-2}=\begin{cases}p_{d-1}^4+\alpha p_{d-1}+\alpha^2 & \text{si } d \text{ est impair,}\\ p_{d-1}^4+\alpha p_{d-1} & \text{si } d \text{ est pair.}\end{cases}$$

Démonstration. L'équation (7.17) est obtenue par une application directe de l'équation (7.6). Le coefficient de X^{2d} dans l'équation (7.14) est $p_d^2 + p_d$, ce qui conduit à $p_d = 1$. Alors, le coefficient de X^{2d} dans l'équation (7.15) est $p_{d-1} + \alpha + \beta$, ce qui donne p_{d-1}. Le coefficient de X^{2d-1} dans l'équation (7.15) est $p_{d-2} + p_{d-1}^4 + \alpha p_{d-1} + \alpha^2$ si d est impair, et $p_{d-2} + p_{d-1}^4 + \alpha p_{d-1}$ si d est pair, ce qui permet de trouver p_{d-2}. Enfin, le coefficient de X dans l'équation (7.15) est égal à $p_0^4 + \alpha^{2d} + \alpha^{2d-1} p_{d-1}$, ce qui nous donne p_0. □

Le corollaire 5 peut être généralisé pour la multiplication par tout entier positif m.

Corollaire 7. *Avec les notations du théorème 47, les polynômes $p(X)$ et $h(X)$ vérifient*

$$f_{m,a}^{\ell}(X) h\left(\frac{g_{m,a}(X)}{f_{m,a}^2(X)}\right) = h^{m^2}(X) f_{m,b}\left(X\frac{p^2(X)}{h^2(X)}\right), \tag{7.18}$$

où m est un entier positif impair, $f_{m,a}(X)$ (resp. $f_{m,b}(X)$) est le polynôme de m-division sur E_a (resp. E_b) et $g_{m,a}(X) = X f_{m,a}^2(X) + f_{m-1,a}(X) f_{m+1,a}(X)$.

7.2 Algorithmes

Nous décrivons ici comment tirer parti des résultats précédents pour calculer explicitement les polynômes $p(X)$ et $h(X)$. Précisément, nous montrons tout d'abord (section 7.2.1) comment utiliser l'équation (7.15) mais malheureusement, la complexité de cette méthode est trop élevée, et nous expliquons ensuite (section 7.2.2) comment nous l'accélérons en utilisant l'équation (7.14). Enfin, nous montrons comment tirer parti des équations (7.4) et (7.18) pour l'améliorer encore sensiblement (section 7.2.3).

7.2.1 Système linéaire défini sur $\mathbb{F}_{2^n}$

La première méthode repose sur l'équation (7.15). D'une part, nous avons

$$X^d \hat{p}\left(\frac{X^2+\alpha^2}{X}\right) = \sum_{i=0}^{d} p_i X^{d-i}(X^2+\alpha^2)^i = \sum_{k=1}^{d}\left(\left(\alpha^{2k}X^{d-k} + X^{d+k}\right)\sum_{i=0}^{\lfloor\frac{d-k}{2}\rfloor} p_{k+2i}\varepsilon_{k+2i,i}\alpha^{2i}\right) + X^d\sum_{i=0}^{\lfloor\frac{d}{2}\rfloor} p_{2i}\varepsilon_{2i,i}\alpha^{2i}$$

où pour tout couple d'entiers (i,j) tels que $0 \leqslant j \leqslant i$, $\varepsilon_{i,j} = \frac{i!}{j!(i-j)!} \bmod 2$. Par conséquent

$$\begin{aligned}(X+\alpha)X^d\hat{p}\left(\frac{X^2+\alpha^2}{X}\right) &= \alpha^{2d+1}p_d + X^{2d+1}p_d + \\ \sum_{k=0}^{d-1} X^{d-k}&\left(\alpha^{2k+1}\sum_{i=0}^{\lfloor\frac{d-k}{2}\rfloor} p_{k+2i}\varepsilon_{k+2i,i}\alpha^{2i} + \alpha^{2k+2}\sum_{i=0}^{\lfloor\frac{d-k-1}{2}\rfloor} p_{k+1+2i}\varepsilon_{k+1+2i,i}\alpha^{2i}\right) \\ +\sum_{k=0}^{d} X^{d+k}&\left(\alpha\sum_{i=0}^{\lfloor\frac{d-k}{2}\rfloor} p_{k+2i}\varepsilon_{k+2i,i}\alpha^{2i} + \sum_{i=0}^{\lfloor\frac{d-k+1}{2}\rfloor} p_{k-1+2i}\varepsilon_{k-1+2i,i}\alpha^{2i}\right).\end{aligned}$$

D'autre part,

$$Xp^2(X) + \beta h^2(X) = \sum_{i=0}^{d} p_i^4 X^{2i+1} + \alpha^{2d+1-4i} p_{d-i}^4 X^{2i}.$$

Une fois factorisées les deux dernières expressions, nous obtenons un polynôme en X dont tous les coefficients sont nuls. En remarquant de plus que le coefficient de X^{d+k} est égal au coefficient de X^{d-k} multiplié par α^{2k}, l'équation (7.15) équivaut finalement au système

$$\boxed{\begin{aligned}\forall k = 0, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor,\ p_k^4 =& \alpha^{2d-4k-1} \sum_{i=0}^{k} p_{d-2k-1+2i} \varepsilon_{d-2k-1+2i,i} \alpha^{2i} \\ &+ \alpha^{2d-4k} \sum_{i=0}^{k} p_{d-2k+2i} \varepsilon_{d-2k+2i,i} \alpha^{2i},\end{aligned}} \tag{7.19}$$

$$\boxed{\begin{aligned}\forall k = 1, \ldots, \left\lfloor \frac{d}{2} \right\rfloor,\ p_{d-k}^4 =& \alpha \sum_{i=0}^{k-1} p_{d+1-2k+2i} \varepsilon_{d+1-2k+2i,i} \alpha^{2i} \\ &+ \sum_{i=0}^{k} p_{d-2k+2i} \varepsilon_{d-2k+2i,i} \alpha^{2i}.\end{aligned}} \tag{7.20}$$

Une première méthode pour résoudre le système (7.19, 7.20) est d'écrire chaque p_i comme une combinaison linéaire dans une base polynomiale $1, t, t^2, \ldots, t^{n-1}$ de $\mathbb{F}_{2^n}$,

$$p_i = p_{i,0} + p_{i,1} t + \cdots + p_{i,n-1} t^{n-1} \text{ avec } \forall j \in \{0, \ldots, n-1\},\ p_{i,j} \in \{0, 1\}.$$

En réécrivant le système (7.19, 7.20) avec ces notations, nous obtenons, une fois substitués p_d, p_{d-1}, p_{d-2}, et p_0 en fonction de α et β (corollaire 6), un système linéaire de $n(d-2)$ équations en $n(d-3)$ variables $p_{i,j}$.

Malheureusement, une telle méthode coûte asymptotiquement $O(\ell^3 n^3)$ opérations élémentaires avec des algorithmes classiques (ou $O(\ell^3 n)$ opérations dans $\mathbb{F}_{2^n}$) et d'autre part, la grande taille de la matrice est un sérieux inconvénient en pratique. Par exemple, pour calculer une isogénie de degré $\ell \simeq 500$ dans $\mathbb{F}_{2^{1000}}$ comme ce que nous avons fait avec le premier algorithme de Couveignes (chapitre 5), nous aurions à stocker une matrice d'un gigaoctet.

Nous suggérons deux améliorations. Plutôt que de résoudre le système (7.19, 7.20) dans $\mathbb{F}_2$, nous écrivons d'abord $[p_1^{2^n}, \ldots, p_{d-3}^{2^n}]$ en fonction de $[p_1^{2^{n-2}}, \ldots, p_{d-3}^{2^{n-2}}]$, et ensuite de la même façon $[p_1^{2^{n-2}}, \ldots, p_{d-3}^{2^{n-2}}]$ en fonction de $[p_2^{2^{n-4}}, \ldots, p_{d-3}^{2^{n-4}}]$. Après $O(n)$ telles itérations, nous obtenons une équation linéaire pour $[p_1, \ldots, p_{d-3}]$ car $p_j^{2^n} = p_j$ dans $\mathbb{F}_{2^n}$. Le coût principal pour résoudre ce système est donc le calcul de $O(n)$ (ou peut-être $O(\log n)$ comme suggéré par Morain) produit de matrices de taille d, c'est-à-dire $O(\ell^3)$ multiplications dans $\mathbb{F}_{2^n}$. Nous aurions donc besoin de $O(\ell^3 n^3)$ opérations (ou peut-être $O(\ell^3 n^2 \log n)$).

Une dernière solution tire parti de la forme du système (7.19, 7.20). Nous obtenons facilement p_{d-3} en fonction de p_1 en fixant $k = 2$ dans l'équation (7.19) et une fois substitué p_{d-3} dans les autres équations, nous obtenons p_{d-4} en fonction de p_1 en fixant $k = 1$ dans l'équation (7.20). Ensuite par récurrence, nous avons facilement p_{d-2i-1} en fonction de $p_1, \ldots, p_i$ en fixant $k = i$ dans l'équation (7.19) ; enfin, une fois substitué p_{d-2i-1} dans les équations restantes, p_{d-2i-2} en fonction de $p_1, \ldots, p_i$ en fixant $k = i+1$ dans l'équation (7.20). Nous réitérons ce procédé jusqu'à $i = d-2i-1$ ou $i = d-2i-2$. Finalement, nous exprimons ainsi $p_{d-3}, \ldots, p_{i+1}$ en fonction de $p_i, \ldots, p_1$ où $i \simeq d/3$. D'autre part les $i+1$ équations (7.19, 7.20) restantes pour $k \geqslant i$ sont des polynômes dont les monômes sont simplement des p_i élevés à une puissance de 2. Ces équations sont donc linéaires une fois considérées sur $\mathbb{F}_2$ et nous pouvons encore utiliser l'une des méthodes décrites précédemment avec une matrice dont la taille est cette fois divisée par 3.

7.2.2 Système quadratique défini sur $\mathbb{F}_2$

L'information obtenue avec l'équation (7.14) nous permet de remplacer un système linéaire à inconnues dans $\mathbb{F}_{2^n}$ par un système non linéaire à inconnues dans $\mathbb{F}_2$.

La partie gauche de l'équation (7.14) est

$$p(X)h(X) = \sqrt{\frac{\alpha}{\beta}} \sum_{k=1}^{d} \left((\alpha^{2k} X^{d-k} + X^{d+k}) \sum_{i=0}^{d-k} p_i^2 p_{k+i}^2 \alpha^{2i-d} \right) + \sqrt{\frac{\alpha}{\beta}} X^d \sum_{i=0}^{d} p_i^4 \alpha^{2i-d}.$$

En remarquant que le coefficient de X^{d-k} est égal au coefficient de X^{d+k} multiplié par α^{2k} dans $p(X)h(X)$ et $X^d\widehat{h}(X+\alpha^2/X)$, l'équation (7.14) conduit aux $d+1$ équations suivantes :

$$\boxed{\forall k = 0, \ldots, d, \ \sqrt[4]{\alpha} \sum_{i=0}^{k} p_i^2 p_{d-k+i}^2 \alpha^{2i} = \sqrt[4]{\beta} \sqrt{\alpha}^{d+2k} \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} p_{k-2i} \varepsilon_{d-k+2i,i}.} \tag{7.21}$$

Du système (7.21), on déduit que chaque p_i est solution d'une équation de degré 2. Par conséquent, $p_i = a_i + \pi_i b_i$, où $b_i \in \mathbb{F}_{2^n}$, a_i dépend de $p_1, \ldots, p_{i-1}$ et $\pi_i \in \mathbb{F}_2$. Avec les idées de la fin de la section 7.2.1, chaque p_k, pour $k = 0, \ldots, d$, peut être réécrit comme un polynôme multivarié en les π_i, ce qui est une simplification importante vis-à-vis des calculs effectués dans la section précédente. D'autre part, comme p_i vérifie une équation de degré 2, les coefficients de cette équation sont reliés par une relation de "compatibilité", ce qui a l'effet surprenant de garder le nombre de ces variables π_i quasi-constant lorsque ℓ croît.

L'algorithme est basé sur ces deux faits. Une fois l'initialisation réalisée (étape 1), il est constitué de deux phases. La première phase est une boucle dans laquelle nous calculons chaque p_k pour $k = 1$, $\ldots$, $d-3$ en fonction des variables binaires π_i. Pour K variant de 0 à $d/3$, nous obtenons p_K avec l'équation (7.21) en fonction des variables binaires π_0, $\ldots$, π_{K-1} (étape 2) et nous extrayons ensuite p_{d-2K+1} (étape 3) et p_{d-2K} (étape 4) en fonction de π_0, $\ldots$, π_{K-1}. Dans une seconde phase (étape 5), nous résolvons les équations satisfaites par les variables binaires π_i et obtenons finalement les p_k.

Algorithme

1. Initialisation : $K = 1$, $K_1 = 0$, $K_2 = 1$ et les coefficients p_0, p_{d-2}, p_{d-1}, p_d sont fixés par le corollaire 6.

Phase 1

2. Au début de cette phase, nous avons p_0, $\ldots$, p_{K-1} et p_{d-2K+2}, $\ldots$, p_d connus en fonction de variables binaires π_0, $\ldots$, π_{K-2}. Nous réécrivons l'équation (7.21) pour $k = K$ en

$$p_K^2 + b_K p_K + c_K = 0 \tag{7.22}$$

où

$$c_K = \left(\sqrt[4]{\alpha} \sum_{i=0}^{K-1} p_i^2 p_{d-K+i}^2 \alpha^{2i} + \sqrt[4]{\beta} \sqrt{\alpha}^{d+2K} \sum_{i=1}^{\lfloor \frac{K}{2} \rfloor} p_{K-2i} \varepsilon_{d-K+2i,i} \right) / \left(\alpha^{2K} \sqrt[4]{\alpha} \right)$$

et

$$b_K = \sqrt[4]{\beta} \sqrt{\alpha}^{d+2K} / (\alpha^{2K} \sqrt[4]{\alpha}).$$

Ainsi, c_K est un polynôme multivarié en les inconnues π_0, $\ldots$, π_{K-2}.
Écrivons $c_K / b_K^2 = \sum_{(\mu_0, \ldots, \mu_{K-2}) \in \{0,1\}^{K-1}} C_\mu \pi_0^{\mu_0} \cdots \pi_{K-2}^{\mu_{K-2}}$. L'équation (7.22) a une solution si et seulement si $\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(c_K/b_K^2) = 0$, c'est-à-dire

$$\sum_{\substack{(\mu_0, \ldots, \mu_{K-2}) \in \{0,1\}^{K-1} \\ \mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(C_\mu) = 1}} \pi_0^{\mu_0} \cdots \pi_{K-2}^{\mu_{K-2}} = 0. \tag{7.23}$$

La partie gauche de l'équation (7.23) peut être

(a) 1 : alors $E_a(\mathbb{F}_{2^n})$ et $E_b(\mathbb{F}_{2^n})$ ne sont pas isogènes et nous retournons `ERREUR`.

(b) 0 : nous posons

$$p_K = b_K \pi_{K-1} + b_K \sum_{(\mu_0,\ldots,\mu_{K-2})\in\{0,1\}^{K-1}} P_\mu \pi_0^{\mu_0} \cdots \pi_{K-2}^{\mu_{K-2}}$$

où $\pi_{K-1} \in \mathbb{F}_2$ et P_μ est égal à n'importe quelle solution de $X^2 + X + C_\mu = 0$ (on vérifie facilement que cette formule nous donne les deux solutions de l'équation (7.22)).

(c) un polynôme multivarié à coefficients dans $\mathbb{F}_2$: nous obtenons un monôme de l'équation (7.23) en fonction des autres et le substituons dans c_K/b_K^2 pour obtenir p_K comme au point 2b. À chaque fois que cela est possible, nous choisissons pour ce monôme une variable π_k, ce qui est toujours le cas en pratique, et nous le substituons en $p_0, \ldots p_{K-1}$ et $p_{d-2K+2}, \ldots, p_d$ afin de réduire le nombre des variables binaires à manipuler.

Finalement, nous incrémentons K et si $K > d - 2K + 1$, nous continuons à étape 5.

3. Nous posons $K_1 = K$ et nous extrayons p_{d-2K+1} en fonction de $\pi_0, \ldots, \pi_{K-2}$ à partir de l'équation (7.19) pour $k = K_1$. Si $K > d - 2K$, nous allons à l'étape 5.

4. Nous posons $K_2 = K - 1$ et nous extrayons p_{d-2K} en fonction de $\pi_0, \ldots, \pi_{K-2}$ à partir de l'équation (7.20) pour $k = K_2$. Puis, nous retournons à l'étape 2 si $K \neq d - 2K$.

Phase 2

5. À cette étape, chaque p_i, $i = 0, \ldots, d$, est un polynôme multivarié en au plus $K-1$ variables binaires, $\pi_0, \ldots, \pi_{K-2}$ avec $K = (d+2)/3$ (resp. $(d+1)/3$, $d/3$), $K_1 = (d-1)/3$ (resp. $(d+1)/3$, $d/3$) et $K_2 = (d-4)/3$ (resp. $(d-5)/3$, $d/3-1$) lorsque $d \equiv 0 \bmod 3$ (resp. $d \equiv 1 \bmod 3$, $d \equiv 2 \bmod 3$). Ainsi, $d - 1 - K_1 - K_2 = K$ équations des systèmes (7.19) et (7.20) demeurent non triviales.

Nous substituons les p_i de ces K équations. Il est alors facile d'obtenir une variable π_{K-2} comme une fraction de $\pi_0, \ldots, \pi_{K-3}$ à partir d'une de ces équations pour le substituer dans les $K-1$ équations restantes et on réitère ce procédé jusqu'à π_0. Si les deux dernières équations ne donnent pas la même valeur pour π_0, $E_a(\mathbb{F}_{2^n})$ et $E_b(\mathbb{F}_{2^n})$ ne sont pas isogènes et nous retournons `ERREUR`. Sinon, il ne reste plus qu'à obtenir pas à pas, tout d'abord les valeurs numériques de $\pi_1, \ldots, \pi_{K-2}$, puis celles de $p_1, \ldots, p_{d-3}$.

Exemple : dans $\mathbb{F}_{2^{10}} \simeq \mathbb{F}_2[t]/(t^{10}+t^3+1)$ avec la notation $\overline{\tau_0 + \tau_1 2 + \cdots + \tau_9 2^9} = \tau_0 + \tau_1 t + \cdots + \tau_9 t^9$, nous allons calculer une isogénie de degré $\ell = 37$ entre $E_{\overline{6}}$ et $E_{\overline{272}}$. Ici, $d = 18$, $\alpha = \overline{794}$ et $\beta = \overline{6}$.

À l'étape 1 de l'algorithme $p_0 = \overline{153}$, $p_{16} = \overline{334}$, $p_{17} = \overline{796}$, $p_{18} = \overline{1}$. Puis, la première phase de l'algorithme est constituée de cinq itérations répertoriées dans le tableau 7.1.

K	1	2	3	4	5
b_K	$\overline{253}$	$\overline{212}$	$\overline{536}$	$\overline{575}$	$\overline{470}$
c_K	$\overline{151}$	$\overline{259}\pi_0 + \overline{714}$	$\overline{847}+\overline{58}\pi_0+\overline{453}\pi_1$	$\overline{374} + \overline{574}\pi_0 + \overline{804}\pi_1 + \overline{387}\pi_2$	$\overline{669} + \overline{353}\pi_0 + \overline{141}\pi_1 + \overline{492}\pi_0\pi_1 + \overline{487}\pi_2 + \overline{418}\pi_3$
p_K	$\overline{253}\pi_0 + \overline{581}$	$\overline{212}\pi_1 + \overline{609}\pi_0 + \overline{444}$	$\overline{536}\pi_2 + \overline{182}\pi_1 + \overline{412}\pi_0 + \overline{329}$	$\overline{575}\pi_3 + \overline{77}\pi_2 + \overline{24}\pi_1 + \overline{574}\pi_0 + \overline{94}$	$\overline{470}\pi_4 + \overline{741}\pi_3 + \overline{86}\pi_2 + \overline{849}\pi_0\pi_1 + \overline{656}\pi_1 + \overline{449}\pi_0 + \overline{724}$
p_{d-2K-1}	$\overline{6}\pi_0 + \overline{364}$	$\overline{6}\pi_1 + \overline{529}\pi_0 + \overline{121}$	$\overline{6}\pi_2 + \overline{529}\pi_1 + \overline{268}\pi_0 + \overline{611}$	$\overline{6}\pi_3 + \overline{529}\pi_2 + \overline{570}\pi_1 + \overline{822}\pi_0 + \overline{853}$	$\overline{6}\pi_4 + \overline{529}\pi_3 + \overline{566}\pi_2 + \overline{521}\pi_0\pi_1 + \overline{187}\pi_1 + \overline{23}\pi_0 + \overline{434}$
p_{d-2K-2}	$\overline{590}\pi_0 + \overline{451}$	$\overline{590}\pi_1 + \overline{811}\pi_0 + \overline{391}$	$\overline{590}\pi_2 + \overline{571}\pi_1 + \overline{802}\pi_0 + \overline{450}$	$\overline{590}\pi_3 + \overline{571}\pi_2 + \overline{320}\pi_1 + \overline{53}\pi_0 + \overline{575}$	$\overline{590}\pi_4 + \overline{571}\pi_3 + \overline{197}\pi_2 + \overline{514}\pi_0\pi_1 + \overline{647}\pi_1 + \overline{23}\pi_0 + \overline{240}$

TAB. 7.1 – Exemple de calculs réalisés dans la première phase.

Dans la seconde phase (étape 5), les équations (7.19) pour $k = 7, 8$ et (7.20) pour $k = 6, 7, 8, 9$ donnent

$$\begin{aligned}
\overline{331} + \overline{615}\pi_0 + \overline{438}\pi_1 + \overline{436}\pi_2 + \overline{331}\pi_3 &= 0 && (7.24)\\
\overline{168} + \overline{125}\pi_0 + \overline{867}\pi_1 + \overline{384}\pi_0\pi_1 + \overline{350}\pi_2 + \overline{795}\pi_3 + \overline{947}\pi_4 &= 0 && (7.25)\\
\overline{444} + \overline{399}\pi_0 + \overline{238}\pi_1 + \overline{849}\pi_0\pi_1 + \overline{523}\pi_2 + \overline{991}\pi_3 + \overline{611}\pi_4 &= 0 && (7.26)\\
\overline{611} + \overline{217}\pi_0 + \overline{245}\pi_1 + \overline{654}\pi_0\pi_1 + \overline{268}\pi_2 + \overline{522}\pi_3 + \overline{105}\pi_4 &= 0 && (7.27)\\
\overline{214} + \overline{574}\pi_0 + \overline{666}\pi_1 + \overline{848}\pi_0\pi_1 + \overline{255}\pi_2 + \overline{2}\pi_3 + \overline{212}\pi_4 &= 0 && (7.28)\\
\overline{127} + \overline{163}\pi_0 + \overline{111}\pi_1 + \overline{394}\pi_0\pi_1 + \overline{704}\pi_2 + \overline{851}\pi_3 + \overline{812}\pi_4 &= 0 && (7.29)
\end{aligned}$$

Avec l'équation (7.26), nous obtenons

$$\pi_4 = \overline{843} + \overline{935}\pi_0 + \overline{255}\pi_1 + \overline{256}\pi_0\pi_1 + \overline{470}\pi_2 + \overline{842}\pi_3$$

Une fois π_4 substitué, l'équation (7.25) conduit à

$$\pi_3 = \overline{1} + \overline{876}\pi_0 + \overline{103}\pi_1 + \overline{796}\pi_2.$$

Alors π_4 et π_3 substitués dans l'équation (7.24) donnent

$$\pi_2 = \overline{979}\pi_0 + \overline{144}\pi_1 + \overline{194}\pi_0\pi_1,$$

puis π_4, π_3 et π_2 substitués dans l'équation (7.27),

$$\pi_1 = \pi_0/(\overline{176} + \overline{795}\pi_0).$$

Finalement, l'équation (7.28) donne $\pi_0 = 0$, et nous avons seulement à vérifier que l'équation (7.29) est triviale, ce qui est le cas. Par conséquent, $\pi_1 = 0$, $\pi_2 = 0$, $\pi_3 = 1$, $\pi_4 = 1$, et

$$\begin{aligned}
p(X) = &X^{18} + \overline{514}\,X^{17} + \overline{560}\,X^{16} + \overline{573}\,X^{15} + \overline{753}\,X^{14} + \overline{364}\,X^{13} + \overline{709}\,X^{12} +\\
&\overline{314}\,X^{11} + \overline{752}\,X^{10} + \overline{627}\,X^9 + \overline{300}\,X^8 + \overline{986}\,X^7 + \overline{129}\,X^6 + \overline{744}\,X^5 +\\
&\overline{318}\,X^4 + \overline{549}\,X^3 + \overline{905}\,X^2 + \overline{295}\,X + \overline{465}.
\end{aligned}$$

Remarques : le nombre des solutions de ce système est malheureusement variable :

- quand $E_a(\mathbb{F}_{2^n})$ et $E_b(\mathbb{F}_{2^n})$ sont isogènes de degré ℓ, il n'y a généralement que deux isogénies de E_a vers E_b définies sur $\mathbb{F}_{2^n}$ (une isogénie et son opposée) et dans ce cas, l'algorithme a toujours produit une unique solution dans nos expérimentations. Si par hasard le système d'équations avait plusieurs solutions, nous aurions à les calculer toutes puis à tester pour chacune d'entre elles s'il s'agit vraiment d'une isogénie.
- dans quelques cas pathologiques extrêmement rares, plus de deux isogénies relient les courbes $E_a(\mathbb{F}_{2^n})$ et $E_b(\mathbb{F}_{2^n})$ et ces systèmes ont plusieurs solutions.
- quand $E_a(\mathbb{F}_{2^n})$ et $E_b(\mathbb{F}_{2^n})$ ne sont pas isogènes, ces systèmes n'ont en pratique aucune solution et ceci peut être détecté rapidement car à l'étape 2, l'équation (7.23) est la plupart du temps fausse (typiquement $1 = 0$) pour un petit indice K.

À l'étape 5 de l'algorithme, les K équations obtenues à partir du système (7.19,7.20) équivalent en fait à nK équations entre les inconnues binaires π_i une fois réécrites dans une base polynomiale $1, \ldots, t^{n-1}$ de $\mathbb{F}_{2^n}$. Premièrement, cela signifie que ce système est très contraint, deuxièmement, nous pouvons utiliser cette remarque pour accélérer les calculs. Par exemple, l'équation $\pi_1(\overline{176} + \overline{795}\pi_0) = \pi_0$ conduit à $\pi_1\pi_0 t^9 + \pi_1\pi_0 t^8 + \pi_1 t^7 + \pi_1 t^5 + \pi_1(\pi_0 + 1)t^4 + \pi_1\pi_0 t^3 + \pi_1\pi_0 t + \pi_1\pi_0 + \pi_0 = 0$ et nous trouvons immédiatement $\pi_0 = 0$ et $\pi_1 = 0$.

Complexité : la complexité de l'algorithme est difficile à estimer. Comme expliqué au début de cette section, le nombre de variables reste quasi-constant pendant le calcul grâce aux équations (7.23). Pour $K > 10$, chaque fois que nous écrivons p_K en fonction des nouvelles variables π_{K-1}, nous avons remarqué que nous étions capables d'obtenir une variable π_k avec un indice $k < K - 1$ en fonction des autres π_i

excepté un nombre logarithmique de fois. Nous supposons donc qu'asymptotiquement le nombre de variables π_i croît heurististiquement comme $O(\log \ell)$.

Avec cette hypothèse, le nombre maximal de termes dans les polynômes multivariés de cet algorithme est au plus $O(2^{\log(\ell)}) = O(\ell)$. Ainsi, nous estimons que le coût de cet algorithme est probablement au plus $O(\ell^3)$ multiplications dans $\mathbb{F}_{2^n}$.

Pour le stockage, nous avons à écrire dans la phase 1 que chaque p_K est un polynôme multivarié de $O(\log \ell)$ variables binaires, d'où un stockage en $O(\ell^2)$ éléments de $\mathbb{F}_{2^n}$. Dans la phase 2, on doit écrire $d/3$ équations en fonction de $O(\log \ell)$ variables binaires, ce qui conduit également à un stockage en $O(\ell^2)$.

7.2.3 Améliorations pratiques

Les améliorations décrites ici sont basées sur l'équation (7.4) et sur l'équation (7.18). Elles sont pratiques dans le sens qu'elles ne changent probablement pas la complexité asymptotique de l'algorithme précédent.

Équations de Vélu.

L'équation (7.4) nous permet de faire décroître de moitié le nombre des variables binaires π_i intermédiaires. Avec

$$h_O(X) = \sqrt[4]{\frac{\alpha}{\beta}} \sum_{k=0}^{\lfloor \frac{d}{2} \rfloor} p_{d-2k} \sqrt{\alpha}^{d-4k} X^k \text{ et } h_E(X) = \sqrt[4]{\frac{\alpha}{\beta}} \sum_{k=0}^{\lfloor \frac{d-1}{2} \rfloor} p_{d-2k-1} \sqrt{\alpha}^{d-4k-2} X^k,$$

nous obtenons

$$\begin{aligned} h_E(X)h_O(X) = \sqrt{\frac{\alpha}{\beta}} \sum_{k=0}^{\lfloor \frac{d-1}{2} \rfloor} X^k \alpha^{d-2k-1} \left(\sum_{i=0}^{k} p_{d-2i-1} p_{d-2k+2i} \right) \\ + \sqrt{\frac{\alpha}{\beta}} \sum_{k=\lfloor \frac{d+1}{2} \rfloor}^{\lfloor \frac{d}{2} \rfloor} X^k \alpha^{d-2k-1} \left(\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} p_{d-2i-1} p_{d-2k+2i} \right) \\ + \sqrt{\frac{\alpha}{\beta}} \sum_{k=\lfloor 1+\frac{d}{2} \rfloor}^{\lfloor d+\frac{1}{2} \rfloor} X^k \alpha^{d-2k-1} \left(\sum_{i=k-\lfloor \frac{d}{2} \rfloor}^{\lfloor \frac{d-1}{2} \rfloor} p_{d-2i-1} p_{d-2k+2i} \right), \end{aligned} \tag{7.30}$$

et

$$p(X) + h(X) = \frac{1}{\sqrt{\beta}} \sum_{k=0}^{d} \left(\sqrt{\beta} p_k^2 + \sqrt{\alpha} p_{d-k}^2 \alpha^{d-2k} \right) X^k.$$

L'équation (7.4) est donc équivalente au système de $d+1$ équations suivant,

$$\boxed{\begin{aligned} &\alpha^{2k-d} \sqrt{\alpha\beta} p_k^2 + \alpha p_{d-k}^2 = \\ &\qquad \begin{cases} \displaystyle\sum_{i=0}^{k} p_{d-2i-1} p_{d-2k+2i}, & \text{si } k = 0, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor, \\ \displaystyle\sum_{i=0}^{d/2-1} p_{d-2i-1} p_{2i}, & \text{si } d \text{ est pair et } k = \frac{d}{2}, \\ \displaystyle\sum_{i=k-\lfloor \frac{d}{2} \rfloor}^{\lfloor \frac{d-1}{2} \rfloor} p_{d-2i-1} p_{d-2k+2i}, & \text{si } k = \left\lfloor \frac{d+3}{2} \right\rfloor, \ldots, d. \end{cases} \end{aligned}} \tag{7.31}$$

L'utilisation de ce système d'équations est immédiat. À l'étape 2 de l'algorithme, quand $K = 1 \bmod 2$ ($K \geqslant 3$), nous remplaçons le calcul de p_K avec l'équation (7.21) par celui de p_K avec l'équation (7.31) pour $k = K$. De cette façon, nous obtenons directement p_K en fonction de $p_1, \ldots, p_{K-1}$ et nous n'avons pas à introduire de nouvelle variable binaire π_K. Ainsi, nous sommes capables d'exprimer $p_0, \ldots, p_d$ en fonction d'au plus $d/6$ variables binaires au lieu de $d/3$. Malheureusement, dans ce cas, l'équation (7.23) devient utile plus tard en pratique ($K \leqslant 22$, ce qui est deux fois plus qu'initialement). Ainsi pour un "grand" degré ℓ ($\ell > 200$), le nombre des variables π_i à utiliser ($O(\log \ell)$) est sensiblement identique à celui de l'algorithme original.

Multiplication par trois.

Notre dernière amélioration repose sur l'équation (7.18) avec $m = 3$. Cela nous permet d'exprimer une variable π_i en fonction de $\pi_0, \ldots, \pi_{i-1}$ bien plus tôt qu'avec l'équation (7.23).

Pour $m = 3$, l'équation (7.18) devient

$$(X^4 + X^3 + a) \sum_{i=0}^{d} h_i (X^4 + \alpha^2 X + a)^i (X^4 + X^3 + a)^{d-i} = X^4 p^8(X) h(X) + X^3 p^6(X) h^3(X) + b h^9(X). \quad (7.32)$$

Tout d'abord, le membre gauche de l'équation (7.32) peut-être réécrit sous la forme

$$\sum_{i=0}^{d} h_i (X^4 + \alpha^2 X + a)^i (X^4 + X^3 + a)^{d-i+1} = \sum_{i=0}^{9d+4} X^i \sum_{\substack{j \\ 0, i-d}}^{8d+4, i\,\prime} h_{i-j}^2 T_{i,j}^2$$

$$\text{avec } T_{i,j} = \sum_{\substack{k \\ 0, 8d+8i-7j-4}}^{j, 8i-8j\,\prime} \left(\sum_{\substack{l \\ 0, \lceil \frac{k-2i+2j}{3} \rceil}}^{\lfloor \frac{i}{4} \rfloor\,\prime} \varepsilon_{2i-2j,l} \varepsilon_{2i-2j-l,k-4l} \alpha^{2l} \right) \left(\sum_{\substack{l \\ 0, 2d-2i+2j+1-\lfloor \frac{j-k}{3} \rfloor}}^{2d-2i+2j+1-\lceil \frac{j-k}{4} \rceil\,\prime} \varepsilon_{2d-2i+2j+1,l} \varepsilon_{2d-2i+2j+1-l, 8d-8i+7j+4+k-4l} \alpha^{2l} \right)$$

où la notation $\sum_{\substack{j \\ k,l}}^{m,n\,\prime}$ signifie $\sum_{j=\max(k,l)}^{\min(m,n)}$. D'autre part,

$$p^8(X) h(X) = \sum_{i=0}^{9d} X^i \sum_{\substack{k \\ 0, \lceil \frac{i-d}{8} \rceil}}^{\lfloor \frac{i}{8} \rfloor, d\,\prime} p_k^{16} h_{i-8k}^2, \quad h^9(X) = \sum_{i=0}^{9d} X^i \sum_{\substack{k \\ 0, \lceil \frac{i-d}{8} \rceil}}^{\lfloor \frac{i}{8} \rfloor, d\,\prime} h_k^{16} h_{i-8k}^2,$$

$$X^3 p^6 h^3(x) = \sum_{i=3}^{9d+3} X^i \sum_{\substack{j \\ 0, \lceil \frac{i-3-d}{2} \rceil}}^{\lfloor \frac{i-3}{2} \rfloor, 4d\,\prime} h_{i-2j-3}^2 \sum_{\substack{k \\ 0, j-d}}^{j, 3d\,\prime} h_{j-k}^4 \sum_{\substack{l \\ 0, \lceil \frac{k-d}{2} \rceil}}^{d, \lfloor \frac{k}{2} \rfloor\,\prime} p_l^8 p_{k-2l}^4.$$

Ce système est donc

$$\sum_{0,\lceil\frac{i-4-d}{8}\rceil}^{\lfloor\frac{i-4}{8}\rfloor,d}{}' p_k^8 h_{i-4-8k} + \beta^2 \sum_{0,\lceil\frac{i-d}{8}\rceil}^{\lfloor\frac{i}{8}\rfloor,d}{}' h_k^8 h_{i-8k} = \sum_{0,i-d}^{8d+4,i}{}' h_{i-j} T_{i,j} + \sum_{0,\lceil\frac{i-3-d}{2}\rceil}^{\lfloor\frac{i-3}{2}\rfloor,4d}{}' h_{i-2j-3} \sum_{0,j-d}^{j,3d}{}' h_{j-k}^2 \sum_{0,\lceil\frac{k-d}{2}\rceil}^{d,\lfloor\frac{k}{2}\rfloor}{}' p_l^4 p_{k-2l}^2. \tag{7.33}$$

avec $i = 4, \ldots, 9d$.

De cette expression, nous déduisons qu'à l'étape 2 de l'algorithme, nous pouvons exprimer uniquement les équations de ce système en fonction des variables binaires π_i pour $i \leqslant 2K$. Ainsi, nous obtenons de nouvelles relations entre les π_i.

Nous avons observé qu'au cours des calculs, seul l'indice i de l'équation (7.33) fixé à $2K-1$ lorsque $K = 1 \bmod 2$ permet d'obtenir une relation non triviale. Dans ce cas, pour K suffisamment grand, nous pouvons exprimer une variable π_i en fonction des autres, sans influence néfaste sur les autres équations. En pratique, ce phénomène se produit déjà pour $K = 11$. Ainsi, au prix d'un calcul additionnel, le nombre de variables binaires à utiliser est asymptotiquement plus petit que l'algorithme original.

7.3 Résultats

Nous avons programmé avec la librairie `ZEN` (cf. chapitre 10) l'algorithme décrit dans la section 7.2.2 (`MULTBY2`) et ses améliorations décrites dans les sections 7.2.3 (`VÉLU`) et 7.2.3 (`MULTBY3`). Remarquons simplement du point de vue informatique que les polynômes multivariés que nous manipulons sont creux ; nous les avons donc implantés comme des listes chaînées. Nous n'avons pas implanté les méthodes de la section 7.2.1 car nous estimons que les ressources nécessaires aussi bien en espace qu'en temps sont trop élevées. D'autre part, nous comparons ces algorithmes avec notre implantation du premier algorithme de Couveignes (`COUVEIGNES`) décrite au chapitre 5.

Nous avons mesuré le temps nécessaire pour calculer des isogénies de degré premier ℓ tel que $3 < \ell < 500$ dans $\mathbb{F}_{2^{10}} \simeq \mathbb{F}_2[T]/(T^{10}+T^3+1)$ sur une station DEC ainsi que le nombre de variables binaires restantes π au début de l'étape 5. Les résultats sont donnés sur la figure 7.1.

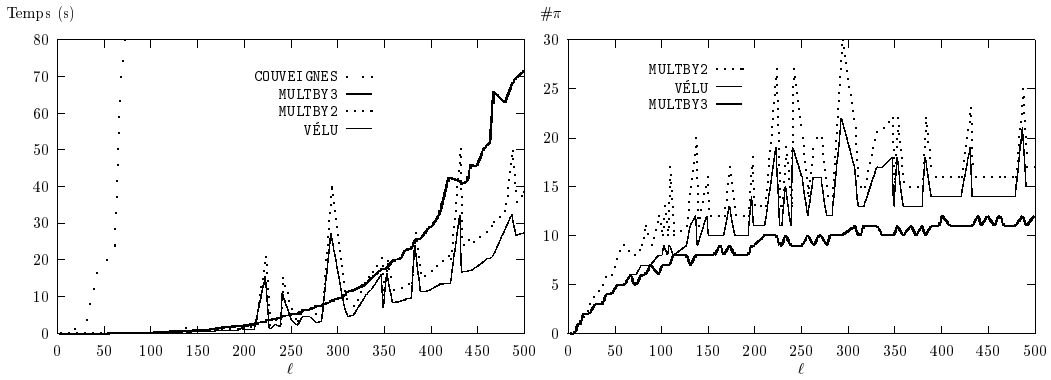


FIG. 7.1 – Temps et nombre de variables nécessaires au calcul d'isogénies (station DEC).

Par souci de comparaison, nous avons ensuite mesuré le temps nécessaire au calcul des isogénies utilisé à la détermination du nombre de points de $E_{\overline{91128}}$ définies sur $\mathbb{F}_{2^{300}} \simeq \mathbb{F}_2[T]/(T^{300}+T^5+1)$ par ces quatre algorithmes.

	COUVEIGNES	MULTBY2	VELU	MULTBY3
Isogenies (s)	10204	75	38	42
Total (s)	13441	2995	2851	2862

FIG. 7.2 – Temps mesurés lors du calcul de la cardinalité de $E_{\overline{91128}}$ sur $\mathbb{F}_{2^{300}}$ (station DEC).

Comme nous pouvons nous en rendre compte au travers du tableau 7.2, ce temps est maintenant complètement négligeable vis-à-vis du temps nécessaire au calcul complet de la cardinalité.

Chapitre 8

Combinaison d'algorithmes en caractéristique impaire

À ce stade, un bref bilan montre que nous disposons d'algorithmes de calcul d'isogénies efficaces pour des corps finis $\mathbb{F}_{p^n}$ de caractéristique p avec $p = 2$ (cf. chapitre 7) ou p grand, disons $p > 1000$ (cf. chapitre 4). Pour des caractéristiques p petites, les deux algorithmes de Couveignes (cf. chapitre 5 et 6) sont utilisables même si pour p de taille moyenne, disons $p \simeq 100$, ils sont par trop inefficaces. Remarquons enfin que pour $\ell > p > 5$, l'algorithme `CCR` donne certains des coefficients du facteur $h(X)$ de degré $(\ell - 1)/2$ du polynôme de ℓ-division mais nous n'avons pas encore exploité ces données.

Notre motivation initiale à la réalisation des travaux de ce chapitre était la généralisation de l'algorithme du chapitre 7 à des corps finis $\mathbb{F}_{p^n}$ de caractéristique p moyenne. En suivant une démarche analogue au cas $p = 2$ et notamment en exploitant la translation par des points d'ordre deux comme nous le montrons dans la section 8.1, nous arrivons aussi à un système d'équations quadratiques. Malheureusement, nous n'avons trouvé aucun algorithme efficace de résolution.

Par contre, les résultats partiels que nous obtenons, et en particulier, pour la première fois, une expression linéaire des coefficients du numérateur $g(X)$ de l'abscisse $g(X)/h^2(X)$ d'une isogénie en fonction de ceux de $h^2(X)$, nous autorisent, en conjonction avec les renseignements partiels de l'algorithme `CCR`, à diminuer d'un facteur au moins égal à quatre la taille des objets nécessaires aux algorithmes de Couveignes. Nous montrons ce point dans la section 8.2. Nous confortons par ailleurs ce gain avec une amélioration heuristique basée sur les formules de Vélu (cf. chapitre 4), amélioration qui prend justement tout son sens pour des caractéristiques moyennes.

Enfin, nous donnons un exemple et des temps précis dans la section 8.3.

8.1 Isogénies et points de 2-torsion

Manifestement, les relations sous-jacentes à l'algorithme du chapitre 7 pour $\mathbb{F}_{2^n}$ découlent de la translation T_{P_a} par le point P_a d'ordre 2 de la courbe. Pour généraliser cet algorithme à d'autres corps finis, il est donc naturel de se préoccuper aussi de cette translation et nous montrons dans une première partie comment obtenir, à partir de celle-ci, une caractérisation des isogénies semblable à celle du chapitre 7.

Nos tentatives pour tirer un algorithme complet de ces résultats se sont malheureusement soldé pas un échec. Néanmoins, nous expliquons dans une deuxième partie comment une application immédiate nous permet de déduire d'une part les coefficients du numérateur $g(X)$ de l'abscisse $g(X)/h^2(X)$ d'une isogénie linéairement en fonction de ceux de $h^2(X)$, et d'autre part au moins $(\ell - 1)/2$ relations linéaires entre les coefficients de $h^2(X)$.

8.1.1 Caractérisation

Nous nous intéressons à des courbes elliptiques non supersingulières définies en toute généralité par l'équation

$$E_a \ : Y^2 = X^3 + a_2X^2 + a_4X + a_6$$

dans un corps fini $\mathbb{F}_{p^n}$ de caractéristique p impaire. En effet, d'après le tableau 2.1, il suffit de prendre $a_4 = 0$ pour retrouver le cas $p = 3$ et $a_2 = 0$ pour le cas $p \geqslant 5$. Les points d'ordre 2 de E_a sont d'ordonnée nulle, c'est-à-dire d'abscisse α telle $\alpha^3 + a_2\alpha^2 + a_4\alpha + a_6 = 0$.

Il est facile de montrer que la translation par un point $P_\alpha = (\alpha, 0)$ d'ordre 2 de $E_a(\bar{\mathbb{F}}_{p^n})$ est donnée par

$$\begin{array}{rccl} T_{P_\alpha} \ : & E_a(\bar{\mathbb{F}}_{p^n}) & \to & E_a(\bar{\mathbb{F}}_{p^n}) \\ & P & \mapsto & \begin{cases} P_\alpha & \text{si } P = O_{E_a}, \\ O_{E_a} & \text{si } P = -P_\alpha, \\ \left(\dfrac{\alpha X + 2\alpha^2 + 2a_2\alpha + a_4}{X - \alpha}, -Y\dfrac{3\alpha^2 + 2a_2\alpha + a_4}{(X-\alpha)^2}\right) & \text{sinon.} \end{cases} \end{array}$$

Dans la suite, nous ferons un large usage de l'abscisse de T_{P_α} que nous abrégeons en

$$T_\alpha(X) = \frac{\alpha X + 2\alpha^2 + 2a_2\alpha + a_4}{X - \alpha}.$$

Notons que de $T_{P_\alpha} \circ T_{P_\alpha}(P) = P + 2P_\alpha = P$, l'on déduit $T_\alpha(T_\alpha(X)) = X$.

Dans le même esprit que le théorème 47, nous avons alors la proposition suivante.

Proposition 28. *Avec les notations précédentes, soit $\mathcal{I}$ une isogénie de degré ℓ impair ($\ell = 2d+1$) entre E_a et E_b donnée par*

$$(X, Y) \mapsto \left(\frac{g(X)}{h^2(X)}, Y\frac{k(X)}{h^3(X)}\right)$$

où $h(X)$, $g(X)$ et $k(X)$ sont des polynômes unitaires de $\bar{\mathbb{F}}_{p^n}[X]$ de degrés respectifs au plus d, ℓ et $3d$. Alors il existe un point $Q_\beta = (\beta, 0)$ d'ordre 2 de $E_b(\bar{\mathbb{F}}_{p^n})$ tel que

1. $$(X - \alpha)^\ell h^2(T_\alpha(X)) = h^2(\alpha)\left(g(X) - \beta h^2(X)\right), \tag{8.1}$$

   où $h^2(\alpha)$ est l'une des racines carrées de $\dfrac{(3\alpha^2 + 2a_2\alpha + a_4)^\ell}{3\beta^2 + 2b_2\beta + b_4}$.

2. *$k(X) = (X - \alpha)^d h(T_\alpha(X))K(X)$ où $K(X)$ est un polynôme de degré $2d$ vérifiant*

$$(3\alpha^2 + 2a_2\alpha + a_4)(X - \alpha)^{2d}K(T_\alpha(X)) = (3\beta^2 + 2b_2\beta + b_4)K(X).$$

Démonstration : tout découle de la propriété de commutativité de $\mathcal{I}$ avec T_{P_α}. En effet, posons $P_\beta = (\beta, 0)$, le point d'ordre 2 de $E_b(\bar{\mathbb{F}}_{p^n})$ égal à $\mathcal{I}(P_\alpha)$. Nous avons alors $\mathcal{I} \circ T_{P_\alpha} = T_{P_\beta} \circ \mathcal{I}$. Pour tout élément X de $\bar{\mathbb{F}}_{p^n}$ distinct de α, ceci se traduit pour les abscisses, par

$$\frac{g(T_\alpha(X))}{h^2(T_\alpha(X))} = \frac{\beta\, g(X) + (2\beta^2 + 2b_2\beta + b_4)h^2(X)}{g(X) - \beta h^2(X)}, \tag{8.2}$$

et, pour les ordonnées, par

$$\frac{3\alpha^2 + 2a_2\alpha + a_4}{(X-\alpha)^2}\frac{k(T_\alpha(X))}{h^3(T_\alpha(X))} = \frac{(3\beta^2 + 2b_2\beta + b_4)k(X)h(X)}{(g(X) - \beta h^2(X))^2}. \tag{8.3}$$

Exploitons tout d'abord l'égalité des abscisses. L'irréductibilité de l'abscisse de $\mathcal{I}$ induit celle de la partie gauche de l'équation (8.2) et donc numérateurs et dénominateurs de cette dernière sont proportionnels. La constante de proportionnalité est simplement $h^2(\alpha)$ car le coefficient de plus haut degré de $(X - \alpha)^d h(T_\alpha(X))$ est $h(\alpha)$. Nous obtenons

$$\begin{cases} (X - \alpha)^\ell g(T_\alpha(X)) &= h^2(\alpha)\left(\beta\, g(X) + (2\beta^2 + 2b_2\beta + b_4)h^2(X)\right), \\ (X - \alpha)^\ell h^2(T_\alpha(X)) &= h^2(\alpha)(g(X) - \beta h^2(X)). \end{cases} \tag{8.4}$$

Substituons maintenant X par $T_\alpha(X)$ dans la première équation. Nous trouvons

$$(3\alpha^2+2a_2\alpha+a_4)^\ell g(X) = h^2(\alpha)(\beta(X-\alpha)^\ell g(T_\alpha(X)) \\ + (2\beta^2+2b_2\beta+b_4)(X-\alpha)^\ell h^2(T_\alpha(X)))$$

et, en égalant les coefficients de plus haut degré, nous avons,

$$(3\alpha^2+2a_2\alpha+a_4)^\ell = h^2(\alpha)(\beta g(\alpha)+(2\beta^2+2b_2\beta+b_4)h^2(\alpha)).$$

Comme $g(\alpha)=\beta h^2(\alpha)$, nous avons

$$h^4(\alpha) = \frac{(3\alpha^2+2a_2\alpha+a_4)^\ell}{3\beta^2+2b_2\beta+b_4},$$

ce qui en conjonction avec la deuxième équation du système (8.4), suffit à montrer le point 1 de la proposition.

En réécrivant maintenant l'équation (8.3) à l'aide de l'équation (8.1), nous trouvons

$$(3\alpha^2+2a_2\alpha+a_4)(X-\alpha)^{3d}k(T_\alpha(X))h(T_\alpha(X)) = (3\beta^2+2a_2\beta+a_4)k(X)h(X). \tag{8.5}$$

Or, comme toute racine dans $\bar{\mathbb{F}}_{p^n}$ de $h(X)$ est l'abscisse d'un point de $\mathrm{Ker}\mathcal{I}$ et que toute racine de $(X-\alpha)^d h(T_\alpha(X))$ est l'abscisse du translaté d'un point de $\mathrm{Ker}\mathcal{I}$ par P_α, $h(X)$ et $(X-\alpha)^d h(T_\alpha(X))$ sont des polynômes premiers entre eux et ainsi $(X-\alpha)^d h(T_\alpha(X))$ divise $k(X)$. Nous écrivons

$$k(X) = (X-\alpha)^d h(T_\alpha(X))K(X)$$

et la simplification de l'équation (8.5) qui s'en déduit démontre le point 2 de la proposition. □

L'indétermination du polynôme $K(X)$ dans le théorème précédent n'est finalement qu'un leurre si nous considérons l'ensemble des points d'ordre 2 de $E_a(\bar{\mathbb{F}}_{p^n})$. En effet, nous pouvons déduire de la proposition 28 le résultat suivant.

Corollaire 8. *Si, avec les notations de la proposition 28, on note P_{α_1}, P_{α_2} et P_{α_3} les trois points d'ordre 2 de $E_a(\bar{\mathbb{F}}_{p^n})$, alors*

$$k(X) = (X-\alpha_1)^d \frac{h(T_{\alpha_1}(X))}{h(\alpha_1)}(X-\alpha_2)^d \frac{h(T_{\alpha_2}(X))}{h(\alpha_2)}(X-\alpha_3)^d \frac{h(T_{\alpha_3}(X))}{h(\alpha_3)}.$$

Démonstration : L'application de la proposition 28 aux trois points d'ordre 2 montre que les polynômes $(X-\alpha_1)^d h(T_{\alpha_1}(X))$, $(X-\alpha_2)^d h(T_{\alpha_2}(X))$ et $(X-\alpha_3)^d h(T_{\alpha_3}(X))$ divisent $k(X)$. Il ne reste plus qu'à écrire que $k(X)$ est unitaire pour conclure. □

8.1.2 Application

Pour appliquer les idées de la section précédente à une courbe E_a, regardons plus en détail la structure des points d'ordre 2 de $E_a(\mathbb{F}_{p^n})$. Ainsi, contrairement aux corps de caractéristique 2, $E_a[2]$ possède trois points *distincts* (puisque $\Delta_{E_a} \neq 0$). Nous avons donc trois structures possibles :

- $E_a(\mathbb{F}_{p^n}) \cap E_a[2] = \{O_{E_a}\}$,
- $E_a(\mathbb{F}_{p^n}) \cap E_a[2] = \{O_{E_a}, P_\alpha\}$,
- $E_a(\mathbb{F}_{p^n}) \cap E_a[2] = \{O_{E_a}, P_{\alpha_1}, P_{\alpha_2}, P_{\alpha_3}\}$.

Nous montrons dans la suite comment utiliser la proposition 28 pour chacun de ces cas, l'idée directrice étant de toujours se placer dans $E_a[2]$.

Aucun point d'ordre 2

Avec les notations de la proposition 28, nous plongeons la courbe E_a dans l'extension $\mathbb{T}$ de $\mathbb{F}_q$ ($q = p^n$) sur laquelle est définie $E_a[2]$, c'est-à-dire

$$\mathbb{T} = \mathbb{F}_q[\alpha]/(\alpha^3 + a_2\alpha^2 + a_4\alpha + a_6).$$

Ainsi, $E_a(\mathbb{T})$ possède trois points d'ordre 2 d'abscisses α et ses conjugués,

$$P_\alpha = (\alpha, 0), P_{\alpha^q} = (\alpha^q, 0) \text{ et } P_{\alpha^{q^2}} = (\alpha^{q^2}, 0).$$

De même, $E_b(\mathbb{T})$ possède trois points d'ordre 2 d'abscisses β (que nous prenons comme l'une des racines de $X^3 + b_2X^2 + b_4X + b_6$ dans $\mathbb{T}$) et ses conjugués,

$$Q_\beta = (\beta, 0), Q_{\beta^q} = (\beta^q, 0) \text{ et } Q_{\beta^{q^2}} = (\beta^{q^2}, 0).$$

Appliquons maintenant la proposition 28 à P_α en supposant $\mathcal{I}(P_\alpha) = Q_\beta$. Nous trouvons

$$g(X) = (X - \alpha)\frac{(X - \alpha)^{2d}h^2(T_\alpha(X))}{h^2(\alpha)} + \beta h^2(X). \tag{8.6}$$

Pour tirer parti de cette égalité, écrivons comme au chapitre 7,

$$\begin{aligned} h^2(X) &= X^{2d} + \mathfrak{h}_{2d-1}X^{2d-1} + \cdots + \mathfrak{h}_0, \\ g(X) &= X^{2d+1} + g_{2d}X^{2d} + \cdots + g_0, \end{aligned}$$

et substituons ces expressions à $h^2(X)$ et $g(X)$ dans l'équation (8.6). On peut alors en écrire la partie droite dans $\mathbb{T}$ sous la forme $g(X) = R_1(X) + \alpha R_\alpha(X) + \alpha^2 R_{\alpha^2}(X)$. Comme $g(X)$ est un polynôme de $\mathbb{F}_q[X]$, nous avons en fait

- d'une part, $g(X) = R_1(X)$, ce qui nous permet, comme annoncé, d'exprimer linéairement les coefficients de $g(X)$ en fonction de ceux de $h(X)$,
- d'autre part, $R_\alpha(X) = 0$ et $R_{\alpha^2}(X) = 0$, ce qui nous donne un système linéaire satisfait par les coefficients $\mathfrak{h}_i$ du polynôme $h^2(X)$.

Nous avons noté que ce dernier système linéaire à $4d+4$ équations et $2d-2$ inconnues $\mathfrak{h}_i$ était en pratique environ de rang d, ce qui ne nous permet pas de déterminer complètement $h^2(X)$. Bien entendu, nous avons cherché à tirer parti du fait que $h^2(X)$ doit être un carré, mais en substance, cela nous conduit à un système de d équations quadratiques en $\mathfrak{h}_i$ sur $\mathbb{F}_q$ pour lequel nous ne connaissons pas d'algorithme efficace de résolution. C'est pourquoi nous ne sommes pas en mesure aujourd'hui de proposer une méthode de résolution complète basée sur ces idées. Remarquons cependant que ce système n'a pas de solution lorsque la courbe E_b n'est pas isogène à E_a.

Bien entendu, $\mathcal{I}$ pourrait envoyer P_α, non pas sur Q_β mais sur Q_{β^q} ou sur $Q_{\beta^{q^2}}$. Lorsque nous nous trompons, nous ne disposons pas "a priori" de moyen pour le détecter, mais nous verrons dans la section 8.2 comment, en conjonction avec les algorithmes de Couveignes, il est possible de s'en rendre compte "a posteriori". Dans ce cas il suffit de remplacer β par l'un de ses conjugués. Nous avons ainsi un maximum de trois essais à effectuer.

Enfin, un dernier problème est la détermination de $h^2(\alpha)$, ou plus précisément du signe ϵ dans l'égalité

$$h^2(\alpha) = \epsilon\sqrt{\frac{(3\alpha^2 + 2a_2\alpha + a_4)^\ell}{3\beta^2 + 2b_2\beta + b_4}}, \ (\epsilon = \pm 1).$$

Lorsque $q \equiv 3 \bmod 4$, ϵ est simplement choisi pour que $\epsilon\sqrt{\frac{(3\alpha^2+2a_2\alpha+a_4)^\ell}{3\beta^2+2b_2\beta+b_4}}$ soit lui-même un carré. Lorsque $q \equiv 1 \bmod 4$, les deux valeurs potentielles pour $h^2(\alpha)$ ont elles-mêmes des racines carrées. Pour trouver ϵ dans ce cas, nous partons de la relation (8.4) obtenue à partir de P_{α^q},

$$g(X) - \beta^q h^2(X) = (X - \alpha^q)\frac{(X - \alpha^q)^{2d}h^2(T_{\alpha^q}(X))}{h^2(\alpha^q)},$$

que nous évaluons en $X = \alpha$. Comme $g(\alpha) = \beta h^2(\alpha)$ et $T_{\alpha^q}(\alpha) = \alpha^{q^2}$, nous trouvons

$$\epsilon(\beta - \beta^q)\sqrt{\frac{(3\alpha^2 + 2a_2\alpha + a_4)^\ell}{3\beta^2 + 2b_2\beta + b_4}} = (\alpha - \alpha^q)^\ell \left(\sqrt{\frac{(3\alpha^{2q} + 2a_2\alpha^q + a_4)^\ell}{3\beta^{2q} + 2b_2\beta^q + b_4}}\right)^{q-1}$$

et cette équation n'est vérifiée que pour une seule valeur de ϵ.

Un point d'ordre 2

L'idée, toujours la même, est de travailler dans l'extension de plus petit degré contenant $E_a[2]$. Pour cela, notons $P_a = (a, 0)$ le point d'ordre 2 de $E_a(\mathbb{F}_q)$ et $P_b = (b, 0)$ le point d'ordre 2 de $E_b(\mathbb{F}_q)$. Alors $X^3 + a_2X^2 + a_4X + a_6 = (X - a)(X^2 + (a_2 + a)X + a^2 + aa_2 + a_4)$ et nous plongeons E_a dans

$$\mathbb{T} = \mathbb{F}_q[\alpha]/(\alpha^2 + (a_2 + a)\alpha + a^2 + aa_2 + a_4).$$

Nous avons ainsi $E_a[2] = \{O_{E_a}, P_a, P_\alpha, P_{\alpha^q}\}$. De même, soit β une racine dans $\mathbb{T}$ du polynôme $X^3 + b_2X^2 + b_4X + b_6$ distincte de b, nous avons $E_b[2] = \{O_{E_b}, Q_b, Q_\beta, Q_{\beta^q}\}$.

Contrairement au cas précédent, nous savons ici que $\mathcal{I}(P_a) = Q_b$. Nous écrivons l'équation (8.4) correspondante,

$$g(X) = (X - a)\frac{(X - a)^{2d}h^2(T_a(X))}{h^2(a)} + bh^2(X)$$

d'où nous déduisons, une fois connu $h^2(a)$, les coefficients g_i de $g(X)$ linéairement en fonction des coefficients $\mathfrak{h}_i$ du polynôme $h^2(X)$.

À ce stade, il est alors nécessaire comme précédemment de supposer que $\mathcal{I}(P_\alpha) = Q_\beta$ quitte, si l'on s'est trompé, à tester ensuite si $\mathcal{I}(P_\alpha) = Q_{\beta^q}$. Il est alors possible de déterminer le signe ϵ_a de $h^2(a)$ quand $q \equiv 1 \bmod 4$. Pour cela, nous procédons comme précédemment : nous supposons que $\mathcal{I}(P_\alpha) = Q_\beta$ et écrivons l'équation (8.4) pour P_α dans laquelle nous substituons a à X, obtenant ainsi

$$\epsilon_a\sqrt{\frac{(3a^2 + 2a_2a + a_4)^\ell}{3b^2 + 2b_2b + b_4}} = \frac{(a - \alpha)^\ell}{b - \beta}\left(\sqrt{\frac{(3\alpha^{2q} + 2a_2\alpha^q + a_4)^\ell}{3\beta^{2q} + 2b_2\beta^q + b_4}}\right)^{q-1}.$$

Pour trouver des relations linéaires entre les coefficients $\mathfrak{h}_i$, nous appliquons ensuite la proposition 28 à P_α ,

$$g(X) = (X - \alpha)\frac{(X - \alpha)^{2d}h^2(T_\alpha(X))}{h^2(\alpha)} + \beta h^2(X)$$

(celle obtenue à partir de P_{α^q} étant la conjuguée de celle-ci). Une fois projetée dans la base $\{1, \alpha\}$ et $\{1, X, \ldots, X^{2d}\}$, nous avons $4d + 4$ équations pour $2d - 2$ inconnues $\mathfrak{h}_i$ et il s'avère encore en pratique que ce système

- n'a pas de solution si E_b n'est pas isogène à E_a,
- a un rang de l'ordre de d sinon,
- n'est valide que si $\mathcal{I}(P_\alpha) = Q_\beta$. Si ce n'est pas le cas, il faut recommencer avec $\mathcal{I}(P_\alpha) = Q_{\beta^q}$.

D'autre part, nous n'avons pas cette fois trouvé de moyen pour déterminer "a priori" le signe ϵ_α de $h^2(\alpha)$ quand $q \equiv 1 \bmod 4$. Nous essayons donc $\epsilon_\alpha = 1$ et s'il s'avère qu'il s'agit d'une hypothèse fausse, le système obtenu entre les coefficients $\mathfrak{h}_i$ n'a pas de solution. Dans ce cas nous rectifions en posant $\epsilon_\alpha = -1$.

Trois points d'ordre 2

Lorsque $E_a(\mathbb{F}_q)$ a trois points d'ordre 2, P_{α_1}, P_{α_2} et P_{α_3}, il est bien sûr possible d'appliquer un algorithme similaire à ceux des cas précédents. Néanmoins, la situation est moins favorable. En effet il est ici nécessaire de choisir arbitrairement l'image par $\mathcal{I}$ non seulement de P_{α_1}, mais aussi celle de P_{α_2} car les racines α_1, α_2 et α_3 de $X^3 + a_2X^2 + a_4X + a_6$ ne se déduisent pas les unes des autres par

conjugaison. De plus, lorsque $q \equiv 1 \bmod 4$, le seul moyen dont nous disposons pour trouver les signes de $h^2(\alpha_1)$, $h^2(\alpha_2)$ et $h^2(\alpha_3)$ est de tester si le système satisfait par les coefficients $\mathfrak{h}_i$ admet une solution. En regardant au plus près, nous nous apercevons que nous avons un maximum de 24 essais, soit 12 essais en moyenne, avant de réussir à exprimer les coefficients g_i linéairement en fonction de ceux de $\mathfrak{h}_i$ puis de trouver un système de d équations linéaires entre les $\mathfrak{h}_i$.

Nous préférons pour notre part utiliser une autre méthode. Il se trouve en effet qu'une courbe isogène de degré 2 à E_a peut avoir une structure de groupe différente de celle de E_a. Prenons par exemple dans $\mathbb{F}_{1009}$, la courbe $Y^2 = X^3 + 539X + 775$ tirée de [33]. Elle possède trois points d'ordre 2 ($(323, 0)$, $(729, 0)$ et $(966, 0)$) et elle est isogène de degré 2 à la courbe $Y^2 = X^3 + 356X + 872$ qui n'a qu'un unique point d'ordre 2, à savoir $(560, 1)$. L'idée est donc en général de trouver dans le graphe des courbes isogènes E'_a de degré 2^k à E_a, une courbe qui ne possède qu'un unique point d'ordre 2 et ainsi se ramener au cas précédent. Nous n'avons bien sûr aucune estimation théorique précise sur k, néanmoins, k est toujours petit en pratique, disons de l'ordre de la dizaine, puisque ce graphe paraît très peu "compact". Nous illustrons ce phénomène sur l'exemple de la figure 8.1 où un nombre représente l'invariant d'une courbe elliptique et un trait matérialise l'existence d'une isogénie de degré 2.

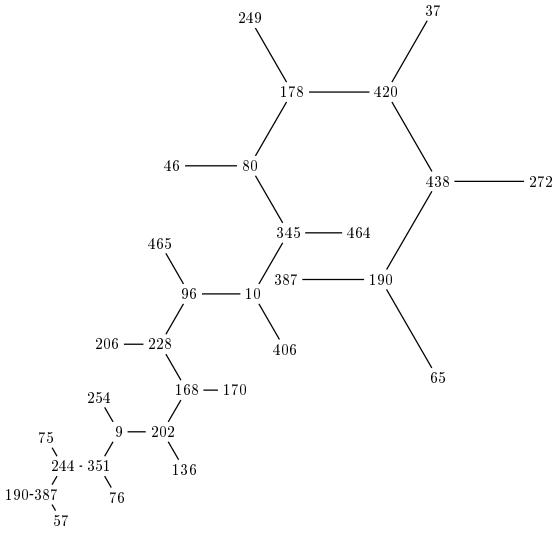


FIG. 8.1 – Graphe des courbes 2^k-isogènes à $E_a : Y^2 = X^3 + 360X + 462$ ($j_{E_a} = 190$, $\#E_a = 520$) dans $\mathbb{F}_{503}$.

Dans l'algorithme `SEA`, cette méthode est particulièrement aisée à mettre en œuvre puisqu'il suffit de remplacer le calcul de la cardinalité d'une courbe E_a ayant trois points d'ordre 2 par celui de la cardinalité d'une courbe isogène de degré 2^k à E'_a et ne possédant qu'un point d'ordre 2 (rappelons que deux courbes isogènes ont même cardinalité). Les nombreuses courbes E'_b isogènes à E'_a de degrés impairs qui sont alors calculées au cours du déroulement de `SEA` auront aussi un unique point d'ordre 2. En effet, comme toute isogénie $\mathcal{I}$ de degré ℓ impair ne peut pas avoir de point d'ordre deux dans son noyau, il n'est pas difficile de montrer que les trois images des points d'ordre 2 de E'_a sont distinctes. Or cette propriété ne peut être satisfaite si E'_a et E'_b disposent respectivement d'un et trois points d'ordre deux sur $\mathbb{F}_q$.

8.2 Connexion entre petite et grande caractéristique

Avec les notations de la proposition 28, nous expliquons tout d'abord comment tirer parti de la connaissance de relations linéaires entre coefficients de $g(x)$ et de $h^2(X)$ pour diminuer dans les algorithmes de Couveignes la taille des objets à calculer par un facteur au moins égal à quatre. Nous en déduisons un algorithme qui fait la connexion entre les méthodes connues pour le calcul d'isogénie dans

des corps finis de petite caractéristique avec celles pour le cas de la grande caractéristique. Puis nous finissons en donnant une amélioration heuristique de ce schéma à l'aide des formules de Vélu, ce qui permet en pratique un calcul d'isogénies uniquement avec les idées de ce chapitre lorsque $\ell < 2p$.

8.2.1 Principe

Le cœur du premier algorithme de Couveignes (cf. chapitre 5) est d'identifier une série $A(Z)$ de $2\ell+2$ termes, $A(Z) = \sum_{k=0}^{2\ell+2} a_k Z^k$, à la fraction rationnelle

$$\frac{F(Z)}{G(Z)} = Z\frac{Z^{2d}h^2(1/Z)}{X^\ell g(1/Z)}.$$

Ce qui revient à résoudre le système

$$\begin{array}{rcll} \sum_{i=0}^{k} g_{\ell-i}a_{k-i} & = & \mathfrak{h}_{2d-k+1} & \text{si } 0 \leqslant k \leqslant \ell, \\ \sum_{i=0}^{\ell} g_{\ell-i}a_{k-i} & = & 0 & \text{si } k > \ell. \end{array} \tag{8.7}$$

En utilisant les résultats de la section 8.1, nous avons d'une part, les coefficients g_i de $g(X)$ qui s'écrivent comme combinaisons linéaires des coefficients $\mathfrak{h}_i$ de $h^2(X)$ et, d'autre part, la moitié des coefficients de $h^2(X)$ (par exemple $\mathfrak{h}_d, \ldots, \mathfrak{h}_{2d}$) qui s'expriment aussi linéairement en fonction des autres (par exemple $\mathfrak{h}_0, \ldots, \mathfrak{h}_{d-1}$). En injectant ces résultats dans le système (8.7), il nous reste donc moins de d inconnues $\mathfrak{h}_i$ à calculer et au final, seule la connaissance de d premiers termes de $A(Z)$ est nécessaire. Nous avons donc besoin de travailler avec des séries qui ont quatre fois moins de termes que dans l'algorithme original.

Un raisonnement analogue s'applique pour la phase d'interpolation du deuxième algorithme de Couveignes (cf. chapitre 6).

8.2.2 Algorithme générique

À partir de la remarque précédente, il n'est pas difficile de déduire un algorithme générique applicable à n'importe quel corps fini de caractéristique p impaire pour calculer une isogénie de degré ℓ impair. Celui-ci procède comme suit :

1. obtention de $\min(p-1, d)$ coefficients de $h(X)$ avec l'algorithme `CCR` du chapitre 4. Si $h(X)$ est complètement déterminé, nous stoppons.
2. pour chacune des images potentielles de $\mathcal{I}(P_\alpha)$, nous effectuons :
   (a) le calcul des coefficients de $g(X)$ linéairement en fonction de ceux de $h^2(X)$.
   (b) le calcul de la moitié des coefficients de $h^2(X)$ linéairement en fonction des autres (par exemple $\mathfrak{h}_d, \ldots, \mathfrak{h}_{2d}$ en fonction de $\mathfrak{h}_0, \ldots, \mathfrak{h}_{d-1}$).
   (c) l'appel du premier ou deuxième algorithme de Couveignes pour calculer les $d-p$ coefficients restants de $h^2(X)$. Si cet algorithme n'a pas trouvé d'isogénies, nous retournons au point 2. Pour tester une autre image de P_α par $\mathcal{I}$. Sinon, nous stoppons.

Avec par exemple le premier algorithme de Couveignes, les séries que nous manipulons sont ainsi au moins quatre fois plus petites. Or, la complexité de l'algorithme étant en $O(\ell^3)$, nous pouvons espérer un gain de $4^3 = 64$. Néanmoins, $1,5$ images en moyenne sont à tester pour P_α, ce qui se traduit par $1,5$ appels à l'algorithme de Couveignes et ramène ce gain à environ 40.

8.2.3 Exploiter les formules de Vélu

Parmi nos nombreuses tentatives pour obtenir un algorithme complet de la proposition 28, une relation obtenue à partir des formules de Vélu nous apporte des relations linéaires supplémentaires entre

les coefficients de $h^2(X)$. Plus précisément, avec les notations de la proposition 28, nous obtenons à partir des formules de Vélu du chapitre 4 (cf. page 54),

$$8(X^3+a_2X^2+a_4X+a_6)(h'(X))^2 = g(X)+(X^3+a_2X^2+a_4X+a_6)(h^2(X))''+ \\ (3X^2+2a_2X+a_4)(h^2(X))'+(2p_1-\ell X)h^2(X).$$

Notons que la partie droite de cette équation est un polynôme à coefficients linéaires en les coefficients $\mathfrak{h}_i$ de $h^2(X)$. La division par $X^3+a_2X^2+a_4X+a_6$ fournit une linéaire supplémentaire entre les $\mathfrak{h}_i$ en écrivant que le reste de cette division doit être nul. Nous obtenons $(h'(X))^2$ linéairement en fonction de $h^2(X)$.

Nous tirons alors parti de cette linéarité en écrivant simplement

$$((h^2)')^2 = 4(h')^2h^2.$$

Cette dernière relation est certes quadratique, mais nous observons en pratique qu'environ p termes de hauts degrés s'expriment linéairement en les $\mathfrak{h}_i$. Il ne nous reste plus qu'à ajouter ces relations à celles déjà collectées. Lorsque $\ell < 2p$, nous disposons d'assez de relations linéaires pour trouver directement $h^2(X)$. Sinon, ces relations permettent de diminuer d'autant la taille des séries à considérer dans le premier algorithme de Couveignes.

8.3 Exemple

Nous donnons en exemple le calcul d'une isogénie de degré 17 dans $\mathbb{F}_{5^2}$ qui met en œuvre l'algorithme précédent. Nous allons chercher l'abscisse $g(X)/h^2(X)$ d'une isogénie $\mathcal{I}$ de degré $\ell = 17$ entre deux courbes E_a et E_b définies sur

$$\mathbb{F}_{5^2} = \mathbb{F}_5[t]/(t^2+2),$$

par

$$E_a \ : Y^2 = X^3+\overline{5}X+\overline{9} \text{ et } E_b \ : Y^2 = X^3+\overline{5}X+\overline{18},$$

avec $p_1 = \overline{2}$ (déterminé comme expliqué au chapitre 3). Nous utilisons la notation habituelle, $\overline{\tau_0+\tau_1 5} = \tau_0+\tau_1 t$.

Pour $p = 5$, les formules de Vélu ne nous sont d'aucune aide. Nous nous préoccupons donc tout d'abord des points d'ordre deux sur E_a. La courbe $E_a(\mathbb{F}_{5^2})$ (resp. $E_b(\mathbb{F}_{5^2})$) possède un unique point d'ordre deux, $P_{\overline{10}} = (\overline{10}, 0)$ (resp. $Q_{\overline{20}} = (\overline{20}, 0)$). Plongeons E_a et E_b dans

$$\mathbb{T} = \mathbb{F}_{5^2}[\alpha]/(\alpha^2+\overline{10}\alpha+\overline{7}),$$

et nous obtenons ainsi les deux autres points d'ordre deux de $E_a[2]$ (resp. $E_b[2]$), $P_\alpha = (\alpha, 0)$ et $P_{\alpha^q} = (\overline{4}\alpha+\overline{15}, 0)$ (resp. $Q_\beta = (\overline{22}\alpha+\overline{2}, 0)$ et $Q_{\beta^q} = (\overline{8}\alpha+\overline{8}, 0)$).

Nous sommes ici dans le cas où $\mathcal{I}(P_{\overline{10}}) = Q_{\overline{20}}$ et donc l'équation (8.1) s'écrit

$$g(X) = (X-\overline{10})\frac{(X-\overline{10})^{2d}h^2\left(\frac{\overline{10}X+\overline{9}}{X-\overline{10}}\right)}{h^2(\overline{10})}+\overline{20}h^2(X).$$

Avant d'exploiter cette relation, il est nécessaire de calculer $h^2(\overline{10})$. À un "signe" ϵ près puisque $p = 1 \bmod 4$, nous avons déjà

$$h^2(\overline{10}) = \epsilon\sqrt{\frac{(2(\overline{10})^2+\overline{5})^{17}}{2(\overline{20})+\overline{5}}} = \epsilon. \tag{8.8}$$

Puis, en fixant par exemple $\mathcal{I}(P_\alpha) = Q_\beta$, nous trouvons ϵ par

$$\epsilon = \frac{(\alpha-\alpha^q)^{17}}{\beta-\beta^q}\left(\sqrt{\frac{(3\alpha^2+\overline{5})^{17}}{3\beta^2+\overline{5}}}\right)^{24} = 1.$$

Nous substituons maintenant à $g(X)$ et $h^2(X)$ les expressions

$$h^2(X) = X^{16} - 2p_1X^{15} + \mathfrak{h}_{14}X^{14} + \cdots + \mathfrak{h}_0 \text{ et } g(X) = X^{17} + g_{16}X^{16} + \cdots + g_0.$$

dans l'équation (8.8) et nous trouvons le système

$$\begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ g_6 \\ g_7 \\ g_8 \\ g_9 \\ g_{10} \\ g_{11} \\ g_{12} \\ g_{13} \\ g_{14} \\ g_{15} \\ g_{16} \\ 1 \end{bmatrix} = \begin{bmatrix} \overline{10} & \overline{9} & \overline{6} & \overline{15} & \overline{9} & \overline{6} & \overline{15} & \overline{9} & \overline{6} & \overline{15} & \overline{9} & \overline{6} & \overline{15} & \overline{9} & \overline{6} \\ \overline{2} & \overline{12} & \overline{17} & \overline{11} & \overline{6} & \overline{19} & \overline{20} & \overline{20} & \overline{16} & \overline{9} & \overline{14} & \overline{18} & \overline{18} & \overline{3} & \overline{15} \\ \overline{20} & \overline{4} & \overline{10} & \overline{23} & \overline{0} & \overline{18} & \overline{8} & \overline{6} & \overline{4} & \overline{0} & \overline{17} & \overline{23} & \overline{9} & \overline{8} & \overline{0} \\ \overline{0} & \overline{0} & \overline{0} & \overline{22} & \overline{16} & \overline{0} & \overline{0} & \overline{0} & \overline{19} & \overline{3} & \overline{0} & \overline{0} & \overline{0} & \overline{14} & \overline{11} \\ \overline{0} & \overline{0} & \overline{0} & \overline{10} & \overline{23} & \overline{0} & \overline{0} & \overline{0} & \overline{24} & \overline{11} & \overline{0} & \overline{0} & \overline{0} & \overline{21} & \overline{16} \\ \overline{2} & \overline{14} & \overline{19} & \overline{3} & \overline{21} & \overline{16} & \overline{24} & \overline{10} & \overline{18} & \overline{7} & \overline{11} & \overline{3} & \overline{16} & \overline{18} & \overline{8} \\ \overline{5} & \overline{7} & \overline{7} & \overline{23} & \overline{12} & \overline{16} & \overline{7} & \overline{12} & \overline{14} & \overline{14} & \overline{18} & \overline{4} & \overline{4} & \overline{21} & \overline{24} \\ \overline{1} & \overline{10} & \overline{2} & \overline{24} & \overline{3} & \overline{13} & \overline{7} & \overline{16} & \overline{8} & \overline{11} & \overline{8} & \overline{6} & \overline{11} & \overline{6} & \overline{16} \\ \overline{0} & \overline{0} & \overline{0} & \overline{20} & \overline{13} & \overline{0} & \overline{0} & \overline{0} & \overline{17} & \overline{4} & \overline{0} & \overline{0} & \overline{0} & \overline{22} & \overline{4} \\ \overline{0} & \overline{0} & \overline{0} & \overline{2} & \overline{5} & \overline{0} & \overline{0} & \overline{0} & \overline{12} & \overline{3} & \overline{0} & \overline{0} & \overline{0} & \overline{12} & \overline{8} \\ \overline{10} & \overline{21} & \overline{24} & \overline{20} & \overline{18} & \overline{15} & \overline{9} & \overline{6} & \overline{1} & \overline{13} & \overline{4} & \overline{6} & \overline{15} & \overline{15} & \overline{10} \\ \overline{3} & \overline{13} & \overline{13} & \overline{15} & \overline{3} & \overline{2} & \overline{17} & \overline{17} & \overline{17} & \overline{8} & \overline{14} & \overline{13} & \overline{18} & \overline{9} & \overline{17} \\ \overline{5} & \overline{1} & \overline{10} & \overline{12} & \overline{5} & \overline{20} & \overline{4} & \overline{15} & \overline{1} & \overline{8} & \overline{17} & \overline{23} & \overline{4} & \overline{11} & \overline{8} \\ \overline{0} & \overline{0} & \overline{0} & \overline{1} & \overline{23} & \overline{0} & \overline{0} & \overline{0} & \overline{15} & \overline{21} & \overline{0} & \overline{0} & \overline{0} & \overline{22} & \overline{16} \\ \overline{0} & \overline{0} & \overline{0} & \overline{5} & \overline{4} & \overline{0} & \overline{0} & \overline{0} & \overline{4} & \overline{10} & \overline{0} & \overline{0} & \overline{0} & \overline{10} & \overline{23} \\ \overline{2} & \overline{14} & \overline{19} & \overline{0} & \overline{15} & \overline{5} & \overline{13} & \overline{12} & \overline{0} & \overline{2} & \overline{4} & \overline{23} & \overline{8} & \overline{0} & \overline{5} \\ \overline{5} & \overline{7} & \overline{7} & \overline{8} & \overline{3} & \overline{4} & \overline{9} & \overline{9} & \overline{24} & \overline{20} & \overline{10} & \overline{14} & \overline{14} & \overline{11} & \overline{1} \\ \overline{1} & \overline{10} & \overline{2} & \overline{20} & \overline{4} & \overline{15} & \overline{3} & \overline{5} & \overline{1} & \overline{10} & \overline{2} & \overline{20} & \overline{4} & \overline{15} & \overline{3} \end{bmatrix} \begin{bmatrix} \mathfrak{h}_0 \\ \mathfrak{h}_1 \\ \mathfrak{h}_2 \\ \mathfrak{h}_3 \\ \mathfrak{h}_4 \\ \mathfrak{h}_5 \\ \mathfrak{h}_6 \\ \mathfrak{h}_7 \\ \mathfrak{h}_8 \\ \mathfrak{h}_9 \\ \mathfrak{h}_{10} \\ \mathfrak{h}_{11} \\ \mathfrak{h}_{12} \\ \mathfrak{h}_{13} \\ \mathfrak{h}_{14} \end{bmatrix} + \begin{bmatrix} \overline{24} \\ \overline{19} \\ \overline{24} \\ \overline{0} \\ \overline{0} \\ \overline{20} \\ \overline{4} \\ \overline{20} \\ \overline{0} \\ \overline{0} \\ \overline{13} \\ \overline{7} \\ \overline{13} \\ \overline{0} \\ \overline{0} \\ \overline{1} \\ \overline{6} \\ \overline{6} \end{bmatrix}.$$

Comme $g(X)$ est unitaire, nous avons une première relation linéaire entre les $\mathfrak{h}_i$:

$$\mathfrak{h}_{14} = 3\mathfrak{h}_0 + 5\mathfrak{h}_1 + \mathfrak{h}_2 + 10\mathfrak{h}_3 + 2\mathfrak{h}_4 + 20\mathfrak{h}_5 + 4\mathfrak{h}_6 + 15\mathfrak{h}_7 + 3\mathfrak{h}_8 + 5\mathfrak{h}_9 + \mathfrak{h}_{10} + 10\mathfrak{h}_{11} + 2\mathfrak{h}_{12} + 20\mathfrak{h}_{13} + 15 \ ;$$

elle nous permet d'exprimer $g(X)$ en fonction de 14 inconnues $\mathfrak{h}_0, \ldots, \mathfrak{h}_{13}$.

En supposant maintenant $\mathcal{I}(P_\alpha) = Q_\beta$, l'équation (8.1) conduit à

$$g(X) = (X-\alpha)\frac{(X-\alpha)^{16}h^2\left(\frac{\alpha X + 2\alpha^2 + \overline{5}}{X-\alpha}\right)}{h^2(\alpha)} + \beta h^2(X) \tag{8.9}$$

avec $h^2(\alpha) = \epsilon'\sqrt{\dfrac{(3\alpha^2+\overline{5})^{17}}{3\beta^2+\overline{5}}} = \epsilon'(\overline{6}\alpha + \overline{12})$ où $\epsilon' = \pm 1$. Choisissons $\epsilon' = 1$ et substituons à $g(X)$ et $h^2(X)$ leurs expressions fonctions de $\mathfrak{h}_0, \ldots, \mathfrak{h}_{13}$ dans l'équation (8.9). Nous obtenons un système linéaire qui n'admet pas de solution. Nous avons donc $\epsilon' = -1$ et cette fois-ci l'équation (8.9) conduit à un système de 36 équations à 14 inconnues, mais de rang seulement 9. Cela nous donne finalement

$$\begin{bmatrix} \mathfrak{h}_{14} \\ \mathfrak{h}_{13} \\ \mathfrak{h}_{12} \\ \mathfrak{h}_{11} \\ \mathfrak{h}_{10} \\ \mathfrak{h}_9 \\ \mathfrak{h}_8 \\ \mathfrak{h}_7 \\ \mathfrak{h}_6 \\ \mathfrak{h}_5 \end{bmatrix} = \begin{bmatrix} \overline{13} & \overline{10} & \overline{10} & \overline{14} & \overline{19} \\ \overline{9} & \overline{5} & \overline{5} & \overline{19} & \overline{1} \\ \overline{10} & \overline{0} & \overline{4} & \overline{3} & \overline{13} \\ \overline{9} & \overline{22} & \overline{10} & \overline{9} & \overline{12} \\ \overline{1} & \overline{18} & \overline{0} & \overline{12} & \overline{24} \\ \overline{22} & \overline{14} & \overline{14} & \overline{24} & \overline{8} \\ \overline{10} & \overline{18} & \overline{18} & \overline{21} & \overline{8} \\ \overline{6} & \overline{15} & \overline{1} & \overline{14} & \overline{18} \\ \overline{22} & \overline{23} & \overline{22} & \overline{16} & \overline{15} \\ \overline{8} & \overline{5} & \overline{10} & \overline{6} & \overline{15} \end{bmatrix} \begin{bmatrix} \mathfrak{h}_0 \\ \mathfrak{h}_1 \\ \mathfrak{h}_2 \\ \mathfrak{h}_3 \\ \mathfrak{h}_4 \end{bmatrix} + \begin{bmatrix} \overline{12} \\ \overline{6} \\ \overline{17} \\ \overline{6} \\ \overline{24} \\ \overline{1} \\ \overline{21} \\ \overline{15} \\ \overline{21} \\ \overline{6} \end{bmatrix}.$$

Exploitons maintenant les formules de Vélu pour exprimer les coefficients de $(h'(X))^2$ comme combinaisons linéaires de $\mathfrak{h}_0, \ldots, \mathfrak{h}_4$. Pour cela nous calculons $l(X) = 8(X^3 + \overline{5}X + \overline{9})(h'(X))^2$ et écrivons que

ce polynôme doit s'annuler en $\overline{10}$, α et α^q, ce qui nous donne une relation linéaire supplémentaire,

$$\mathfrak{h}_4 = \overline{20}\mathfrak{h}_0 + \overline{19}\mathfrak{h}_1 + \overline{17}\mathfrak{h}_2 + \overline{10}\mathfrak{h}_3 + \overline{24}.$$

Nous avons ainsi

$$(h'(X))^2 = \overline{4}X^{14} + X^{13} + (\overline{17}\mathfrak{h}_0 + \overline{8}\mathfrak{h}_1 + \overline{19}\mathfrak{h}_2 + \overline{2}\mathfrak{h}_3 + \overline{3})X^{12} \\ + (\overline{3}\mathfrak{h}_0 + \overline{15}\mathfrak{h}_1 + \overline{10}\mathfrak{h}_2 + \mathfrak{h}_3 + \overline{24})X^{11} + \cdots$$

Nous calculons les termes de plus haut degré de $((h^2)')^2 - 4(h')^2h^2$, ce qui donne

$$0 = (\overline{20}\mathfrak{h}_0 + \overline{19}\mathfrak{h}_1 + \overline{19}\mathfrak{h}_2 + \mathfrak{h}_3 + \overline{2})X^{28} + (\overline{9}\mathfrak{h}_0 + \mathfrak{h}_1 + \overline{4}\mathfrak{h}_2 + \overline{21})X^{27} + (\overline{13}\mathfrak{h}_0^2 + \overline{3}\mathfrak{h}_0\mathfrak{h}_1 \\ + \overline{23}\mathfrak{h}_0\mathfrak{h}_2 + \overline{23}\mathfrak{h}_0\mathfrak{h}_3 + \overline{14}\mathfrak{h}_1^2 + \overline{16}\mathfrak{h}_1\mathfrak{h}_2 + \overline{19}\mathfrak{h}_1\mathfrak{h}_3 + \overline{8}\mathfrak{h}_2\mathfrak{h}_3 + \overline{3}\mathfrak{h}_3^2 + \overline{21}\mathfrak{h}_1 + \overline{2}\mathfrak{h}_2 + \overline{12}\mathfrak{h}_3 + \overline{17})X^{26} + \cdots$$

et, par conséquent,

$$\begin{aligned} \mathfrak{h}_3 &= \overline{5}\mathfrak{h}_0 + \overline{11}\mathfrak{h}_1 + \overline{11}\mathfrak{h}_2 + \overline{3}, \\ \mathfrak{h}_2 &= \overline{9}\mathfrak{h}_0 + \mathfrak{h}_1 + \overline{21}. \end{aligned}$$

À ce stade, il ne nous reste plus que deux coefficients $\mathfrak{h}_0$ et $\mathfrak{h}_1$ à découvrir. Pour cela, nous utilisons le premier algorithme de Couveignes, non pas avec des séries de plus de 70 termes comme il l'aurait fallu initialement, mais avec des séries de moins de 10 termes. Le calcul donne finalement $\mathfrak{h}_0 = \overline{14}$, $\mathfrak{h}_1 = \overline{12}$ et

$$h(X) = X^8 + \overline{3}X^7 + \overline{14}X^6 + \overline{2}X^5 + \overline{3}X^4 + \overline{9}X^3 + \overline{5}X^2 + \overline{4}X + \overline{24}.$$

Troisième partie

Implantation efficace et résultats

Chapitre 9

Plus grand diviseur commun de deux entiers

Le calcul du Plus Grand Diviseur Commun de deux entiers (pgcd) est sous-jacent à de nombreuses applications : normalisation de nombres rationnels [17], inverses d'entiers modulo un nombre premier [87], calculs avec des idéaux de corps de nombres par des formes normales d'Hermite [27]...

Pour des entiers de grandes tailles, c'est-à-dire plusieurs milliers de chiffres, les meilleurs algorithmes pour calculer des pgcd reposent sur des schémas de multiplication par transformées de Fourier rapides [108, 86, 3]. Dans le cadre de cette thèse, les entiers manipulés sont bien inférieurs à cette taille et ces algorithmes ne sont pas encore efficaces.

L'algorithme communément utilisé pour effectuer ce calcul est alors l'algorithme d'Euclide [56]. Or les quotients qui y sont calculés sont égaux à 1 dans 41% des cas, cet algorithme est donc clairement inadapté à une implantation sur des ordinateurs maintenant construits autour de microprocesseurs capables d'effectuer des opérations arithmétiques avec tout entier inférieur à 2^{32}, voire à 2^{64}. C'est pourquoi des algorithmes de même complexité asymptotique que l'algorithme d'Euclide [56] mais en pratique plus efficaces sont utilisés. Dans cette annexe, nous faisons le point sur ces algorithmes[1], aussi bien pour calculer des pgcd simples (section 9.1) que des pgcd étendus (section 9.2). Nous n'en évoquons pas les applications au calcul du symbole de Jacobi, mais ces méthodes se généralisent sans difficulté dans cette optique [81].

Pour chacune de ces deux familles de pgcd, on peut facilement identifier deux types d'algorithmes : les algorithmes dits "euclidiens" et les algorithmes dits "binaires". Les algorithmes euclidiens reposent essentiellement sur la propriété

$$\forall (A,B) \in \mathbb{Z}^2,\ \mathrm{pgcd}(A,B) = \mathrm{pgcd}(B, A \bmod B). \tag{9.1}$$

La référence est un algorithme dû à Lehmer [62]. Il fut optimisé récemment par Jebelean [51] et nous en proposons à notre tour une amélioration sensible.

Les algorithmes "binaires", quant à eux, utilisent plutôt les propriétés

$$\forall (A,B,C) \in \mathbb{Z}^3,\ \begin{cases} \mathrm{pgcd}(CA,CB) &= |C|\mathrm{pgcd}(A,B), \\ \mathrm{pgcd}(A,B) &= \mathrm{pgcd}(B,A-B). \end{cases} \tag{9.2}$$

Nous décrivons l'algorithme binaire et son amélioration pour de grands entiers [56], l'algorithme introduit par Brent and Kung [16], et une généralisation récente de Jebelean [52].

Enfin, nous avons programmé chacune de ces méthodes grâce au package multiprécision `BigNum` [48] et comparons à la fin de ce chapitre les résultats obtenus (section 9.3). Il s'avère bien entendu que le gain est variable suivant les ordinateurs considérés mais il est néanmoins toujours conséquent. Par exemple, sur une station DEC, par rapport à l'algorithme d'Euclide, le calcul de pgcd simple est environ dix fois

[1] Ce travail a été réalisé dans le cadre du stage de fin d'étude de l'auteur à l'École Nationale Supérieures des Techniques Avancées [66].

plus rapide par l'algorithme binaire et le calcul de pgcd étendu est sept fois plus rapide par l'algorithme de Lehmer.

9.1 Algorithmes de pgcd simples

Des algorithmes originaux aux plus optimisés, la méthode consiste pour l'essentiel à remplacer récursivement le calcul du pgcd de deux entiers A et B par le pgcd de deux entiers plus petits A' et B'. L'efficacité dépend alors du nombre d'itérations ainsi que du temps nécessaire pour obtenir A' et B' en fonction de A et B.

9.1.1 pgcd euclidiens

Ces pgcd sont des variantes de l'algorithme d'Euclide.

Algorithme d'Euclide

Le cœur de l'algorithme d'Euclide [56] consiste, en utilisant la propriété (9.1), à remplacer le calcul du pgcd d'un couple (A', B') par le calcul du pgcd du couple $(B', A' \bmod B')$. Par soucis de clarté, nous adoptons une notation matricielle de cette opération,

$$\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix} \text{ où } q = A' \div B'.$$

La procédure `Euclide` de l'algorithme 9.1 met en œuvre ce procédé.

procedure `Euclide`(A, B)
if $B = 0$ **then return**$(|A|)$ **endif**
$A' := |A|; B' := |B|$
while $B' \neq 0$ **do**
  $q := A' \div B'$
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix}$
  $A' := A''; B' := B''$
endwhile
return(A')
endprocedure

FIG. 9.1 – Algorithme d'Euclide.

Le coût majeur en est le calcul en multiprécision du quotient $A' \div B'$. Or ce dernier est égal à 1 dans 41% des cas [56] ! Pour les algorithmes que nous allons décrire par la suite, nous ne disposons pas d'une telle statistique. Pour pouvoir néanmoins les comparer, nous effectuons systématiquement l'expérience suivante ; nous prenons 100 entiers aléatoires de 100 mots de Ω bits et mesurons à chaque itération la différence de taille entre A', B' et A'', B'' afin de finalement obtenir la décroissance moyenne de A' et B' à chaque itération. Ainsi pour `Euclide`, nous obtenons quelque soit Ω, des entiers A' et B' plus courts de 2 bits en moyenne à chaque itération.

Variantes dues à Lehmer

Ces variantes introduites de 1931 à nos jours utilisent deux idées principales :
– le calcul en simple précision de $A' \div B'$ à l'aide des chiffres significatifs de A' et B'.

– le cumul de plusieurs étapes de l'algorithme d'Euclide en une seule étape réalisée en simple précision. Autrement dit, le calcul du produit de plusieurs des matrices utilisées par `Euclide`

$$\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_1 \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & -q_i \end{bmatrix}$$

pour appliquer en une seule fois la matrice obtenue aux entiers A' et B'.

Elles sont toutes construites suivant le schéma de l'algorithme 9.2. À chaque étape, nous y substituons au calcul du pgcd de deux entiers, A' et B', le calcul du pgcd de A'' et B'', deux combinaisons linéaires de A' et B' dont les coefficients sont fournis par une routine spécifique `pgcd_euclidien_partiel`. Pour une combinaison linéaire quelconque, notons au passage que nous avons seulement,

$$\forall (A,B) \in \mathbb{N}^2,\ \forall (u,v,u',v') \in \mathbb{Z}^4, \text{pgcd}(A,B) \text{ divise } \text{pgcd}(uA+vB, u'A+v'B). \tag{9.3}$$

Ces pgcd partiels doivent donc de plus assurer l'égalité dans (9.3) pour être valides.

procedure `Lehmer`(A,B)
\# On ordonne A et B.
if $|A| > |B|$ **then**
  $A' := |A|$; $B' := |B|$
else
  $A' := |B|$; $B' := |A|$
endif
\# Boucle principale
while $B' \neq 0$ **do**
  $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} :=$ `pgcd_euclidien_partiel`(A', B')
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix}$
  $A' := A''; B' := B''$
endwhile
return(A')
endprocedure

FIG. 9.2 – Schéma des variantes dues à Lehmer.

D'autre part, l'efficacité de tels algorithmes dépend de la "vitesse" de convergence de $\begin{bmatrix} A \\ B \end{bmatrix}$ vers $\begin{bmatrix} \text{pgcd}(A,B) \\ 0 \end{bmatrix}$. Elle dépend donc :

– des temps d'appel à `pgcd_euclidien_partiel`. Pour en diminuer le coût, les variantes proposées n'utilisent que du calcul simple précision pour obtenir une matrice $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$.

– du nombre d'itérations. Pour en diminuer le nombre, nous privilégions des fonctions `pgcd_euclidien_partiel` qui permettent d'obtenir des entiers $A'' = uA' + vB'$ et $B'' = u'A' + v'B'$ "bien" plus petits que A' et B' tout en conservant u, v, u' et v' en simple précision, c'est-à-dire inférieurs à 2^Ω.

L'algorithme original de Lehmer : avant de décrire l'algorithme de Lehmer [62] proprement dit, introduisons tout d'abord les suites dont nous avons besoin dans le théorème 48.

Définition 21. *On appelle séquence associée à deux entiers A et B, $(A > B)$, l'ensemble des 4 suites finies $(A_i, Q_i, U_i, V_i)_0^N$ définies par*

$$\begin{aligned}
&A_0 = A,\ Q_0 = 0,\ U_0 = 1,\ V_0 = 0,\\
&A_1 = B,\ Q_1 = A \div B,\ U_1 = 0,\ V_1 = 1,\\
&\forall i \in \{2,\ldots,N\},\ Q_i = A_{i-1} \div A_i,\\
&(A_i, U_i, V_i) = (A_{i-2}, U_{i-2}, V_{i-2}) - Q_{i-1}(A_{i-1}, U_{i-1}, V_{i-1}),\\
&A_N \neq 0,\ A_{N+1} = 0.
\end{aligned}$$

Remarquons qu'alors,

$$A_i = AU_i + BV_i.$$

Le théorème 48 nous permet de savoir dans quelle mesure les quotients calculés à l'aide des chiffres significatifs a et b de A et B sont égaux à ceux réellement déterminés à partir de A et B.

Théorème 48. *Soient deux entiers $(A, B) \in \mathbb{N}^{*2}$, $A > B$, associés à la séquence $(A_i, Q_i, U_i, V_i)_0^N$ et a, b, les chiffres significatifs de A et B donnés par $a = A \div 2^h$ et $b = B \div 2^h$ avec $h \in \mathbb{N}$ fixé. Soient aussi*
- *$(a_i, q_i, u_i, v_i)_0^n$, la séquence associée à (a, b),*
- *$(a'_i, q'_i, u'_i, v'_i)_0^{n'}$, la séquence associée à $(a, b+1)$,*
- *$(a''_i, q''_i, u''_i, v''_i)_0^{n''}$, la séquence associée à $(a+1, b)$.*

On a alors $Q_j = q_j$ pour tous les indices j tels que $q'_j = q''_j$. Dans ce cas $q'_j = q_j = q''_j$.

En pratique, nous calculons donc les quotients q_i et les entiers a_i jusqu'au premier indice k tels que les quotients q'_k et q''_k soient différents. Nous obtenons alors des coefficients u, v, u' et v' qui sont ceux de

$$\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_{k-1} \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & -q_2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -q_1 \end{bmatrix}.$$

La procédure `Lehmer_partiel` de l'algorithme 9.3 [27] est une formulation de ces remarques. Cette

procedure `Lehmer_partiel`(A, B)
\# a et b chiffres significatifs de $A > B$
\# taille$(a) \leqslant \Omega$ et taille$(b) \leqslant \Omega$
$a := A \div 2^{\max(\text{taille}(A)-\Omega,0)}$; $b := B \div 2^{\max(\text{taille}(A)-\Omega,0)}$
\# Initialisation
$u := 1; v := 0; u' := 0; v' := 1$
\# Boucle principale
while true do
  if $b = -u'$ **or** $b = -v'$ **then break endif**
  $q' := (a + u) \div (b + u'); q'' := (a + v) \div (b + v')$
  if $q' \neq q''$ **then break endif**
  $T := a - q'b; a := b; b := T$
  $T := u - q'u'; u := u'; u' := T$
  $T := v - q'v'; v := v'; v' := T$
endwhile
\# Résultat
if $v = 0$ **then return**$(\begin{bmatrix} 0 & 1 \\ 1 & -A \div B \end{bmatrix})$ **endif**
return$(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix})$
endprocedure

FIG. 9.3 – pgcd partiel de l'algorithme original de Lehmer.

routine a déjà un coût bien moindre que celle de `Euclide` puisque nous observons sur des statistiques obtenues expérimentalement (cf. la routine `Euclide`) qu'elle retourne une matrice $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ qui une fois appliquée à $\begin{bmatrix} A' \\ B' \end{bmatrix}$, permet d'obtenir un résultat plus court de 13 bits en moyenne lorsque $\Omega = 32$ et plus court de 29 bits lorsque $\Omega = 64$.

Les variantes suivantes amélioreront `Lehmer_partiel` en supprimant le calcul de l'un des deux quotients q' et q''.

Apports de Collins et de Jebelean : le théorème 48 donne une condition suffisante pour que les calculs en simple précision de l'algorithme 9.3 soient valides en multiprécision. Il est néanmoins possible d'obtenir une condition nécessaire et suffisante qui nous affranchit du calcul des deux quotients q' et q'' à chaque étape [51].

Théorème 49. *Avec les notations du théorème 48, on a $q_i = Q_i$ si et seulement si*

$$a_{i+1} \geqslant -u_{i+1} \text{ et } a_i - a_{i+1} \geqslant v_{i+1} - v_i \text{ si } i \text{ pair}$$

ou

$$a_{i+1} \geqslant -v_{i+1} \text{ et } a_i - a_{i+1} \geqslant u_{i+1} - u_i \text{ si } i \text{ impair.}$$

En affaiblissant la condition précédente, nous retrouvons alors une condition suffisante proposée par Collins [28].

Corollaire 9. *Avec les notations du théorème 48, $q_i = Q_i$ si*

$$a_{i+1} \geqslant |v_{i+1}| \text{ et } a_i - a_{i+1} \geqslant |v_{i+1} - v_i|.$$

Démonstration. En remarquant que $\forall\ i \geqslant 1,\ |V_i| \geqslant Q_1|U_i|$, le résultat découle immédiatement du théorème 49. □

Cette condition permet de supprimer le calcul d'un quotient dans la fonction `Lehmer_partiel`, nous obtenons ainsi la fonction `Collins_partiel` de l'algorithme 9.4.

procedure `Collins_partiel`(A, B)
\# a et b chiffres significatifs de $A > B$
$a := A \div 2^{\max(\text{taille}(A)-\Omega,0)}$; $b := B \div 2^{\max(\text{taille}(A)-\Omega,0)}$
\# Initialisation
$u := 1; v := 0; u' := 0; v' := 1$
\# Boucle principale
while true do
  if $b = 0$ **then break endif**
  $q := a \div b$; $T := a - qb$
  $u'' := u - qu'$; $v'' := v - qv'$
  if $T < |v''|$ **or** $b - T < |v' - v''|$ **then break endif**
  $a := b$; $b := T$
  $u := u'$; $u' := u''$
  $v := v'$; $v' := v''$
endwhile
\# Résultat
if $v = 0$ **then return**$\left(\begin{bmatrix} 0 & 1 \\ 1 & -A \div B \end{bmatrix}\right)$ **endif**
return$\left(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}\right)$
endprocedure

FIG. 9.4 – pgcd partiel utilisant la condition de Collins.

La matrice $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$, une fois appliquée à $\begin{bmatrix} A' \\ B' \end{bmatrix}$, permet encore en pratique d'obtenir un résultat plus court de 13 bits en moyenne lorsque $\Omega = 32$ et plus court de 29 bits lorsque $\Omega = 64$.

Optimisations de l'algorithme de Lehmer : nous observons lorsque a et b sont de la taille d'un entier simple précision (Ω bits) que u, v, u' et v' ont environ $\Omega \div 2$ bits. Néanmoins, pour obtenir une efficacité meilleure, nous aimerions avoir u, v, u' et v' de Ω bits.

Le choix de a et b en double précision, c'est-à-dire tels que $2^{2\Omega-1} \leqslant a < 2^{2\Omega}$ et $b < 2^{2\Omega}$, permet d'atteindre cet objectif mais, en contre partie, les opérations réalisées maintenant en double précision dans `Collins_partiel` s'avèrent relativement lourdes [51].

Il est cependant possible non seulement de simplifier encore la condition de Collins afin d'obtenir une condition calculable avec un coût nul, mais aussi de remplacer la majeure partie de ces calculs double précision par des calculs en simple précision.

Simplification de la condition de Collins ; pour cela, nous suivons la démarche de Jebelean [51] (notons qu'indépendamment, K. Weber obtient des résultats similaires [117]) et nous introduisons les "polynômes continuants".

Définition 22. *La famille des polynômes continuants est définie par*

$$\begin{aligned} &P_0 = 1,\ P_1(X_1) = X_1,\\ &P_{i+2}(X_1,\ldots,X_i,X_{i+1},X_{i+2}) = P_i(X_1,\ldots,X_i) + X_{i+2}P_{i+1}(X_1,\ldots,X_i,X_{i+1}). \end{aligned}$$

Ces polynômes vérifient la symétrie $P_i(X_1,\ldots,X_i) = P_i(X_i,\ldots,X_1)$. En comparant les définitions respectives de u_i, v_i et P_i, nous obtenons

$$\begin{aligned} |u_i| &= P_{i-2}(q_2,\ldots,q_{i-1}),\\ |v_i| &= P_{i-1}(q_1,\ldots,q_{i-1}). \end{aligned}$$

De même, à partir de la définition de a_i, nous avons

$$\begin{aligned} a &\geqslant a_i P_i(q_i,\ldots,q_1),\\ b &\geqslant a_i P_{i-1}(q_i,\ldots,q_2). \end{aligned}$$

D'où

$$\begin{aligned} |v_{i+1}| &\leqslant a/a_i,\\ |u_{i+1}| &\leqslant b/a_i. \end{aligned}$$

Corollaire 10. *Lorsque a et b sont en double précision ($2^{2\Omega-1} \leqslant a < 2^{2\Omega}$ et $b < 2^{2\Omega}$), on a dans les conditions du théorème 49,*

$$q_i = Q_i \text{ si } a_{i+2} \geqslant 2^{\Omega}.$$

Démonstration. La condition de Collins est vérifiée car

$$|v_{i+1}| \leqslant a/a_i < a/2^{\Omega} < 2^{\Omega} < a_{i+1},$$

et

$$\begin{aligned} &a_i - a_{i+1} \geqslant a_i - q_{i+1}a_{i+1} = a_{i+2} \geqslant 2^{\Omega},\\ &|v_i - v_{i+1}| \leqslant |v_i| + |v_{i+1}| \leqslant |v_i| + q_{i+1}|v_{i+1}| = v_{i+2} \leqslant a/a_{i+1} < a/2^{\Omega} < 2^{\Omega}. \end{aligned}$$

□

Remarque : nous sommes assurés que les calculs de la procédure `Collins_partiel` qui met en œuvre cette condition sont toujours réalisés en simple précision car

$$\begin{cases} |v_0| \leqslant \cdots \leqslant |v_i| \leqslant |v_{i+1}| \leqslant |v_{i+2}| \leqslant |v_{i+3}| \leqslant a/a_{i+2} < 2^{\Omega},\\ |u_0| \leqslant \cdots \leqslant |u_i| \leqslant |u_{i+1}| \leqslant |u_{i+2}| \leqslant |u_{i+3}| \leqslant a/b_{i+2} < 2^{\Omega}. \end{cases}$$

Simplification des calculs en double précision ; notre amélioration consiste à utiliser deux fois la fonction `Collins_partiel`. Elle se décompose en trois étapes :

1. nous utilisons la fonction `Collins_partiel` (dans laquelle nous avons remplacé la condition du corollaire 9 par la condition du corollaire 10) à partir des chiffres significatifs en simple précision de A et B ($A \div 2^{\max(\text{taille}(A)-\Omega,0)}$ et $B \div 2^{\max(\text{taille}(A)-\Omega,0)}$). Nous avons ainsi des coefficients u, v, u', v' plus petits que $2^{\Omega/2}$.

2. nous appliquons la matrice $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ aux chiffres significatifs $\begin{bmatrix} a \\ b \end{bmatrix}$ en double précision de A et B plutôt que directement à $\begin{bmatrix} A \\ B \end{bmatrix}$.
3. nous tronquons alors les entiers en double précision a et b à leurs chiffres significatifs en simple précision auxquels nous continuons d'appliquer la routine `Collins_partiel` modifiée du point 1. Nous obtenons alors finalement des coefficients u, v, u' et v' de l'ordre de 2^Ω.

C'est ce que réalise la fonction `Euclidien_partiel` de l'algorithme 9.5. La validité de cet algorithme provient du théorème 50 qui nous assure que les chiffres significatifs de $uA + vB$ et $u'A + v'B$ sont très proches des chiffres significatifs de $ua + vb$ et $u'a + v'b$.

Théorème 50. *Pour tout $\Omega \in \mathbb{N}$, tout couple $(a, b) \in \{0, \ldots, 2^\Omega - 1\}^2$, tout entier h, tout couple $(A', B') \in \{0, \ldots, 2^h - 1\}^2$ et tout $(u, v) \in \{1 - 2^\Omega, \ldots, 2^\Omega - 1\}^2$, on a*

$$|(u(A' + a2^h) + v(B' + b2^h)) \div 2^h - (ua + vb)| < 2^{\Omega+1}.$$

Ici, la matrice $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$, une fois appliquée à $\begin{bmatrix} A' \\ B' \end{bmatrix}$, permet en pratique d'obtenir un résultat plus court de 22 bits en moyenne lorsque $\Omega = 32$ et plus court de 53 bits lorsque $\Omega = 64$, d'où un gain appréciable par rapport aux routines précédentes.

À titre d'exemple, calculons avec $\Omega = 64$ le pgcd de A et B donnés en notation hexadécimale par

$$\begin{aligned} A &= \text{3929A55FFFFFFFFFF1767CA50000000028CAECBD}, \\ B &= \text{32BF5B92800000002D3A1E037FFFFFFFEB95064B}. \end{aligned}$$

À la première itération de l'algorithme, `Euclidien_partiel(A, B)` retourne

$$\begin{bmatrix} \text{71BDF78E61B} & -\text{801EE20CC19} \\ -\text{844FE9FBFDC} & \text{9509CB85921} \end{bmatrix}$$

et nous obtenons donc

$$\begin{aligned} A &= \text{144CB4BDC24E80694DF6C9D1992C1}, \\ B &= \text{50BC92834532B034D218D0C409B64}. \end{aligned}$$

À la deuxième itération, `Euclidien_partiel(A, B)` retourne

$$\begin{bmatrix} -\text{13D79D59473BB} & \text{4EEADF09775FF} \\ \text{147ABE22F9CF8} & -\text{5173ABEA7DB25} \end{bmatrix}$$

et nous avons finalement

$$\begin{aligned} A &= \text{D56073A3A992F533}, \\ B &= \text{29AC02867D7F27FB}. \end{aligned}$$

Ces entiers étant bornés par 2^{64}, une routine de pgcd en simple précision suffit alors pour finir le calcul, le résultat étant ici 1.

procedure `Euclidien_partiel`(A, B)
\# a et b, chiffres significatifs de A et B en simple précision.
$a := A \div 2^{\max(\text{taille}(A)-\Omega,0)}$; $b := B \div 2^{\max(\text{taille}(A)-\Omega,0)}$
\# Initialisation.
$u := 1$; $v := 0$; $u' := 0$; $v' := 1$; $T := 0$
if $b \neq 0$ **then** $q := a \div b$; $T := a - qb$ **endif**
if $T \geqslant 2^{\Omega \div 2}$ **then**
  while true do
    $q' := b \div T$; $T' := b - q'T$
    if $T' < 2^{\Omega \div 2}$ **then break endif**
    $a := b$; $b := T$
    $T := u - qu'$; $u := u'$; $u' := T$
    $T := v - qv'$; $v := v'$; $v' := T$
    $T := T'$; $q := q'$
  endwhile
endif
if $v = 0$ **then return**$(\begin{bmatrix} 0 & 1 \\ 1 & -A \div B \end{bmatrix})$ **endif**
\# a et b, chiffres significatifs de A et B, en double précision.
$a := A \div 2^{\max(\text{taille}(A)-2\Omega,0)}$; $b := B \div 2^{\max(\text{taille}(A)-2\Omega,0)}$
\# Approximation en double précision des chiffres significatifs de
\# $uA + vB$ et $u'A + v'B$ par $ua + vb$ et $u'a + v'b$
$\begin{bmatrix} a \\ b \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$
\# Restriction de a et b à leurs chiffres significatifs en simple précision.
$a := a \div 2^{\max(\text{taille}(a)-\Omega,0)}$; $b := b \div 2^{\max(\text{taille}(a)-\Omega,0)}$; $T := 0$
if $b \neq 0$ **then** $q := a \div b$; $T := a - qb$ **endif**
if $T \geqslant 2^{\Omega \div 2}$ **then**
  while true do
    $q' := b \div T$; $T' := b - q'T$
    if $T' < 2^{\Omega \div 2}$ **then return**$(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix})$ **endif**
    $a := b$; $b := T$
    $T := u - qu'$; $u := u'$; $u' := T$
    $T := v - qv'$; $v := v'$; $v' := T$
    $T := T'$; $q := q'$
  endwhile
endif
return$(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix})$
endprocedure

FIG. 9.5 – pgcd partiel pour obtenir des coefficients u, v, u' et v' de Ω bits.

9.1.2 pgcd binaires

Contrairement aux algorithmes euclidiens qui se préoccupent des bits de poids fort des entiers manipulés, les algorithmes "binaires" tiennent plutôt compte de la parité de ces derniers et sont plus adaptés à une arithmétique binaire. Ainsi, nous effectuons à chaque étape des combinaisons linéaires pour obtenir des entiers divisibles par une puissance de 2.

De nombreux algorithmes de ce type ont été proposés [106, 97], mais nous n'avons considéré que les plus efficaces en pratique. Les premiers algorithmes présentés sont les algorithmes binaires et plus-minus. Outre leurs optimisations adaptées aux longs entiers, ils ont donné lieu récemment à l'algorithme binaire généralisé que nous décrivons ensuite.

Variantes binaires

Après avoir rappelé la formulation des algorithmes binaires et plus-minus, nous montrons comment ils peuvent s'optimiser pour des grands entiers.

Bien entendu, les restes et quotients des divisions par deux que nous utilisons intensivement dans ces algorithmes ne sont en fait que des opérations de masque ou de décalage effectuées par les processeurs.

Algorithme binaire original : cet algorithme repose sur les quatre relations suivantes.

$$\begin{array}{l} \text{si } A \text{ et } B \text{ sont pairs, } \gcd(A,B) = 2\gcd(A/2, B/2), \\ \text{si } A \text{ est pair et } B \text{ est impair, } \gcd(A,B) = \gcd(A/2, B), \\ \text{si } A \text{ est impair et } B \text{ est pair, } \gcd(A,B) = \gcd(A, B/2), \\ \text{si } A \text{ et } B \text{ sont impairs, } \gcd(A,B) = \gcd(A, |A-B|/2). \end{array} \tag{9.4}$$

L'algorithme original est exactement une implantation des relations (9.4) comme illustré par l'algorithme 9.6.

```
procedure binaire_original(A, B)
A' := |A|;  B' := |B|
l_A := 0;  l_B := 0
if A' = 0 then return(B')  endif
while A' mod 2 = 0 do A' := A' ÷ 2;  l_A := l_A + 1 endwhile
if B' = 0 then return(2^{l_A} A') endif
while B' mod 2 = 0 do B' := B' ÷ 2;  l_B := l_B + 1 endwhile
l := min(l_A, l_B)
if A' < B' then T := A';  A' := B';  B' := T endif
while true do
   T := A' - B'
   if T = 0 then return(2^l A') endif
   while T mod 2 = 0 do T := T ÷ 2 endwhile
   if T > B' then
      A' := T
   else
      A' := B';  B' := T
   endif
endwhile
endprocedure
```

FIG. 9.6 – Algorithme binaire original.

Aucune multiplication ni division ne sont requises dans cet algorithme. Seules les opérations d'addition, de soustraction et de décalage arithmétique sont nécessaires.

Algorithme plus-minus original : la nécessité de comparer A à B nous oblige à gérer simultanément leurs bits de poids fort et de poids faible. Pour échapper à cette contrainte, la variante plus-minus fut développée par Brent et Kung [16]. L'observation

$$\text{“Si } A \text{ et } B \text{ sont impairs, alors 4 divise } \begin{cases} A-B \text{ si } A = B \bmod 4, \\ A+B \text{ sinon”} \end{cases} \tag{9.5}$$

en est la base.

```
procedure plus_minus_original(A, B)
A' := |A|; B' := |B|; l_A := 0; l_B := 0
if A' = 0 then return(B') endif
while A' mod 2 = 0 do A' := A' ÷ 2; l_A := l_A + 1 endwhile
if B' = 0 then return(2^{l_A} A') endif
while B' mod 2 = 0 do B' := B' ÷ 2; l_B := l_B + 1 endwhile
l := min(l_A, l_B)
while true do
  if A' = B' mod 4 then T := |A' − B'| ÷ 4 else T := (A' + B') ÷ 4 endif
  if T = 0 then return(2^l A') endif
  while T mod 2 = 0 do T := T ÷ 2 endwhile
  if T > B' then
    A' := T
  else
    A' := B'; B' := T
  endif
endwhile
endprocedure
```

FIG. 9.7 – Algorithme plus_minus original.

Algorithmes `binaire` et `plus_minus` : les algorithmes 9.6 et 9.7 manipulent des entiers en base 2 ; néanmoins, l'expérience prouve que les machines manipulent bien plus rapidement des entiers sous forme de mots de Ω bits.

L'idée est donc ici de suivre la démarche utilisée dans l'algorithme d'Euclide et de cumuler plusieurs étapes élémentaires en une seule fonction `pgcd_binaire_partiel` (`binaire_partiel` pour l'algorithme `binaire` ou `plus_minus_partiel` pour l'algorithme `plus_minus`), s'inscrivant dans une routine `schéma_binaire` (algorithme 9.8) s'occupant des opérations en multiprécision [56].

Un examen de l'algorithme 9.6 met en évidence deux caractéristiques majeures :

- il nous faut connaître la parité de A et B, ce qui nécessite essentiellement la connaissance des bits de poids faible de ces entiers.
- il nous faut savoir si $A > B$ ou $A \leqslant B$, ce qui nécessite essentiellement la connaissance des bits de poids fort de A et B.

Nous déterminons dans `binaire_partiel` (algorithme 9.9), comme pour l'algorithme d'Euclide, la matrice résultat $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ à l'aide des Ω bits de poids fort et de poids faible de A et B. Tout se passe comme si nous codions dans u, v, u' et v' les additions et les décalages à réaliser et qu'ensuite nous appliquions ces opérations en une seule fois par l'intermédiaire d'une multiplication. La seule difficulté est de contrôler avec précision (ici avec les variables k_A et k_B) la taille de u, v, u' et v'.

Les opérations réalisées dans cette fonction partielle sont toutes en simple précision et assurent d'avoir une matrice de sortie $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ avec des coefficients de Ω bits. Lorsque nous appliquons une telle matrice à (A, B) et que nous divisons le résultat par une puissance de 2, nous observons expérimentalement (cf. la routine `Euclide`) un résultat plus court de 22 bits en moyenne lorsque $\Omega = 32$ et plus court de 43 bits lorsque $\Omega = 64$.

procedure `schéma_binaire`(A, B)
$A' := |A|;\ B' := |B|$
if $A' = 0$ **then return**(B') **endif**
if $B' = 0$ **then return**(A') **endif**
$l := 0$
while $A' \bmod 2 = 0$ **and** $B' \bmod 2 = 0$ **do**
  $l := l + 1;\ A' := A' \div 2;\ B' := B' \div 2$
endwhile
if $A' < B'$ **then** $T := A';\ A' := B';\ B' := T$ **endif**
\# Boucle principale.
while $B' \neq 0$ **do**
  $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} :=$ `pgcd_binaire_partiel`(A', B');
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix}$
  $A' := |A''|; B' := |B''|$
  while $A' \neq 0$ **and** $A' \bmod 2 = 0$ **do** $A' := A' \div 2$ **endwhile**
  while $B' \neq 0$ **and** $B' \bmod 2 = 0$ **do** $B' := B' \div 2$ **endwhile**
  if $A' < B'$ **then** $T := A';\ A' := B';\ B' := T$ **endif**
endwhile
return$(2^l A')$
endprocedure

FIG. 9.8 – Algorithme binaire cumulé.

Quant à la routine `plus_minus_partiel`, elle est plus simple que la routine `binaire_partiel` car il n'est plus nécessaire de tenir compte des bits de poids fort de A et B. À chaque appel de cette fonction, les entiers obtenus A et B sont en pratique plus courts de 26 bits en moyenne lorsque $\Omega = 32$ et plus courts de 52 bits lorsque $\Omega = 64$.

Algorithme `binaire_généralisé`

Une généralisation de l'algorithme `plus_minus` due à Jebelean [52] consiste à trouver une combinaison linéaire de deux entiers A et B divisibles par une puissance de 2 importante. Elle est basée sur le calcul de "conjugués modulaires".

Théorème 51. *Soient a et b deux entiers impairs bornés strictement par 2^{2m} ($m \in \mathbb{N}^*$). Il existe deux entiers x et y bornés strictement par 2^m tels que*

$$xa + yb = 0 \bmod 2^{2m} \text{ ou } xa - yb = 0 \bmod 2^{2m}. \tag{9.6}$$

Les entiers x et y sont appelés les "conjugués modulaires" de a et b.

Démonstration. Si b est impair, alors $b^{-1} \bmod 2^{2m}$ existe et (9.6) est équivalent à

$$xc \pm y = 0 \bmod 2^{2m} \tag{9.7}$$

avec $c = ab^{-1} \bmod 2^{2m}$.

Appliquons l'algorithme d'Euclide étendu à 2^{2m} et c, c'est-à-dire calculons leur séquence associée $(a_i, q_u, u_i, v_i)_0^n$ (définition 21). On obtient alors $a_n = \text{pgcd}(2^{2m}, c) = 1$ puisque c est impair. Comme $a_0 = 2^{2m}$, il existe un indice k tel que

$$a_{k-1} \geqslant 2^m \text{ et } a_k < 2^m.$$

Or,

$$a_k = u_k 2^{2m} + v_k c.$$

On retrouve ainsi exactement (9.7) en posant $x = v_k$ et $y = a_k$. □

procedure `binaire_partiel`(A, B)
\# a' et b', les bits de poids faible de A et B
\# a'' et b'', les bits de poids fort de A et B
$a' := A \bmod 2^{\Omega}$; $b' := B \bmod 2^{\Omega}$
$a'' := A \div 2^{\max(\text{taille}(A)-\Omega,0)}$; $b'' := B \div 2^{\max(\text{taille}(A)-\Omega,0)}$
\# Initialisation
$k_A := \Omega$; $k_B := \Omega$; $l_A := 0$; $l_B := 0$
$u := 1$; $v := 0$; $u' := 0$; $v' := 1$
while $a' \bmod 2 = 0$ **and** $k_A > 0$ **do**
  $a' := a' \div 2$; $a'' := a'' \div 2$; $l_B := l_B + 1$; $k_A := k_A - 1$
endwhile
while $b' \bmod 2 = 0$ **and** $k_B > 0$ **do**
  $b' := b' \div 2$; $b'' := b'' \div 2$; $l_A := l_A + 1$; $k_B := k_B - 1$
endwhile
\# Boucle principale
while $a'' > 0$ **and** $b'' > 0$ **do**
  if $a'' > b''$ **or** $(a'' = b''$ **and** $a' > b')$ **then**
    \# Ici, on aurait eu $A > B$
    $u := u2^{l_A} - u'2^{l_B}$; $v := v2^{l_A} - v'2^{l_B}$
    $a'' := a'' - b''$; $a' := a' - b'$; $l_A := 0$
    while $a' \bmod 2 = 0$ **and** $k_A > 0$ **do**
      $l_B := l_B + 1$; $k_A := k_A - 1$; $a' := a' \div 2$; $a'' := a'' \div 2$
    endwhile
  else
    \# Ici, on aurait eu $B > A$
    $u' := u'2^{l_B} - u2^{l_A}$; $v' := v'2^{l_B} - v2^{l_A}$
    $b'' := b'' - a''$; $b' := b' - a'$; $l_B := 0$
    while $b' \bmod 2 = 0$ **and** $k_B > 0$ **do**
      $l_A := l_A + 1$; $k_B := k_B - 1$; $b' := b' \div 2$; $b'' := b'' \div 2$
    endwhile
  endif
endwhile
return$\left(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}\right)$
endprocedure

FIG. 9.9 – pgcd partiel de l'algorithme binaire cumulé.

procedure `plus_minus_partiel`(A, B)
\# a et b, les Ω bits de poids faible de A et B. a impair.
$b := A \div 2^{\max(\text{taille}(A)-2\Omega,0)}$; $c := B \div 2^{\max(\text{taille}(A)-2\Omega,0)}$
\# Initialisation
$u := 1$; $v := 0$; $u' := 0$; $v' := 1$
\# Boucle principale
while $c \neq 0$ **do**
  $l := 0$
  while $c \bmod 2 = 0$ **do** $c := c \div 2$; $l := l + 1$ **endwhile**
  $a := b$; $b := c$
  $u'' := u$; $v'' := v$
  if $a = b \bmod 4$ **then**
    $c := a - b$;
    if $|u''2^l - u'| > 2^\Omega$ **or** $|v''2^l - v'| > 2^\Omega$ **then break endif**
    $u := u'$; $v := v'$; $u' := u''2^l - u$; $v' := v''2^l - v$
  else
    $c := a + b$;
    if $|u''2^l + u'| > 2^\Omega$ **or** $|v''2^l + v'| > 2^\Omega$ **then break endif**
    $u := u'$; $v := v'$; $u' := u''2^l + u$; $v' := v''2^l + v$
  endif
endwhile
if $v = 0$ **then return**$(\begin{bmatrix} 0 & 1 \\ 1 & -A \div B \end{bmatrix})$ **endif**
return$(\begin{bmatrix} u & v \\ u' & v' \end{bmatrix})$
endprocedure

FIG. 9.10 – pgcd partiel de l'algorithme plus-minus cumulé.

La routine `conjugue` de l'algorithme 9.11 calcule de tels conjugués pour $m = \Omega/2$.

Soient maintenant deux grands entiers A et B dont nous considérons les Ω bits de poids faible, a et b. Nous calculons alors leurs conjugués modulaires x et y et nous avons $A' = (xA \pm yB)/2^\Omega$ qui est $\Omega/2$ bits plus court que $\max(A, B)$.

Pour obtenir B', nous appliquons le schéma de "division exacte" décrit par Jebelean [52] qui à partir de deux entiers A et A' ($A > A'$, A' impair) permet d'obtenir un entier B' du même nombre de bits que A'. À cet effet, nous calculons d, la différence en bits des tailles de A et A', puis $c = A/A' \bmod 2^d$, afin d'obtenir $B' = (A - cA')/2^d$ (algorithme 9.12).

Dans les algorithmes 9.11 et 9.12, nous avons besoin de calculer la quantité $1/x \bmod 2^{2m}$, ce qui est fort coûteux quand nous utilisons l'algorithme d'Euclide étendu. Il est néanmoins possible de calculer une telle quantité beaucoup plus rapidement en remarquant que si $x = x_1 2^m + x_0$, alors

$$1/x \bmod 2^{2m} = (x_0^{-1} \bmod 2^m)(2 - x(x_0^{-1} \bmod 2^m)) \bmod 2^{2m}. \tag{9.8}$$

Ainsi grâce à une table d'inverses calculée modulo 2^{16}, nous avons avec seulement quelques multiplications l'inverse d'un entier modulo 2^{32} ou 2^{64}.

La routine `binaire_généralisé` (algorithme 9.13) utilise les routines `division_exacte` et `conjugue`. Contrairement aux autres pgcd, nous n'avons plus à chaque étape $\text{pgcd}(A, B) = \text{pgcd}(A', B')$ mais seulement

$$\text{pgcd}(A, B) = \eta \text{pgcd}(A', B'),$$

où η est un facteur parasite. En pratique il s'avère que ce "bruit" η est très petit et que l'obtention du pgcd de A et B à l'aide du résultat G de cet algorithme par l'égalité

$$\text{pgcd}(A, B) = \text{pgcd}(\text{pgcd}(G, A \bmod G), B \bmod G),$$

procedure `conjugue`(a, b)
\# a, b impair, 2 entiers de Ω bits
\# Initialisation
if $a = b$ **then return**$([1, -1])$ **endif**
$c := a \div \mod 2^{\Omega}$
if $c < 2^{\Omega \div 2}$ **then return**$([c, -1])$ **endif**
\# Algorithme d'Euclide étendu pour 2^{Ω} et c
\# Première étape à part car 2^{Ω} a $\Omega + 1$ bits.
$a := -1 \bmod 2^{\Omega}$; $a' := c$
$v := 0$; $v' := 1$
$q := a \div a'$; $a'' := 1 + a \bmod a'$
$v'' := v - qv'$; $v := v'$; $v' := v''$
$a := a'$; $a' := a''$
while $a' \geqslant 2^{\Omega \div 2}$ **do**
  $q := a \div a'$; $a'' := a \bmod a'$
  $v'' := v - qv'$; $v := v'$; $v' := v''$
  $a := a'$; $a' := a''$
endwhile
return$([v', -a'])$
endprocedure

FIG. 9.11 – Conjugués modulaires de deux entiers.

procedure `division_exacte`(A, B)
\# On suppose $A > B$, B impair
$d := \lceil \log_2(A) \rceil - \lceil \log_2(B) \rceil$
$c := A \div B \bmod 2^d$
return$([c, d])$
endprocedure

FIG. 9.12 – Réduction par division exacte.

```
procedure binaire_généralisé(A, B)
A' := |A|; B' := |B|
if A' = 0 then return(B') endif
if B' = 0 then return(A') endif
l := 0
while A' mod 2 = 0 and B' mod 2 = 0 do
  l := l + 1; A' := A' ÷ 2; B' := B' ÷ 2
endwhile
if A' mod 2 = 0 then T := A'; A' := B'; B' := T endif
while true do
  if A' < 2^Ω then break endif
  [x, y] := conjugue(B' mod 2^Ω, A' mod 2^Ω)
  B' := |xA' + yB'| ÷ 2^Ω
  if B' = 0 then A' := A' ÷ x; break endif
  while B' mod 2 = 0 do B' := B' ÷ 2 endwhile
  [c, d] := division_exacte(A', B')
  A' := |A' − cB'| ÷ 2^d
  while A' ≠ 0 and A' mod 2 = 0 do A' := A' ÷ 2 endwhile
endwhile
return(2^l Euclide(Euclide(A', A mod A'), B mod A')
endprocedure
```

FIG. 9.13 – Algorithme binaire généralisé.

en utilisant par exemple l'algorithme d'Euclide, prend moins de 5% du temps total en moyenne.

La routine `conjugue` de l'algorithme 9.11 retourne des entiers x et y qui ont au plus $\Omega/2$ bits, ce qui fait qu'à chaque étape les entiers A' et B' sont au plus $\Omega/2$ bits plus petits que A et B.

La version finale de `binaire généralisé` est identique à celle donnée précédemment à l'exception près que nous y réalisons le calcul des conjugués modulaires en double précision afin d'obtenir un couple (x, y) de Ω bits et ainsi avoir des entiers A' et B' plus petits que A et B de Ω bits. Dans ce cas, à chaque étape, les entiers A' et B' obtenus en pratique sont plus courts de 33 bits en moyenne lorsque $\Omega = 32$ et plus courts de 65 bits lorsque $\Omega = 64$.

Recalculons ainsi avec $\Omega = 64$ le pgcd de A et B donnés en notation hexadécimale par

$$\begin{aligned} A &= \text{3929A55FFFFFFFFFF1767CA50000000028CAECBD}, \\ B &= \text{32BF5B92800000002D3A1E037FFFFFFFEB95064B}. \end{aligned}$$

À la première itération de l'algorithme, nous avons

$$[x, y] = [\text{56F005513AC6D838}, \text{10D613369B374B78}]$$

et nous obtenons

$$B = (xA + yB)/2^{128} = \text{14FE44A76E4C2DA4C86E764D}.$$

Par division exacte, nous trouvons alors

$$A = \text{2D6AD3D8C92E80A8F1F05DFF}.$$

À la deuxième itération, nous avons

$$[x, y] = [\text{B9A14D9E3210A0E}, -\text{51B3909A8911B1CA}]$$

et nous obtenons

$$B = (xA - yB)/2^{128} = \text{D8B18C3}.$$

Par division exacte, nous avons finalement

$$A = 12567173.$$

Ces entiers étant bornés par 2^{64}, une routine de pgcd en simple précision suffit alors pour finir le calcul. Le résultat étant ici 1, nous n'avons pas besoin d'éliminer un éventuel entier parasite η.

9.1.3 Bilan

Nous avons résumé dans le tableau 9.14 les caractéristiques principales de chacune des variantes des algorithmes précédents, c'est-à-dire :

poids faible : nombre de bits de poids faible de A' et B' nécessaires à chaque itération pour obtenir $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$.

poids fort : nombre de bits de poids forts de A' et B' nécessaires à chaque itération pour obtenir $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$.

gain observé : différence de taille de A' et B' entre 2 itérations (observée expérimentalement).

Algorithme	Variante	Poids faible	Poids fort	Gain observé	
				$\Omega = 32$	$\Omega = 64$
Euclide		-	2	2	2
Lehmer	Knuth	-	Ω	13	29
	Collins	-	Ω	13	29
	Jebelean	-	Ω	13	29
	Auteur	-	2Ω	22	53
Binaire	Original	1	1	1	1
	Cumulé	Ω	Ω	22	43
Plus-minus	Original	2	-	2	2
	Cumulé	Ω	-	26	52
Binaire	Généralisé	2Ω	-	33	65

FIG. 9.14 – Caractéristiques des algorithmes présentés.

9.2 Algorithmes de pgcd étendus

Les algorithmes de pgcd étendus permettent de calculer à partir d'un couple $(A, B) \in \mathbb{N}^2$, un couple $(U, V) \in \mathbb{Z}^2$, $|U| < B$, $|V| < A$, vérifiant la relation de Bézout,

$$AU + BV = \text{pgcd}(A, B). \tag{9.9}$$

Nous décrivons ici comment étendre les algorithmes de pgcd simple de la section 9.1 pour pouvoir obtenir à partir d'un couple (A, B), un triplet $(U, V, \text{pgcd}(A, B))$ vérifiant la relation (9.9).

Schématiquement, les algorithmes de pgcd simples réalisaient des combinaisons linéaires à partir d'un couple d'entiers (A, B) pour aboutir au couple $(\text{pgcd}(A, B), 0)$. Les algorithmes de pgcd étendus réalisent de même une succession de combinaisons linéaires, mais il ne s'agit plus d'entiers mais d'équations. Ainsi, à partir du système trivial

$$\begin{cases} A &= A, \\ B &= B, \end{cases} \tag{9.10}$$

nous aboutissons au système

$$\begin{cases} AU + BV &= \text{pgcd}(A, B), \\ AB - BA &= 0. \end{cases} \tag{9.11}$$

Pour ce faire, nous cherchons exactement les mêmes combinaisons linéaires que celles de la section 9.1, à savoir celles qui font converger le plus rapidement possible (A, B) vers $(\text{pgcd}(A, B), 0)$. Par exemple, si nous avons un système intermédiaire

$$\left\{ \begin{array}{lcl} AU_A + BV_A & = & A', \\ AU_B + BV_B & = & B', \end{array} \right. \tag{9.12}$$

et que, dans ce cas, les algorithmes de pgcd simples auraient appliqué la matrice de transformation $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ à $\begin{bmatrix} A' \\ B' \end{bmatrix}$, nous appliquerons ici cette transformation au système (9.12) pour obtenir le système

$$\left\{ \begin{array}{lcl} A(uU_A + vU_B) + B(uV_A + vV_B) & = & uA' + vB', \\ A(u'U_A + v'U_B) + BV_B(u'V_A + v'V_B) & = & u'A' + v'B'. \end{array} \right. \tag{9.13}$$

Remarquons qu'il n'est pas nécessaire de calculer dans ces algorithmes les combinaisons linéaires faisant intervenir à la fois U et V. Seule la connaissance de V suffit, car nous obtenons, à la fin de l'exécution, U à partir de V et $\text{pgcd}(A, B)$ par l'équation

$$U = (\text{pgcd}(A, B) - BV)/A. \tag{9.14}$$

Comme les matrices $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix}$ jouent un rôle identique lors du calcul des pgcd simples ou étendus, nous utilisons ici les mêmes routines de "pgcd partiels" que celles utilisées dans la section 9.1 .

9.2.1 Algorithme de Lehmer

Cet algorithme est une application des idées précédentes.

procedure `Euclide_étendu`(A, B)
# Ordonne A et B.
if $|A| > |B|$ **then**
  $A' := |A|;\ B' := |B|;\ inv :=$ **false**
else
  $A' := |B|;\ B' := |A|;\ inv :=$ **true**
endif
$V_A := 0;\ V_B := 1$
# Boucle principale.
while $B' \neq 0$ **do**
  $q := A \div B$
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix}$
  $\begin{bmatrix} V'_A \\ V'_B \end{bmatrix} := \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix}$
  $A' := A''; B' := B''; V_A := V'_A; V_B := V'_B$
endwhile
# Résultat.
if $inv =$ **false then**
  return$([\text{sgn}(A)(A' - V_A|B|) \div |A|, \text{sgn}(B)V_A, A'])$
else
  return$([\text{sgn}(A)V_A, \text{sgn}(B)(A' - V_A|A|) \div |B|, A'])$
endif
endprocedure

FIG. 9.15 – pgcd étendu d'Euclide.

La routine `Lehmer_étendu` de l'algorithme 9.16 est une illustration de (9.13), la routine `pgcd_euclidien_partiel` étant au choix `Lehmer_Partiel`, `Collins_partiel` ou encore `Euclidien_partiel`.

La trame y est exactement celle de l'algorithme 9.2, à ceci près, qu'à l'aide de combinaisons linéaires, nous y maintenons non seulement un couple de pseudo-pgcd (A', B'), mais aussi un couple de pseudo-coefficients de Bézout (V_A, V_B).

procedure `Lehmer_étendu`(A, B)
# Ordonne A et B.
if $|A| > |B|$ **then**
  $A' := |A|;\ B' := |B|;\ inv :=$ **false**
else
  $A' := |B|;\ B' := |A|;\ inv :=$ **true**
endif
$V_A := 0;\ V_B := 1$
# Boucle principale.
while $B' \neq 0$ **do**
  $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} :=$ `pgcd_euclidien_partiel`(A', B')
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} A' \\ B' \end{bmatrix}$
  $\begin{bmatrix} V'_A \\ V'_B \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix}$
  $A' := A''; B' := B''; V_A := V'_A; V_B := V'_B$
endwhile
# Résultat.
if $inv =$ **false then**
  return$([\mathrm{sgn}(A)(A' - V_A|B|) \div |A|, \mathrm{sgn}(B)V_A, A'])$
else
  return$([\mathrm{sgn}(A)V_A, \mathrm{sgn}(B)(A' - V_A|A|) \div |B|, A'])$
endif
endprocedure

FIG. 9.16 – pgcd étendu de Lehmer.

9.2.2 Algorithmes binaire et plus-minus

Ici aussi, la routine `schéma_binaire_étendu` adapte la routine `schéma_binaire` de l'algorithme 9.6 au calcul du coefficient V de Bézout. Néanmoins, la situation n'est pas aussi simple que dans `Lehmer_étendu` car non seulement nous effectuons des combinaisons linéaires, mais nous divisons de plus le résultat par une puissance de 2. Ainsi, à chaque étape, à partir d'un système

$$\begin{cases} AU_A + BV_A &= A', \\ AU_B + BV_B &= B', \end{cases}$$

nous obtenons un nouveau système du type

$$\begin{cases} A\widehat{U_A} + B\widehat{V_A} &= 2^{l_A}\widehat{A'}, \\ A\widehat{U_B} + B\widehat{V_B} &= 2^{l_B}\widehat{B'}. \end{cases} \tag{9.15}$$

Pour que la méthode soit effective, il faudrait pouvoir diviser les équations du système précédent respectivement par 2^{l_A} et 2^{l_B}. Or, $\widehat{V_A}$ et $\widehat{V_B}$ n'ont aucune raison d'être divisibles par ces valeurs. Il est cependant possible de trouver deux entiers c_A et c_B tels que $\widehat{V_A} + c_A A$ soit divisible par 2^{l_A} et $\widehat{V_B} + c_B A$ par 2^{l_B}. Lorsque A est impair, il suffit de choisir $c_A = \widehat{V_A}/A \mod 2^{l_A}$ et $c_B = \widehat{V_B}/A \mod 2^{l_B}$. Dans ce cas, le système (9.15) se réécrit

$$\begin{cases} A\frac{(\widehat{U_A} - c_A B)}{2^{l_A}} + B\frac{(\widehat{V_A} + c_A A)}{2^{l_A}} &= \widehat{A'}, \\ A\frac{(\widehat{U_B} - c_B B)}{2^{l_B}} + B\frac{(\widehat{V_B} + c_B A)}{2^{l_B}} &= \widehat{B'}. \end{cases}$$

La routine `normalise` de l'algorithme 9.17 met en œuvre un tel procédé pour calculer $\frac{(\widehat{V}+cA)}{2^l}$ à partir de $\widehat{V}$, A et l en postulant que l'on sait calculer "rapidement" l'inverse d'un entier modulo 2^Ω (cf équation (9.8)).

procedure `normalise`(V, A, l)
$d := l \div \Omega$; $b := l \mod \Omega$
$U := V$; $i := 1$
while $i \leqslant d$ **do**
  $c := U \div A \mod 2^\Omega$
  $U := (U - cA) \div 2^\Omega$
  $i := i + 1$
endwhile
$c := U \div A \mod 2^b$
$U := (U - cA) \div 2^b$
return(U)
endprocedure

FIG. 9.17 – Normalisation d'un coefficient de Bézout.

Une première implantation consiste à "normaliser" le couple (V_A, V_B) après chaque combinaison linéaire pour obtenir un couple cohérent avec $\widehat{A'}$ et $\widehat{B'}$. Mais avec un tel scénario, V_A et V_B sont de la même taille que A dès la première itération, si bien que les combinaisons linéaires suivantes vont avoir le même coût que la multiplication de A par un entier simple précision.

Il est préférable de ne réaliser cette normalisation qu'à la fin de l'algorithme en divisant A' et B' par la même puissance de 2 à chaque étape et en additionnant cette puissance à un compteur n. V_A et V_B vont alors croître progressivement pendant l'exécution de l'algorithme et en moyenne le coût de la multiplication de V_A et V_B par u, v, u' et v' n'est plus celui de la multiplication de A par un entier simple précision, mais plutôt celui de la multiplication de $\lfloor\sqrt{A}\rfloor$ par un entier simple précision.

C'est dans cet esprit que la routine `schéma_binaire_étendu` de l'algorithme 9.18 a été écrite. La fonction `pgcd_binaire_partiel` que nous utilisons est bien sûr, au choix, la routine `binaire_partiel` (algorithme 9.9) ou `plus_minus_partiel` (algorithme 9.10).

9.2.3 Algorithme binaire généralisé

Ici, la généralisation de la routine `binaire_généralisé` (algorithme 9.13) au cas du pgcd étendu se heurte à deux difficultés.

1. Au cours de cet algorithme, contrairement à `schéma_binaire_étendu` (algorithme 9.18) où à chaque itération nous obtenons un nouveau système (9.13) uniquement à partir du système initial (9.12), nous calculons tout d'abord une unique équation

$$A(xU_A + yU_B) + B(xV_A + yV_B) = xA' + yB', \tag{9.16}$$

   que l'on est ensuite obligé de normaliser immédiatement pour pouvoir appliquer le schéma de la division exacte à un système du type

$$\begin{cases} AU_A + BV_A & = & A', \\ A\frac{(\widehat{U_B} - c_B B)}{2^{l_B}} + B\frac{(\widehat{V_B} + c_B A)}{2^{l_B}} & = & \widehat{B'}. \end{cases} \tag{9.17}$$

   Ces normalisations réalisées à chaque étape pénalisent donc cet algorithme.

2. À l'issue de cet algorithme nous obtenons un système

$$\begin{cases} AU + BV & = & G, \\ AB - BA & = & 0, \end{cases} \tag{9.18}$$

procedure `schéma_binaire_étendu`(A, B)
\# Initialisation.
$A' := |A|;\ B' := |B|$
if $A' = 0$ **then return**(B') **endif**
if $B' = 0$ **then return**(A') **endif**
$l := 0$
while $A' \mod 2 = 0\ \&\ B' \mod 2 = 0$ **do**
  $l := l + 1;\ A' := A' \div 2;\ B' := B' \div 2$
endwhile
$inv :=$ **false**
if $A' \mod 2 = 0$ **then** $T := A';\ A' := B';\ B' := T;\ inv :=$ **true endif**
$V_A := 0;\ V_B := 1;\ \overline{A} := A';\ \overline{B} := B';\ n := 0$
\# Boucle principale.
while $B' \neq 0$ **do**
  $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} :=$ `pgcd_binaire_partiel`(A, B)
  $\begin{bmatrix} A'' \\ B'' \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}$
  $\begin{bmatrix} V'_A \\ V'_B \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix}$
  $A' := A''; B' := B''; V_A := V'_A; V_B := V'_B$
  if $A' < 0$ **then** $A' := -A';\ V_A := -V_A$ **endif**
  if $B' < 0$ **then** $B' := -B';\ V_B := -V_B$ **endif**
  while $A' \mod 2 = 0\ \&\ B' \mod 2 = 0$ **do**
    $n := n + 1;\ A' := A' \div 2;\ B' := B' \div 2$
  endwhile
  if $A' \mod 2 = 0$ **then**
    $T := A';\ A' := B';\ B' := T$
    $T := V_A;\ V_A := V_B;\ V_B := T$
  endif
endwhile
\# Résultat.
$V_A :=$ `normalise`$(V_A, |A|, n)$
if $inv =$ **false then**
  return$([\mathrm{sgn}(A)(A' - V_A\overline{B}) \div \overline{A}, \mathrm{sgn}(B)V_A, 2^l A'])$
else
  return$([\mathrm{sgn}(A)V_A, \mathrm{sgn}(B)(A' - V_A\overline{B}) \div \overline{A}, 2^l A'])$
endif
endprocedure

FIG. 9.18 – pgcd binaire étendu.

où G est, seulement un multiple de pgcd(A, B) comme dans l'algorithme `binaire_généralisé`. Il est donc nécessaire, pour obtenir la relation de Bézout recherchée, d'employer à la fin de `binaire_généralisé_étendu` un algorithme exact, par exemple la routine `Lehmer_étendu`, pour reprendre les calculs à partir de la réduction du système initial (9.10) par l'équation $AU + BV = G$, à savoir

$$\begin{cases} A(1-(A \div G)U) - B(A \div G)V & = & A & \mod G, \\ -A(B \div G)U + B(1-(B \div G))V & = & B & \mod G. \end{cases} \tag{9.19}$$

Néanmoins, en pratique, G est suffisamment petit pour que le coût de cette correction soit négligeable.

L'algorithme 9.19 tient compte de ces deux remarques.

9.3 Résultats

Toutes les variantes décrites aussi bien pour les pgcd simples qu'étendus ont été implantées en C. Par souci de clarté, nous présentons seulement les versions les plus performantes pour cinq types d'algorithmes ; Euclide, Lehmer, binaire, plus_minus et binaire_généralisé.

La librairie multiprécision `BigNum` [48] a été utilisée à cet effet. Elle fournit des routines de calcul en multiprécision qui sont portables, car écrites en C, et efficaces, car s'appuyant sur un noyau de 20 procédures écrites en assembleur pour la majeure partie des architectures.

La longueur Ω (en bits) des entiers en simple précision est un facteur déterminant pour ces algorithmes. C'est pourquoi nous les avons expérimentées sur un ordinateur DEC où $\Omega = 64$, et sur un ordinateur SUN où $\Omega = 32$ (cf. l'introduction). De manière générale, l'influence du microprocesseur est prépondérante, d'une part lors des opérations de multiprécision où nous utilisons intensivement des multiplications d'entiers et d'autre part dans les routines de pgcd partiels ayant besoin de nombreux quotients. En pratique, tous les coups sont bien sûrs permis : codage en assembleur, utilisation des coprocesseurs flottants...

Les temps donnés dans la suite sont des temps moyens obtenus à partir de 1000 essais sur des entiers aléatoires de tailles 10Ω, 20Ω,..., 100Ω.

9.3.1 pgcd simples

Les pgcd simples considérés sont les suivants.

- `Euclide` : la routine `Euclide` de l'algorithme 9.1 qui va nous servir de référence.
- `Lehmer` : la routine `Lehmer` de l'algorithme 9.2 associée à la fonction partielle `Euclidien_partiel` de l'algorithme 9.5.
- `binaire` : la routine `schéma_binaire` de l'algorithme 9.8 associée à la fonction partielle `binaire_partiel` de l'algorithme 9.9.
- `plus_minus` : la routine `schéma_binaire` de l'algorithme 9.6 associée à la fonction partielle `plus_minus_partiel` de l'algorithme 9.9.
- `binaire_généralisé` : la routine `binaire_généralisé` de l'algorithme 9.13 avec un appel à la routine `conjugue` de l'algorithme 9.11 réalisée en double précision.

Il apparaît au travers de ces courbes que le meilleur algorithme dépend fortement de l'architecture utilisée. La raison essentielle est l'absence dans l'assembleur des processeurs alpha des machines DEC d'une instruction de division (présente par contre sur les processeurs sparc des SUN), ce qui pénalise `Euclide`. Les algorithmes `binaire` et `plus_minus` semblent néanmoins être une bonne alternative. Ainsi, par exemple, calculer un pgcd de deux entiers de 100 mots de 64 bits est sur une station DEC environ 10 fois plus rapide par celui-ci que par l'algorithme `Euclide`, mais seulement 5 fois plus rapide pour 100 mots de 32 bits sur une station SUN.

L'efficacité de ces méthodes est modulée par la taille des entiers. Ainsi, les algorithmes dont les fonctions partielles sont rapides à calculer (comme, par exemple, `binaire_partiel`) sont avantagés pour les petites longueurs car il s'agit alors du coût majeur de ces derniers.

Par contre, les algorithmes avec des fonctions partielles permettant de diminuer le nombre d'itérations (donc faisant décroître rapidement les entiers A et B) semblent asymptotiquement les meilleurs car

procedure `binaire_généralisé_étendu`(A, B)
\# Initialisation.
$A' := |A|;\ B' := |B|$
if $A' = 0$ **then return**(B') **endif**
if $B' = 0$ **then return**(A') **endif**
$l := 0$
while $A' \mod 2 = 0\ \&\ B' \mod 2 = 0$ **do**
  $l := l + 1;\ A' := A' \div 2;\ B' := B' \div 2$
endwhile
$inv :=$ **false**
if $A' \mod 2 = 0$ **then** $T := A';\ A' := B';\ B' := T;\ inv :=$ **true endif**
$V_A := 0;\ V_B := 1;\ \overline{A} := A';\ \overline{B} := B'$
\# Boucle principale.
while $B' \neq 0$ **do**
  if $A' < 2^\Omega$ **then break endif**
  $\begin{bmatrix} x \\ y \end{bmatrix} :=$ `conjugue`$(A' \mod 2^\Omega, B' \mod 2^\Omega)$
  $B' := (xB' + yA') \div 2^\Omega;\ n := \Omega$
  if $B' = 0$ **then break endif**
  while $B' \mod 2 = 0$ **do** $n := n + 1;\ B' := B' \div 2$ **endwhile**
  $V_B :=$ `normalise`$(xV_B + yV_A, \overline{A}, n)$
  if $B' < 0$ **then** $B' := -B';\ V_B := -V_B$**endif**
  $[c, d] :=$ `division_exacte`(A', B')
  $A' := (A' - c * B') \div 2^d;\ n := d$
  while $A' \mod 2 = 0$ **do** $n := n + 1;\ A' := A' \div 2$ **endwhile**
  $V_A :=$ `normalise`$(V_A - cV_B, \overline{A}, n)$
  if $A' < 0$ **then** $A' := -A';\ V_A := -V_A$**endif**
  if $A' \mod 2 = 0$ **then**
    $T := A';\ A' := B';\ B' := T$
    $T := V_A;\ V_A := V_B;\ V_B := T$
  endif
endwhile
\# Correction à l'aide de Lehmer_étendu.
$q := \overline{B} \div A';\ B' := \overline{B} - qA';\ V_B := 1 - qV_A$
$q := \overline{A} \div A';\ A' := \overline{A} - qA';\ V_A := -qV_A$
while $B' \neq 0$ **do**
  $\begin{bmatrix} u & v \\ u' & v' \end{bmatrix} :=$ `pgcd_binaire_partiel`(A, B)
  $\begin{bmatrix} A' \\ B' \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}$
  $\begin{bmatrix} V_A \\ V_B \end{bmatrix} := \begin{bmatrix} u & v \\ u' & v' \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix}$
  $A' := A''; B' := B''; V_A := V'_A; V_B := V'_B$
endwhile
if $inv =$ **false then**
  return$([\mathrm{sgn}(A)(A' - V_A\overline{B}) \div \overline{A}, \mathrm{sgn}(B)V_A, 2^l A'])$
else
  return$([\mathrm{sgn}(A)V_A, \mathrm{sgn}(B)(A' - V_A\overline{B}) \div \overline{A}, 2^l A'])$
endif
endprocedure

FIG. 9.19 – pgcd binaire généralisé étendu.

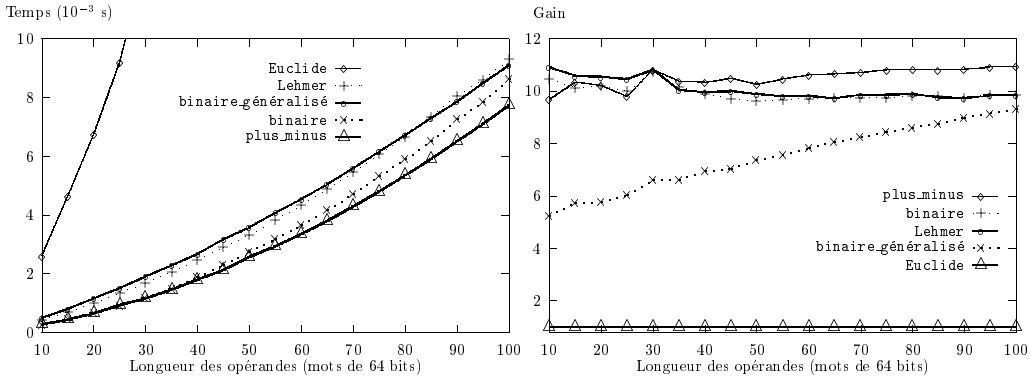


FIG. 9.20 – Temps et gain par rapport à `Euclide` (station DEC).

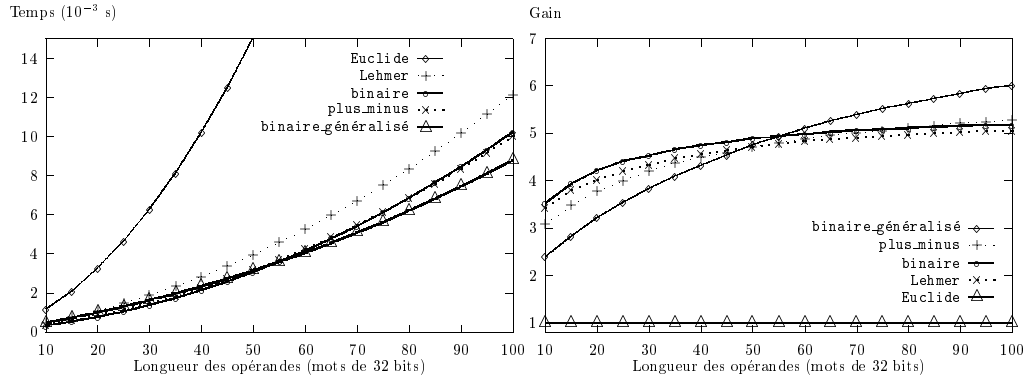


FIG. 9.21 – Temps et gain par rapport à `Euclide` (station SUN).

le coût majeur est alors celui des multiplications en multiprécision. Ainsi, par exemple l'algorithme `binaire_généralisé` qui fait décroître deux entiers A et B de 33 ou 65 bits suivant les architectures mais qui demande du calcul double précision, est assez lent pour des petites longueurs, mais peut devenir la meilleure méthode pour des entiers de 100 mots.

9.3.2 pgcd étendus

Les pgcd étendus considérés sont les suivants.

- `Euclide_étendu` : la routine `Euclide_étendu` de l'algorithme 9.15 qui va nous servir de référence.
- `Lehmer_étendu` : la routine `Lehmer_étendu` de l'algorithme 9.16 avec la fonction partielle `Euclidien_partiel` de l'algorithme 9.5.
- `binaire_étendu` : la routine `schéma_binaire_étendu` de l'algorithme 9.18 avec la fonction partielle `binaire_partiel` de l'algorithme 9.9.
- `plus_minus_étendu` : la routine `schéma_binaire_étendu` de l'algorithme 9.18 avec la fonction partielle `plus_minus_partiel` de l'algorithme 9.10.
- `binaire_généralisé_étendu` : la routine `binaire_généralisé_étendu` de l'algorithme 9.13 avec un appel à la routine `conjugue` de l'algorithme 9.11 réalisé ici en simple précision.

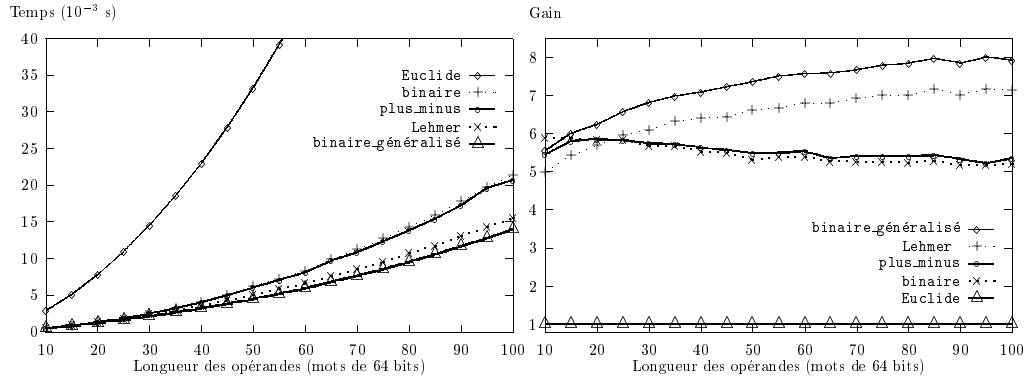


FIG. 9.22 – Temps et gain par rapport à `Euclide` (station DEC).

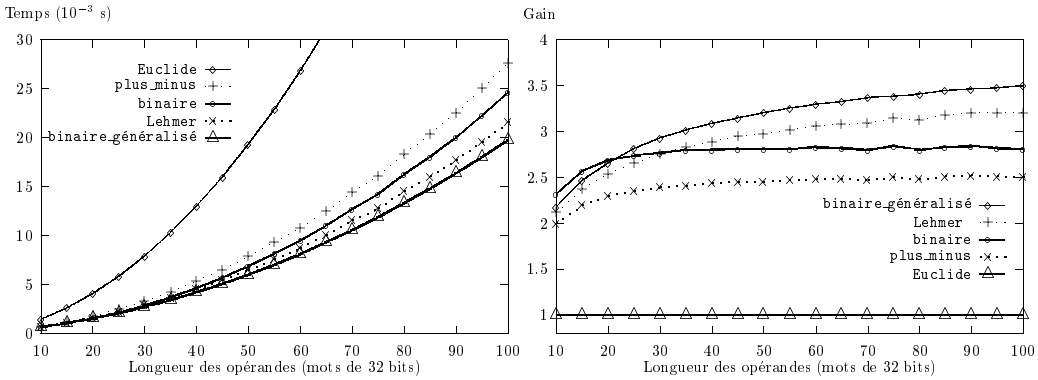


FIG. 9.23 – Temps et gain par rapport à `Euclide` (station SUN).

Ici, les algorithmes les plus rapides sont `binaire_généralisé_étendu` et `Lehmer_étendu` avec par exemple un gain d'environ 7 sur `Euclide_étendu` pour calculer un pgcd étendu de deux entiers de 100 mots de 64 bits sur une station DEC (seulement $3,5$ sur une station SUN avec 100 mots de 32 bits). Contrairement aux pgcd simples, les algorithmes binaires sont pénalisés par l'étape de normalisation

nécessaire lorsque l'on réduit les systèmes d'équations intermédiaires par une puissance de 2. Ceci explique leur recul par rapport à l'algorithme `Lehmer_étendu`.

Le raisonnement de la section précédente concernant le compromis entre rapidité d'exécution et efficacité des fonctions partielles est ici encore valide. Ainsi, vu le surcoût des multiplications en multiprécision à chaque étape (4 multiplications pour un pgcd simple contre 8 pour un pgcd étendu), les algorithmes permettant de faire décroître rapidement les entiers à chaque itération sont intéressants pour de plus petits entiers. Ceci est le cas, par exemple, de l'algorithme `Lehmer_étendu` sur ordinateur DEC.

Chapitre 10

Corps finis et programmation

La programmation des algorithmes décrits dans les parties I et II nécessite l'utilisation de procédures informatiques pour manipuler des éléments, polynômes, séries, matrices, etc, définis sur des corps finis. Pour effectuer un tel travail, deux approches sont envisageables : l'utilisation de logiciels de calcul formel ou celle de librairies spécialisées écrites en langage `C`, `C++`, etc.

Les logiciels de calculs formels comme `Maple` [25], `Axiom` [44] ou `Mathematica` [1] semblent généralement offrir, outre une interface utilisateur conviviale, l'ensemble des opérations dont nous avons besoin. La réalité est malheureusement bien différente. Ainsi, par exemple, si la version V.3 de `Maple` permet de travailler aisément dans un corps fini $\mathbb{F}_{p^n}$ défini comme $\mathbb{Z}/p\mathbb{Z}[X]/(P(X))$ où $P(X)$ est un polynôme irréductible de degré n de $\mathbb{Z}/p\mathbb{Z}[X]$, on ne peut travailler sur une extension finie $\mathbb{F}_{p^{nm}}$ de $\mathbb{F}_{p^n}$, par exemple $(\mathbb{Z}/p\mathbb{Z}[X]/(P(X)))[Y]\ /(Q(Y))$ où $Q(Y)$ est un polynôme irréductible de degré m sur $\mathbb{F}_{p^n}$, qu'au prix d'astuces techniques lourdes et cela devient complètement impossible pour un sur-corps d'un tel $\mathbb{F}_{p^{nm}}$. De manière générale, les performances de ces outils sont pauvres, nous invitons pour s'en convaincre à consulter [22]. Ainsi, en `Maple`, les objets mathématiques sont stockés dans des structures informatiques bien adaptées aux objets creux (arbres, listes chaînées). Cette politique a pour effet de ralentir notablement les accès mémoires (par exemple lors de l'accès aux coefficients d'un polynôme). Malheureusement, les objets que nous manipulons sont toujours denses et ce temps perdu est alors par trop pénalisant. Ces outils sont donc à conseiller pour tester un algorithme, certainement pas pour en étudier le comportement en terme d'efficacité.

À l'opposé, des librairies spécialisées et écrites en langage `C` comme `LiDIA` [73], `BigMod` [90], etc, sont disponibles. Ces librairies sont bien sûr plus difficiles à utiliser, mais en contrepartie, elles fournissent des routines optimisées. Elles permettent ainsi d'utiliser des entiers de taille arbitraire, car elles sont elles-mêmes construites au dessus de librairies `C` comme `BigNum` [48], `GMP` [42] ou `LIP` [63]. Cependant, elles ne permettent de manipuler efficacement qu'un petit nombre de corps finis, principalement $\mathbb{Z}/p\mathbb{Z}$, et ceci est bien insuffisant pour nos besoins.

Entre ces deux mondes, des produits plus aboutis existent, par exemple `Pari` [7], `Magma` [18] ou `Kant` [98], mais si les routines qu'ils fournissent sont inestimables pour des problèmes définis sur $\mathbb{Q}$, les corps finis n'y sont accessibles que de façon détournée et au prix d'une perte d'efficacité.

C'est pourquoi, Chabaud, pour des applications cryptographiques, et l'auteur, pour pouvoir tester en toute généralité les algorithmes des parties I et II, ont été amenés à écrire un nouvel outil, la librairie `ZEN` [23]. Son but premier est la manipulation efficace de n'importe quel corps fini. Après en avoir exposé les aspects principaux (section 10.1), nous décrivons ensuite une application qui en tire parti, l'exponentiation (section 10.2).

10.1 Librairie de programmation ZEN

Le but de la librairie `ZEN` est la manipulation aisée, non seulement de tout corps fini, mais plus généralement de tout anneau fini polynomial, c'est-à-dire $\mathbb{Z}/N\mathbb{Z}$ où N est un entier quelconque ou récursivement toute extension définie par un polynôme $P(X)$ sur un sous-anneau.

10.1.1 Trois principes fondateurs

L'écriture d'un logiciel d'une telle ampleur, environ 70000 lignes à ce jour, ne peut sérieusement avoir lieu sans un certain nombre de principes fondateurs. Ainsi, trois principes assurent la cohérence de `ZEN`, ce sont l'efficacité, l'aspect générique et la portabilité. Nous les décrivons en détail dans les trois sous-sections qui suivent, nous restreignant au cas des corps finis par soucis de clarté.

Efficacité

Dans ce domaine, le choix du langage de programmation et des algorithmes est crucial.

Programmation : le choix du langage et des concepts de programmation utilisés sont des aspects trop souvent négligés et relayés au second plan dans l'écriture d'un logiciel. Ce sont pourtant eux qui en cautionnent la pérennité et l'évolution. Ceci est d'autant plus vrai pour un outil de recherche où il n'est pas rare que de nombreux intervenants aient à en modifier le contenu. Cette erreur est alors ensuite difficilement réparable et conduit généralement à une remise en question de la totalité du logiciel. C'est pourquoi, nous avons été amenés à faire un certain nombre de choix à la naissance `ZEN` que nous allons essayer de justifier ici.

Par expérience, il est apparu très vite que le langage de programmation devait être le langage `C`. En effet, l'utilisation d'un langage "plus évolué" implique l'adjonction intempestive par le compilateur d'instructions annexes. Citons par exemple l'inférence de type en `C++`, l'utilisation d'un "garbage collector" en `LISP`... Tout comme `ZEN`, le compilateur utilisé ne doit produire que ce qui est rendu nécessaire par les spécifications de l'utilisateur, ce qui est le cas pour le langage `C`. D'autre part, ce langage est normalisé[1] et ses spécifications n'évoluent plus depuis maintenant quelques années, ce qui est un gage de pérennité pour les applications. De plus il est maintenant largement utilisé dans la communauté scientifique. Enfin de nombreux langages plus évolués s'interfacent directement avec des modules écrits en `C`.

Regrettons toutefois l'absence dans ce langage de gestion de ce que l'on appelle communément les "exceptions". En effet, dès qu'un logiciel fait appel à une ressource du système, mémoire, entrées/sorties, etc, le système d'exploitation peut être dans l'impossibilité de lui donner satisfaction et retourne dans ce cas un code d'erreur. Ce genre d'événement, normalement rare il est vrai, est généralement ignoré par les librairies écrites en `C` comme, par exemple, `BigMod`, mais pris en compte dans des langages plus évolués grâce au concept d'exception. Les routines de `ZEN` qui effectuent ces "appels système" testent systématiquement leurs codes de retour et retournent, si besoin est, un code d'erreur. Pour l'utilisateur avisé, une fonction similaire à la fonction `perror` des systèmes d'exploitation UNIX est alors disponible pour un complément d'information, à savoir la fonction `ZENError`. Ce mécanisme, certes contraignant à l'écriture, est néanmoins sans influence notable du point de vue de l'efficacité et donc tout à fait satisfaisant.

Par contre, l'un des points forts du langage `C` particulièrement utile à `ZEN` est sa gestion de la mémoire qui est d'un niveau suffisamment bas pour permettre l'implantation des structures les plus complexes. Elle est certes souvent critiquée pour "son abord difficile au débutant" mais avec l'expérience, on en apprécie pleinement la puissance. Plus particulièrement, en ce qui concerne la "mémoire dynamique", c'est-à-dire la mémoire dont la taille est déterminée à l'exécution par les paramètres fournis par l'utilisateur, une routine d'allocation quasi universelle communément appelée `malloc` est fournie. Sous peine d'encombrement de la machine, il est nécessaire contrairement à des langages plus évolués disposant d'un "garbage collector", de "rendre" cette mémoire au système une fois qu'elle est utilisée par une routine de libération appelée `free`. C'est pourquoi les routines de `ZEN` allouent simplement à leur début la mémoire dont elles auront besoin et libèrent systématiquement celle-ci à leur sortie.

[1] La norme du langage `C` est définie par quatre documents :
- la norme ISO/IEC 9899 adoptée par ISO en 1990. L'organisme ANSI (Norme nationale américaine pour les systèmes d'information) remplaça alors sa première norme X3.159-1989 par la norme ANSI/ISO 9899 identique à ISO/IEC 9899 :1990.
- un amendement (ISO/IEC 9899 AM1) à la norme ISO/IEC 9899 approuvé en 1995 et couvrant la norme ISO 646 et les extensions "multi-octets".
- un corrigendum technique (ISO/IEC 9899 TCOR1) approuvé en 1995.
- un corrigendum technique (ISO/IEC 9899 TCOR2) publié en 1996.

Algorithmique : du point de vue algorithmique, `ZEN` tire parti de la collaboration de l'auteur à deux librairies qui n'ont pas été diffusées. D'une part, la librairie `CESAR` initiée par Morain pour $\mathbb{Z}/p\mathbb{Z}$ et d'autre part, la librairie `GFM`, initiée par Chabaud pour $\mathbb{F}_{2^n}$. Les algorithmes qui y sont programmés ayant été longuement optimisés (voir à ce sujet [88, 65, 21]), ils ont bien sûr été une source d'inspiration précieuse pour `ZEN`. Néanmoins, `ZEN` est autre chose que la réunion de `CESAR` et de `GFM`.

Le facteur déterminant pour une implantation optimale est le choix de la structure informatique sous-jacente au stockage des éléments, polynômes, etc, du corps fini considéré. Pour les entiers de taille arbitraire, il est ainsi maintenant communément admis que la structure informatique optimale pour les stocker est un tableau contigu de "mots" (entiers de la taille Ω des registres de la machine, généralement 32 ou 64 bits). C'est ce qui est programmé dans `BigNum` et certaines des routines de `CESAR` qui ont pour arguments ces entiers ont été reprises en y ajoutant la gestion des erreurs décrite au paragraphe précédent. Cela est par exemple le cas du pgcd d'entiers développé initialement par l'auteur pour `CESAR` et décrit dans le chapitre 9, de la racine carré ou du logarithme approché d'un entier développés par F. Morain...

Pour les corps finis, la situation est moins aisée puisque les structures optimales dépendent du corps fini et plus précisément de sa caractéristique et de sa taille. Nous décrivons dans la suite celles de `ZEN`, description certes un peu rébarbative, mais indispensable pour se rendre compte du travail réalisé. Nous donnons pour chaque type de corps, la structure informatique correspondante pour les éléments, les polynômes et les matrices. Pour les séries tronquées et les courbes elliptiques, également implantés dans `ZEN`, le besoin d'une structure adaptée ne s'est pas encore fait sentir et celle-ci est identique pour tous les corps.

"Petits" corps finis : nous en avons distingué quatre.

$\mathbb{F}_2$ **(`Ze2`) :** un élément est un mot prenant comme valeur 0 ou 1. Le fait de stocker un bit dans un mot est rendu obligatoire pour des raisons de compatibilité avec les autres arithmétiques. Néanmoins, cela nous importe peu puisque nous ne connaissons pas d'applications où l'on ait besoin de nombreux éléments sans qu'il soit possible de les organiser sous forme de polynômes ou de matrices. Or la structure de ces deux objets est optimale puisqu'il s'agit d'un tableau contigu de mots où chaque bit correspond à un coefficient. `Ze2` coïncide en partie avec l'héritage de `GFM`.

$\mathbb{Z}/p\mathbb{Z}$**,** $p < 2^{\Omega}$ **(`Zeps`) :** un élément est un mot, un polynôme ou une matrice un tableau contigu de mots dont chaque entrée correspond à un coefficient.

$\mathbb{F}_{q^n}, q^n \leqslant 256$ **(`Zetab`) :** les éléments de ces corps sont indicés et ces indices compris entre 0 et $q^n - 1$ sont utilisés par `ZEN`, un polynôme ou une matrice est alors un tableau contigu de ces indices dont chaque entrée correspond à un coefficient.

$\mathbb{F}_{q^n}, q^n \leqslant 2^{16}$ **(`Zelog`) :** les corps finis étant multiplicativement cycliques, nous représentons chaque élément y par son logarithme x dans la base d'un générateur (g si $y = g^x$). Un polynôme ou une matrice est alors un tableau contigu de ces logarithmes dont chaque entrée correspond à un coefficient. Calculer le logarithme discret d'un élément étant algorithmiquement difficile, l'utilisation d'une telle méthode pour des corps plus gros n'est pas envisageable.

"Grands" corps finis : nous en avons distingué deux.

$\mathbb{Z}/p\mathbb{Z}$**,** $p > 2^{\Omega}$ **(`Zep`) :** un élément est un pointeur sur un entier de taille arbitraire au format `BigNum`, les polynômes et les matrices, des tableaux contigus d'éléments dont chaque entrée correspond un coefficient. `Zep` coïncide avec l'héritage de `CESAR`.

$\mathbb{F}_{q^n}$**,** $q^n > 2^{16}$ **(`Zext`) :** un tel corps fini est en fait le corps défini récursivement comme une extension polynomiale sur un sous-corps. Les éléments du corps sont donc des polynômes du sous-corps, les polynômes et les matrices, des tableaux contigus d'éléments dont chaque entrée correspond à un coefficient.

Autres structures : en dehors des corps précédents, nous citons par soucis d'exhaustivité mais sans entrer dans les détails, trois types d'anneaux implantés dans `ZEN`.

Représentation de Montgomery (`Zem`) : il s'agit d'un analogue de `Zep` dans lequel les éléments sont représentés à la Montgomery [22].

Théorème chinois (`Zec`) : pour un entier $N = pq$ ou p et q sont des nombres premiers, il est parfois judicieux d'effectuer les calculs dans $\mathbb{Z}/p\mathbb{Z}$ et $\mathbb{Z}/q\mathbb{Z}$ pour obtenir ensuite un résultat dans $\mathbb{Z}/N\mathbb{Z}$ par application du théorème chinois. `Zec` est une généralisation de cette idée pour des corps finis ayant le même sous-corps de définition [22].

$\mathbb{Q}$ **(`Zef`) :** une implantation du corps des rationnels $\mathbb{Q}$ pour quelques usages ponctuels.

Notons que cette liste, exhaustive à ce jour, évoluera cependant puisqu'il est relativement aisé d'ajouter à `ZEN` une nouvelle arithmétique.

Application : pour illustrer les effets de ces arithmétiques en pratique, nous avons mesuré le temps nécessaire pour calculer t^{p^n} par l'algorithme binaire de la section 10.2 pour de nombreux corps finis $\mathbb{F}_{p^n}$ vérifiant $2 \leqslant p \leqslant 65521$ et $2 \leqslant p^n \leqslant 2^{512}$. Les quelques lignes du cœur de ce programme sont données par la suite comme exemple d'utilisation de la librairie. Le graphe 10.1 fait état des résultats.

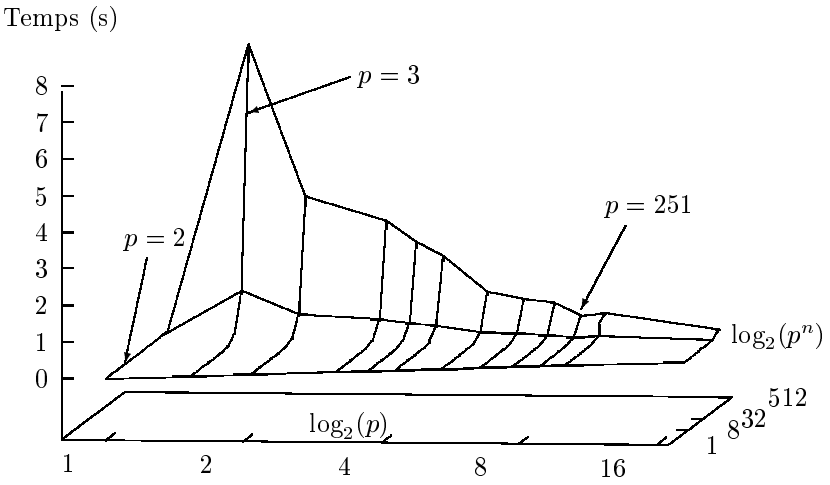


FIG. 10.1 – Calcul de t^{p^n} dans $\mathbb{F}_{p^n} \simeq \mathbb{Z}/p\mathbb{Z}[t]/(P(t))$ sur station DEC.

Tout d'abord l'influence de la caractéristique y apparaît clairement. Ainsi, pour $p = 2$ ($\log_2(p) = 1$), les temps d'exécution sont quasi-nuls grâce à l'arithmétique `Ze2`, ensuite pour des caractéristiques p plus grandes, l'arithmétique `Zetab` entre en jeu et les temps décroissent rapidement lorsque p varie de 3 jusqu'à 251. Notons au passage que les corps de caractéristique 3 avec $3^n > 2^{16}$ sont ceux dont les performances sont les plus mauvaises. Il est donc fort possible à l'avenir que nous complétions la librairie par une arithmétique `Ze3` spécialisée. Enfin, on remarque un léger sursaut pour les corps de caractéristique p immédiatement supérieure à 256 correspondant à l'arithmétique `Zelog`. Un tel phénomène est bien entendu prévisible lors de l'activation respective des arithmétiques `Zeps` et `Zep`, même si nous ne l'avons pas représenté sur le graphe par soucis de clarté.

Les arithmétiques qui se préoccupent de la taille du corps sont `Zetab` et `Zelog`. Ici, tous les corps avec $p^n \leqslant 2^{16}$ bénéficient de ces arithmétiques et les temps d'exécution sont quasi nuls. Par contre, pour $p^n > 2^{16}$, l'arithmétique `Zext` est utilisée, ce qui se traduit par des temps plus importants.

Aspect générique

Il n'est pas rare de constater dans le domaine de la théorie algorithmique des nombres que des algorithmes spécifiés pour tout corps fini soient uniquement testés pour $\mathbb{Z}/p\mathbb{Z}$ car les seules implantations efficaces coïncident avec cette arithmétique. Avec `ZEN`, l'idéal recherché est effectivement de tester cet algorithme dans $\mathbb{Z}/p\mathbb{Z}$, mais ensuite, sans changer une seule ligne du programme, de tester l'algorithme

pour d'autre corps, par exemple, $\mathbb{F}_{2^n}$. C'est ce que nous appelons l'aspect générique. Sa mise en œuvre est d'autant plus malaisée que de nombreuses optimisations sont possibles. Pour ne pas pénaliser l'efficacité, la solution retenue est l'utilisation "manuelle" de mécanismes habituellement réservés aux langages "orientés objet".

Puisqu'un utilisateur doit pouvoir tester différents corps finis sans changer une ligne de son programme, il doit donc fournir au début de l'exécution de ce dernier le corps souhaité. Cela signifie déjà de notre point de vue, que toute routine de `ZEN` doit prendre comme argument une structure, un `ZENRing` dans la terminologie `ZEN`, contenant le corps fini courant.

Un `ZENRing` contenant, entre autres, le type du corps qu'il représente (`Ze2`, `Zep`, `Zext`, ...), il est ensuite aisé d'appeler les routines adaptées. Dans certaines librairies écrites en langage `C`, par exemple `Pari`, le choix des opérations utilisables a lieu au travers d'une série de tests au début de chacune des routines. Outre une écriture pesante, une telle solution s'accompagne automatiquement d'une perte d'efficacité non négligeable.

Dans `ZEN`, la solution retenue est autre. Dans un premier temps, la routine chargée de l'initialisation d'un `ZENRing`, non seulement y stocke les informations que l'on s'attend à y trouver (type, caractéristique, cardinalité, ...) mais aussi des pointeurs (des adresses d'appels) sur les routines valides pour ce corps. La valeur de ces pointeurs dépend donc du corps considéré. Dans un second temps, lorsque le pré-processeur du compilateur `C` analyse le programme d'un utilisateur, il remplace les appels aux routines de `ZEN`, qui ne sont en fait que ce que l'on appelle des macros en terminologie `C`, par des appels aux valeurs des pointeurs du `ZENRing` courant. Ainsi, l'utilisateur ne programme son algorithme qu'une fois, mais suivant le `ZENRing` dynamiquement utilisé, les valeurs de ces pointeurs et donc les routines effectivement appelées sont différentes.

Ce mécanisme, habituellement commun aux langages "orienté objets" n'est pas difficile à mettre en œuvre en langage `C`. Dans notre cas, ce fut d'autant plus facile que le corps fini ambiant conditionne complètement les choix algorithmiques. Il s'est de plus traduit par une facilité d'écriture de la librairie et une perte négligeable d'efficacité.

Portabilité

Les techniques de portabilité décrites brièvement ici proviennent en grande partie des enseignements tirés par l'auteur d'une semaine passée au sein de l'équipe de T. Setz chargée de développer à l'université de Saarbrücken un outil de calcul partagé, la librairie `LiPS` [105].

La compilation d'un logiciel de plus 600 fichiers sous plusieurs systèmes d'exploitation comme, par exemple, les systèmes UNIX (BSD et système V), Windows NT ou OS/2 pour ne citer que les plus connus, se heurte à de multiples écueils : environnement (`USER` contre `LOGNAME`), conventions pour les fichiers (`/` contre \, `.o` contre `.obj`,`.a` contre `.lib`), utilitaires (spécifications différentes pour des commandes usuelles)... Face à ce monde hétéroclite, le langage `C` reste le seul point fixe. C'est pourquoi, les deux outils que nous avons mis en place pour organiser la compilation de la librairie et la génération automatique de la documentation l'ont pour seul pré-requis.

Pour ordonner la compilation de fichiers, il est courant, notamment en UNIX, d'organiser les dépendances entre eux par des fichiers appelés `Makefile` utilisés ensuite par la commande `make`. Si cet outil est très adapté à la compilation sur *un* système, il est toutefois difficile d'emploi pour la mise en œuvre simultanée sur *plusieurs*. D'autre part il est très difficile de faire des changements globaux à l'ensemble des `Makefile`. Pour contourner ces difficultés, la solution est de générer automatiquement ces `Makefile`. À ce stade, l'écriture d'un programme `C` *ad hoc* est tout à fait envisageable. En pratique, nous avons utilisé `imake` [38], un outil développé par les programmeurs de la librairie "X Window System" du Massachusetts Institute of Technology. Il s'agit tout simplement d'un ensemble de fichiers de configuration pour le pré-processeur `cpp` du compilateur `C`, répertoriant les spécificités d'une multitude de systèmes d'exploitation. Reste à l'utilisateur à écrire des fichiers appelés `imakefile` dépendant uniquement de son application. À la compilation, `cpp`, au travers de la commande `imake`, fait alors la fusion entre les fichiers de configuration et les fichiers `imakefile` pour obtenir des fichiers `Makefile` dépendant à la fois de la machine considérée et de l'application. Ainsi, la librairie `ZEN` a pu être compilée et testée sur de nombreuses architectures comme l'atteste le tableau 10.2.

Machine	Processeur	Système d'exploitation	Compilateur
SUN	sparc	SunOS	cc, gcc
		Solaris	SUNWspro/cc, gcc
DEC	vax	Ultrix	cc, gcc
	mips		
	alpha	OSF, Digital UNIX	
HP	PA-RISC	HP-UX	cc, gcc
IBM	RS6000	AIX	cc, gcc
PC	486, pentium	netbsd	gcc
		linux	
		OS/2	

FIG. 10.2 – Portabilité de `ZEN`.

Plus brièvement, pour générer la documentation automatiquement, nous utilisons un filtre ad hoc écrit en langage `C` (voir [22] pour plus de détails), pour gérer aisément les contributions des développeurs et les versions successives de `ZEN`, nous utilisons `cvs` [41], un utilitaire devenu maintenant courant.

10.1.2 Utilisation

Une fois la librairie `ZEN` compilée, seuls deux fichiers sont nécessaires.

`zen.h :` un fichier à inclure dans tout programme utilisant des routines de `ZEN` par

```
#include "zen.h"
```

`libzen.a :` l'ensemble des routines de `ZEN` à compiler avec un programme utilisateur, ici dans notre exemple `test.c`, par

```
cc -lzen test.c
```

Syntaxe

Par souci de simplicité, la majeure partie des routines de `ZEN` se déclinent sous la forme

`<Type><Opération>(<X>,<Arguments>,<Anneau>)`

où

`Type` est l'un des types de la librairie. Elle en compte principalement sept : `ZENRing`, les anneaux (pour nous les corps finis), `ZENElt`, les éléments d'un `ZENRing`, `ZENPoly`, les polynômes à coefficients dans un `ZENRing`, `ZENMat`, les matrices à coefficients dans un `ZENRing`, `ZENSr`, les séries tronquées à coefficients dans un `ZENRing`, `ZENEcPt`, les points d'une courbe elliptique `ZENEc` et enfin `ZBN` pour l'interface avec `BigNum`.

`Opération` est la routine désirée. De nombreuses sont disponibles ; l'allocation avec `Alloc` et `Free`, l'initialisation avec `SetToZero`, `SetToOne`, `Assign`..., les tests avec `IsZero`, `IsOne`, `AreEqual`, les opérateurs arithmétiques avec `Add`, `Multiply`, `Inverse`...ou encore les entrées/sorties avec `ReadFromFile`, `PrintToString`...

`X` est l'objet à modifier, généralement préalablement alloué.

`Arguments` représente les paramètres, leur nombre et leur type dépendent bien sûr de la routine.

`Anneau` est l'anneau courant. Cet argument est toujours nécessaire, sauf pour les routines préfixées par `ZBN`.

Comme exemple d'application, il n'est pas difficile d'écrire la procédure `TestExponentielle` utilisée pour le graphe 10.1 afin de mesurer le temps de calcul de t^q pour un corps fini $\mathbb{F}_q$ donné. Tout d'abord le "header de la fonction" est classiquement défini par ce qui suit.

```
double TestExponentielle(GFq)
       ZENRing GFq;
```

Nous avons alors besoin de trois variables locales ; un chronomètre `tps` et deux éléments `e1`, `e2`.

```
{
  ZENElt e1, e2;
  double tps;
```

Nous allouons ensuite `e1`, `e2` et nous initialisons `e1` à t :

```
  ZENEltAlloc(e1, GFq);
  ZENEltAlloc(e2, GFq);
  ZENEltSetToGenerator(e1, GFq);
```

Une fois fixé le chronomètre au temps courant avec la fonction `runtime` de `ZEN`, nous calculons t^q dans $\mathbb{F}_q$ et nous arrêtons le chronomètre :

```
  tps = runtime();
  ZENEltExp(e2, ZENRingQ(GFq), ZENRingQl(GFq), e1, GFq);
  tps = runtime()-tps;
```

Notons ici que `ZENRingQ(GFq)` et `ZENRingQl(GFq)` sont deux champs de la structure `GFq` qui représentent au format `BigNum` le nombre d'éléments de `GFq`. Nous vérifions ensuite que t^q est bien égal à t (sinon, cela signifie que $\mathbb{F}_q$ n'est pas un corps fini) :

```
  if (ZENEltAreEqual(e1, e2, GFq) == 0) {
    printf("Cet anneau n'est pas un corps fini\n");
  }
```

Nous terminons alors en libérant la mémoire et en retournant le temps mesuré :

```
  ZENEltFree(e1, GFq);
  ZENEltFree(e2, GFq);
  return(tps);
}
```

Il ne reste plus qu'à initialiser des corps finis $\mathbb{F}_q$ pour servir d'argument à `TestExponentielle`. S'il s'agit par exemple de

$$\mathbb{Z}/18446744073709551629\mathbb{Z},$$

la séquence d'instructions suivante convient :

```
  double temps;
  BigNum n; int nl;
  ZENRing GFq1;

  ZBNReadFromString(&n, &nl, "18446744073709551629", 10);
  ZENBaseRingAlloc(GFq1, n, nl);
  temps = TestExponentielle(GFq1);
```

Pour

$$\mathbb{F}_{18446744073709551629^4} \simeq (\mathbb{Z}/18446744073709551629\mathbb{Z})[t]/(t^4+2),$$

il suffit simplement d'ajouter ce qui suit.

```
  {
    ZENPoly P;
    ZENRing GFq2;
```

```
  ZENPolyReadFromString(P, "(1)*X^4+(2)", 10, GFq1);
  ZENExtRingAlloc(GFq2, P, GFq1);
  temps = TestExponentielle(GFq2);
}
```

Bien entendu, il suffit de réitérer le mécanisme précédent pour obtenir `GFq3`, une extension polynomiale de `GFq2`, ou `GFq4`, une extension de `GFq3`...

Optimisations

Une fois un programme écrit et testé avec succès, on cherche généralement à l'optimiser. Dans cette optique, `ZEN` fournit deux mécanismes aisés à utiliser, les précalculs et les clones.

Le but d'un précalcul est d'effectuer pour un `ZENRing` donné, au prix d'un temps de calcul non négligeable, des mémorisations qui serviront ultérieurement à accélérer une routine particulière de ce `ZENRing`. En fait, on peut voir un précalcul comme une composante du corps. Ainsi, il est possible d'ajouter ou d'enlever un ou plusieurs précalculs à un corps, et ce de façon transparente, avec les routines `ZENRingAddPrc` et `ZENRingRmPrc`.

Un clone d'un `ZENRing` `GFp` est quant à lui un `ZENRing` de même cardinalité que `GFp` mais de structure différente. En effet, si des corps finis à cardinalité égale sont mathématiquement isomorphes, de nombreuses représentations informatiques de ces derniers sont au contraire possibles. Par défaut, `ZEN` utilise par défaut la représentation polynomiale, c'est-à-dire considère qu'un anneau est soit $\mathbb{Z}/N\mathbb{Z}$ où $N \in \mathbb{N}$, soit un espace vectoriel sur un sous-anneau avec comme base $1, t, t^2, \ldots t^{n-1}$ et $P(t) = 0$ où $P(X)$ est polynôme de degré n à coefficients dans le sous-anneau. Néanmoins, d'autres représentations pouvant être plus efficaces pour un problème donné existent. Dans la terminologie `ZEN`, un corps fini `GFc` avec une représentation autre que celle polynomiale est appelé un clone de la représentation par défaut `GFp`. Pour spécifier une représentation particulière, l'utilisateur dispose de la routine `ZENRingClone(GFp, cln)` qui retourne le clone spécifié par le drapeau `cln` de l'anneau `GFp`.

Précalculs et clones sont indépendants. Il est ainsi tout à fait possible (et même conseillé) d'ajouter des précalculs à un clone. Nous développons plus en détail ces concepts dans les deux paragraphes suivants.

Précalculs : par défaut, `ZEN` fournit des routines qui n'utilisent aucun précalcul. Notamment, l'allocation d'un anneau `GFq` par `ZENBaseRingAlloc` ou `ZENExtRingAlloc` est quasi immédiate. Cependant, pour des applications ponctuelles effectuant dans `GFq` un grand nombre d'une opération particulière, typiquement `ZENEltMultiply`, il est possible de remplacer la routine utilisée par défaut par une routine plus rapide mais nécessitant le calcul préalable de quantités ne dépendant que de `GFq`.

Par exemple, imaginons qu'il soit nécessaire de calculer dans un programme la trace d'un élément `x` d'un corps fini `GFq` $= \mathbb{F}_{q^n}$ par rapport à son corps de définition $\mathbb{F}_q$. En `ZEN`, il suffit d'écrire pour cela `ZENEltTrace(x, GFq)`. Si `GFq` ne contient pas de précalculs, la routine par défaut est activée, elle effectue simplement ce calcul par

$$\mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(x) = \sum_{i=0}^{n-1} x^{q^i}.$$

Si par contre sont stockées dans `GFq` les quantités

$$\mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(1), \mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(t), \ldots, \mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(t^{n-1}),$$

où $1, t, \ldots, t^{n-1}$ est une base de $\mathbb{F}_{q^n}$ considéré comme un espace vectoriel sur $\mathbb{F}_q$, `ZENEltTrace(x, GFq)` appelle une routine spécifique qui écrit `x` dans la base précédente, `x`$= \sum_{i=0}^{n-1} x_i t^i$ avant d'effectuer

$$\mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(x) = \sum_{i=0}^{n-1} x_i \mathrm{Tr}_{\mathbb{F}_{q^n}/\mathbb{F}_q}(t^i).$$

Pour activer un précalcul donné, l'utilisateur le spécifie dans un drapeau `prc` de type `ZENPrc` à l'aide de la routine `ZENPrcSet`. Il ne lui reste alors plus qu'à faire effectuer les calculs par un appel à `ZENRingAddPrc(GFq, prc)`.

Les précalculs disponibles sont actuellement :

__ `ZENPRC_ELT_MULTIPLY :` accélération de `ZENEltMultiply`. Pour un `ZENRing` de type `Zep`, la méthode de multiplication d'entiers par Karatsuba est activée. Sinon, dans une extension algébrique, le polynôme réciproque du modulo est calculé, ce qui permet de tirer parti de la multiplication polynomiale par l'algorithme de Karatsuba présent dans toutes les arithmétiques de la librairie.

__ `ZENPRC_ELT_EXP :` accélération de `ZENEltExp`. L'allocation de tampons permet d'activer la méthode d'exponentiation m-aire (cf. section 10.2).

__ `ZENPRC_TRACE :` accélération de `ZENEltTrace` comme expliqué précédemment.

__ `ZENPRC_POLY_ROOTS_CANONICAL :` accélération de `ZENPolyRootsCanonical` en trouvant un élément de trace non nulle.

__ `ZENPRC_FINITE_FIELD :` permet aux précalculs précédents de tirer parti du fait qu'un anneau est en fait un corps fini. Par exemple, l'exposant est ainsi réduit modulo le nombre d'éléments du corps avant toute exponentiation quand le précalcul `ZENPRC_ELT_EXP` est utilisé.

Ainsi, pour accélérer le calcul de t^q dans la section précédente, il suffit simplement d'ajouter, après l'initialisation de `GFq1` (ou `GFq2`), les instructions suivantes.

```
{
  ZENPrc prc;

  ZENPrcSetAll(prc);
  ZENRingAddPrc(GFq1, prc);
}
```

Ce qui a pour effet d'emmagasiner dans `GFq1` tous les précalculs prévus par la librairie.

Clones : comme expliqué précédemment, un clone est un `ZENRing` qui utilise une représentation distincte de la représentation polynomiale. Trois types de clones sont actuellement disponibles :

__ `ZENCLN_TAB :` activation de l'arithmétique `Zetab`.

__ `ZENCLN_LOG :` activation de l'arithmétique `Zelog`.

__ `ZENCLN_MONTGOMERY :` activation de l'arithmétique `Zem`.

Le mécanisme que nous avons implanté pour obtenir un clone `GFc` d'un `ZENRing GFp` est similaire à celui des précalculs. L'utilisateur spécifie le type de clone désiré dans un drapeau `cln` de type `ZENCln` à l'aide de la routine `ZENClnSet`. Il ne lui reste alors plus qu'à obtenir le clone proprement dit par "`GFc = ZENRingAddPrc(GFp, prc)`".

Notons que si l'utilisateur introduit plusieurs types de clone dans `cln`, par exemple par `ZENClnSetAll(cln)`, des choix par défaut sont effectués. Ainsi `Zetab` est utilisé en priorité si `GFp` a moins de 256 éléments. Sinon, `Zelog` est utilisé si `GFp` a moins de 2^{16} éléments. Dans le cas contraire, `Zem` est finalement activé si `GFp` est un corps premier. Si aucun clone disponible ne convient comme c'est le cas actuellement si `GFp` est une extension algébrique de plus de 2^{16} éléments, une simple copie de `GFp` est retournée.

Ainsi, pour accélérer le calcul de t^q pour de petit corps finis dans la section précédente, il suffit simplement d'ajouter après l'initialisation de `GFq1` (ou `GFq2`) les instructions suivantes.

```
{
  ZENCln cln;

  ZENClnSetAll(cln);
  GFq1 = ZENRingClone(GFq1, cln);
}
```

10.2 Exponentiation

Le choix d'un "bon" algorithme d'exponentiation est un exemple de problème auquel on est confronté lorsque l'on écrit une librairie comme `ZEN`. En effet, de nombreuses applications et en particulier un grand

nombre des algorithmes énoncés précédemment, nécessitent la programmation efficace de l'exponentiation aussi bien dans le cadre d'un corps fini $\mathbb{F}_q$ pour calculer x^k ($x \in \mathbb{F}_q$, $k \in \mathbb{Z}$) que dans le cadre d'une courbe elliptique E pour calculer kP ($P \in E$, $k \in \mathbb{Z}$). Ce problème est en fait plus généralement le calcul pour un élément g d'un groupe G muni d'une loi $\star$ de la quantité

$$\underbrace{g \star \cdots \star g}_{k \text{ fois}}, \text{ avec } k \in \mathbb{N}^*.$$

Dans le cas particulier des corps finis et des courbes elliptiques, les groupes considérés sont respectivement multiplicatif ($\star = \times$) et additif ($\star = +$).

Deux classes de méthodes ont été proposées pour effectuer ce calcul.

- La première classe est basée sur des chaînes d'addition. Si k est un entier, une chaîne d'addition pour k est la donnée d'un r-uplet $(k_0, k_1, \ldots, k_r)$ tel que $k_0 = 1$, $k_r = k$ et
$$\forall i \in \{1, \ldots, r\},\ \exists a(i), b(i) < i,\ k_i = k_{a(i)} + k_{b(i)}.$$
Les méthodes "binaire", "m-aire" ainsi que les méthodes "euclidiennes" appartiennent à cette classe. La méthode de Bos-Coster [12] en fait aussi partie, mais cette méthode basée sur des heuristiques a des performances similaires à la méthode m-aire et nous ne la considérons donc pas ici.
- La seconde classe est basée sur des chaînes d'addition-soustraction et est efficace quand l'inverse d'un élément est rapide à calculer, ce qui est par exemple le cas pour une courbe elliptique. Si k est un entier, une chaîne d'addition-soustraction pour k est la donnée d'un r-uplet $(k_0, k_1, \ldots k_r)$ tel que $k_0 = 1$, $k_r = k$ et
$$\forall i \in \{1, \ldots, r\},\ \exists a(i), b(i) < i,\ k_i = \pm k_{a(i)} \pm k_{b(i)}.$$
Les méthodes "binaire signée" et "m-window" en font partie.

L'efficacité de ces méthodes dépend bien entendu de la longueur des chaînes qu'elles fournissent. Rappelons à ce propos que trouver la longueur minimale $l(k)$ d'une chaîne d'addition pour un entier k est un problème NP-complet [37] et que $l(k)$ vérifie

$$\log_2(k) + \log_2(\nu(k)) - 2.13 \leqslant l(k) \leqslant \log_2(k) + \log_2(k)/\log_2(\log_2(k)) + o(\log_2(k)/\log_2(\log_2(k))).$$

La borne inférieure est due à Schönhage [109] et la borne supérieure à Brauer [13].

10.2.1 Chaîne d'addition

Méthode binaire

procedure `binaire_exp`(x, k)
$e := k;\ y := 1;\ c := x$
while $e > 0$ **do**
  if $e \bmod 2 = 1$ **then** $y := yc$ **endif**
  $e := e \div 2;\ c := c^2$
endwhile
return(y)
endprocedure

FIG. 10.3 – Méthode binaire "droite-gauche".

La méthode usuelle est la méthode binaire [56]. Elle se déduit aisément des formules de récurrences suivantes

$$x^k = \begin{cases} x^{2(\frac{k-1}{2})}x \text{ si } k \text{ impair}, \\ x^{2(\frac{k}{2})} \text{ si } k \text{ pair}. \end{cases}$$

L'algorithme 10.3 met en œuvre ces relations.

Par exemple, une chaîne d'addition pour 957 (16 opérations) est

$$(1, 2, 3, 6, 7, 14, 28, 29, 58, 59, 118, 119, 238, 239, 478, 956, 957).$$

Pour un entier k, une telle chaîne d'addition a donc une longueur de $\lfloor \log_2(k) \rfloor + \nu(k) - 1$. En moyenne, $\frac{3}{2}\lfloor \log_2(k) \rfloor$ multiplications sont donc nécessaires pour évaluer x^k.

Méthode m-aire

En préstockant des puissances x^t pour des entiers t plus petits qu'une borne 2^m où m est un entier arbitraire, la méthode binaire se généralise en la méthode m-aire [56]. Pour cela, on décompose l'exposant k en base 2^m,

$$k = \sum_{i=0}^{s} k_i 2^{mi}.$$

L'exponentielle s'écrit alors

$$x^k = (\cdots((((x^{k_s})^{2^k} x^{k_{s-1}})^{2^k}) x^{k_{s-2}}))^{2^k} \cdots)^{2^k} x^{k_0}.$$

L'idée est donc de stocker toutes les quantités x^{k_i} pour tous les k_i possibles. Un raffinement consiste à

```
procedure m-aire_exp(x, k, m)
# Précalculs
s := (taille(k) − 1) ÷ m
τ[1] := x; τ[2] := x^2
for i := 1, ..., 2^(m−1) − 1 do τ[2i + 1] := τ[2i − 1]τ[2] endfor
# Poids fort
κ := (k mod 2^(m(s+1))) ÷ 2^(ms); r := 0
while κ mod 2 = 0 do r := r + 1; κ := κ/2 endwhile
y := τ[κ]; for i := 1, ..., r do y := y^2 endfor
# Boucle principale
for j := s − 1, ..., 0 do
  κ := (k mod 2^(m(j+1))) ÷ 2^(mj)
  if κ = 0 then
    for i := 1, ..., m do y := y^2 endfor
  else
    r := 0
    while κ mod 2 = 0 do r := r + 1; κ := κ/2 endwhile
    for i := 1, ..., m − r do y := y^2 endfor
    y := yτ[κ]
    for i := 1, ..., r do y := y^2 endfor
  endif
endfor
return(y)
endprocedure
```

FIG. 10.4 – Méthode m-aire.

ne calculer que les puissances impaires de x, les puissances paires étant englobées dans les carrés. Ce qui conduit à l'algorithme 10.4.

Par exemple, une chaîne d'addition pour 957 (14 opérations) est ainsi

$$(1, 2, 3, 6, 7, 14, 28, 56, 59, 118, 236, 239, 478, 956, 957).$$

Cette méthode nécessite asymptotiquement $(1 + 1/(m + 1)) \log_2(k)$ multiplications au prix d'un stockage égal à 2^{m-1} registres, c'est-à-dire un gain de $1/3 - 2/(3m+3)$ par rapport à la méthode binaire. Ainsi, pour des exposants k de 20000 bits et $m = 9$, un gain de 24% est possible (cf. graphe 10.7).

Méthodes euclidiennes

Des méthodes basées sur des algorithmes similaires à l'algorithme d'Euclide ont été récemment développées [9]. Pour cela, on définit le produit $\sigma \otimes \sigma'$de 2 chaînes d'addition $\sigma = (n_0, n_1, \ldots, n_s)$ et $\sigma' = (m_0, m_1, \ldots, m_t)$ par

$$\sigma \otimes \sigma' = (n_0, n_1, \ldots, n_s, n_s m_1, \ldots, n_s m_t),$$

ainsi que l'addition $\sigma \oplus j$ d'une chaîne σ à un entier j de σ par

$$\sigma \oplus j = (n_0, n_1, \ldots, n_s + j).$$

L'algorithme consiste alors à remplacer récursivement le calcul d'une chaîne d'addition pour un exposant k par celui d'une chaîne d'addition passant par un entier ℓ ($\ell < k$) et $k \bmod \ell$, et celui d'une chaîne pour $k \div \ell$ grâce à la relation (10.1).

$$(1, \ldots, \ell, \ldots, k) = (1, \ldots, k \bmod \ell, \ldots, l) \otimes (1, \ldots, k \div \ell) \oplus (k \bmod \ell). \tag{10.1}$$

Remarquons déjà que pour $\ell = k \div 2$, les chaînes d'addition générées sont identiques à celles de la méthode binaire. Par contre, pour $\ell = 2^{\text{taille}(k) \div 2}$, les auteurs obtiennent expérimentalement de meilleurs résultats. L'algorithme 10.5 met en œuvre cette méthode.

```
procedure chain_exp(k)
if k = 2^ℓ then return((1, 2, 4, ..., 2^ℓ)) endif
if k = 3 then return((1, 2, 3)) endif
return(chain(k, 2^(taille(k)÷2)))
endprocedure

procedure chain(k, j)
u := k ÷ j ; v := k mod j
if v = 0 then
  return(chain_exp(j) ⊗ chain_exp(u))
else
  return(chain(j, v) ⊗ chain_exp(u) ⊕ v)
endif
endprocedure
```

FIG. 10.5 – Algorithme euclidien pour calculer des chaînes d'addition.

Par exemple, une chaîne d'addition pour 957 (13 opérations) est

$$(1, 2, 3, 6, 7, 14, 28, 29, 58, 116, 119, 232, 464, 928, 957).$$

En pratique, ces algorithmes conduisent seulement à des gains de 12% (cf. graphe 10.7). Cependant, notons que l'intérêt de ces méthodes est de pouvoir calculer des chaînes d'addition qui contiennent un certain nombre d'entiers fixés par avance.

10.2.2 Chaîne d'addition-soustraction

Méthode binaire signée

Sur une courbe elliptique, une soustraction de points a le même coût qu'une addition. Ainsi, au lieu de calculer $15P$ avec la méthode binaire par

$$15P = 2(2(2P + P) + P) + P \text{ (6 additions)},$$

on peut faire

$$15P = 2(2(2(2P))) - P, \text{ (4 additions, 1 soustraction)}.$$

Une généralisation de la méthode binaire basée sur ce constat et appelée la méthode binaire signée a été proposé par Morain et Olivos [93].

On peut décomposer cet algorithme en deux étapes. La première étape consiste à remplacer l'écriture de k en base 2,

$$k = \sum_{i=0}^{s} k_i 2^i, \ k_i \in \{0, 1\},$$

par une écriture de k en base "2 signée" (ici, $\overline{1} = -1$),

$$k = \sum_{i=0}^{s} k_i 2^i, \ k_i \in \{\overline{1}, 0, 1\}.$$

On applique pour cela à l'écriture binaire de k la transformation

$$1^a 0 1^b \to 1 0^a \overline{1} 0^{b-1} \overline{1}. \tag{10.2}$$

On construit ainsi deux entiers k_+ et k_- tels que l'on ait $k = k_+ - k_-$. Ne reste plus alors dans une deuxième étape qu'à appliquer la méthode binaire pour calculer

$$kP = k_+ P - k_- P, \tag{10.3}$$

comme dans l'algorithme 10.6.

```
procedure binaire_signé_exp(P, k)
# Initialisation
s := 0;
while k ≠ 0 do
  κ[s] := k mod 2; k := k ÷ 2; s := s + 1
endwhile
b := 0; R := O; Q := P
# Boucle principale
for i := 0, ..., s − 1 do
  if κ[0] = 0 then
    if b = 2 then
      b := 1
    else if b = 1 then
      R := R + Q; Q := 4Q; b := 0
    else
      Q := 2Q
    endif
  else
    if b = 0 then
      b := 1
    else if b = 1 then
      R := R − Q; Q := 4Q; b := 2
    else
      Q := 2Q
    endif
  endif
endfor
R := R + Q
return(R)
endprocedure
```

FIG. 10.6 – Méthode binaire signée.

Par exemple, une chaîne d'addition pour 957 (13 opérations) est

$$(1, 2, 4, -3, 8, 16, 32, 64, -67, 128, 256, 512, 1024, 957).$$

En pratique, un gain de 10% par rapport à la méthode binaire est possible (cf. graphe 10.7).

Méthode m-window

Une généralisation de la méthode binaire signée construite sur le même schéma que la méthode m-aire a été proposée par Kenji Koyama et Yukio Tsuroka [60]. Elle est appelée la méthode "binaire des fenêtres signées" ou encore "m-window". Elle combine les idées de la méthode binaire signée et de la méthode m-aire. Ainsi, dans une première étape on transforme l'écriture binaire de l'exposant k en une écriture binaire signée par la transformation (10.2), mais à la seconde étape, on utilise la méthode m-aire pour évaluer (10.3).

Par exemple, une chaîne d'addition pour 957 (13 opérations) est

$$(1, 2, 3, 4, 8, 16, 15, 30, 60, 120, 240, 480, 960, 957).$$

Cette méthode permet d'atteindre asymptotiquement un coût de $(1+1/(m+3/2))\log_2(k)$ opérations au prix d'un stockage de 2^{m-1} registres, c'est-à-dire un gain de $1/3-2/(6m+3)$ par rapport à la méthode binaire. En pratique avec $m = 9$ et $k \simeq 2^{14}$, un gain de 25% peut être obtenu (cf. graphe 10.7).

10.2.3 Résultats

Pour comparer ces cinq algorithmes, on a mesuré expérimentalement, pour des exposants de taille variant de 640 à 6400 bits, la moyenne des longueurs de chaînes d'addition obtenues avec 1000 entiers aléatoires divisée par la taille de ces entiers. Les résultats sont répertoriés sur le graphe 10.7.

Pour les méthodes m-aire et m-window, l'entier m utilisé (compris entre 4 et 10 suivant les exposants) est choisi expérimentalement pour des résultats optimaux.

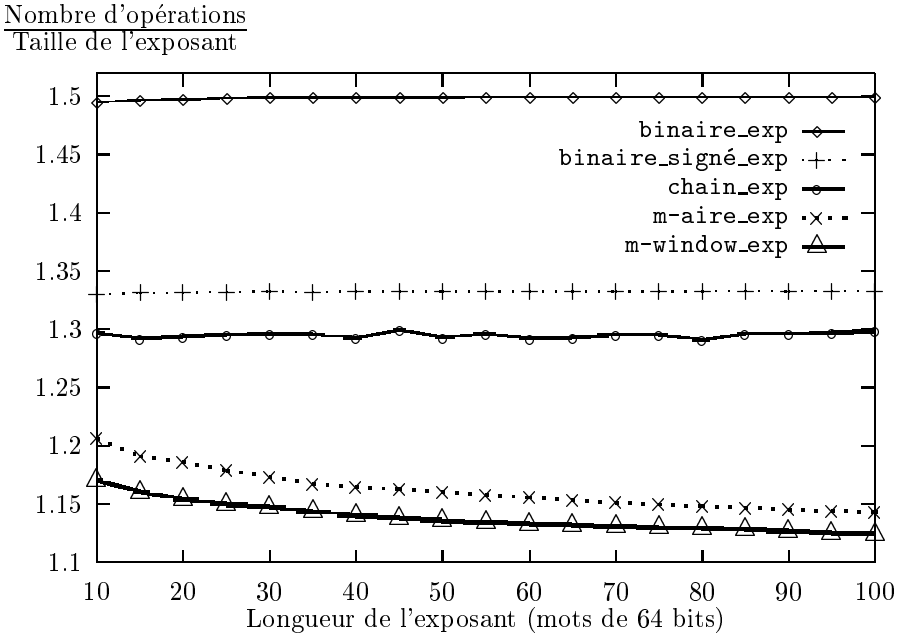


FIG. 10.7 – Longueurs de chaînes d'addition-soustraction ramenées à la taille des exposants.

Les algorithmes m-aire et m-window obtiennent clairement les meilleurs résultats, et ce d'autant plus que pour des exposants de taille croissante, une croissance associée de m fait tendre asymptotiquement la longueur moyenne des chaînes d'addition calculées vers l'optimum théorique.

C'est pourquoi, dans `ZEN`, d'une part, Chabaud a adapté l'algorithme m-aire de `BigMod` au cas des corps finis et d'autre part, l'auteur a programmé un algorithme m-window pour le cas des courbes elliptiques.

Chapitre 11

Quelques perfectionnements à l'algorithme SEA

La mise en œuvre des résultats du chapitre 3 en coordination avec ceux de la partie II suffit déjà amplement pour calculer la cardinalité d'une courbe elliptique définie sur un corps fini de moins de 10^{300} éléments. Pour atteindre des tailles supérieures, il devient important d'optimiser les algorithmes utilisés. Dans cette optique, des techniques de type "pas de bébé, pas de géant" [107] ont été appliquées avec succès à de nombreux endroits de l'algorithme `SEA` et ont conduit à des gains appréciables sans pour autant en changer la complexité asymptotique.

Précisément, nous faisons dans une première partie le point sur les méthodes qui ont été proposées pour optimiser les algorithmes du chapitre 3 (section 11.1) avant d'expliquer plus en détail dans une deuxième partie comment Atkin combine les informations obtenues sur $c \bmod \ell$ (notamment quand nous avons plusieurs candidats pour $c \bmod \ell$) par un algorithme appelé "tri-recherche" (section 11.2). Enfin, nous proposons à notre tour une amélioration de ce dernier algorithme qui nous permet d'accélérer les calculs par un facteur supérieur à deux.

11.1 Optimisations de l'algorithme de Schoof.

Nous rappelons ici comment nous pouvons appliquer des optimisations de type "pas de bébé, pas de géant" aux étapes les plus coûteuses de l'algorithme `SEA` ; la factorisation de $\Phi_\ell^{c,*}(X,Y)$ [110], la recherche d'un entier κ satisfaisant $\phi_E(P) = \kappa P$ dans la phase finale de l'algorithme d'Elkies [36, 94] ainsi que la recherche d'un entier θ satisfaisant $\phi_E^2(P) + qP = \theta\phi_E^2(P)$ dans l'algorithme de Schoof [36].

11.1.1 Factorisation des équations modulaires

Avec les notations de la proposition 35, les algorithmes du chapitre 3 nécessitent, pour environ un ℓ sur deux, la factorisation de $\Phi_\ell^{c,*}(X, j_E)$ afin de déterminer l'entier r correspondant. Ce problème est le même que celui auquel Shoup doit faire face à l'étape "distinct degree factorization" pour factoriser des polynômes définis sur $\mathbb{F}_q$ [110].

L'idée est que pour tout entier i et j, le polynôme $X^{q^i} - X^{q^j}$ est divisible par les polynômes irréductibles de $\mathbb{F}_q[X]$ dont les degrés divisent $i - j$. Ainsi, si nous notons B, le plus grand entier de l'ensemble

$$R = \left\{ r \in \mathbb{N}, \left(\frac{q}{\ell}\right) = (-1)^{(\ell \pm 1)/r} \right\}$$

inférieur strictement à $\max(R)$, et $l = \lceil \sqrt{B} \rceil$, $m = \lfloor B/l \rfloor$, il nous suffit de tester pour tous les entiers i de $[1, l]$ et j de $[1, m]$ si nous avons

$$X^{q^i} = X^{q^{jl}} \text{ dans } \mathbb{F}_q[X]/(\Phi_\ell^{c,*}(X, j_E)).$$

Si cela se produit, nous trouvons $r = jl - i$, sinon $r = \max(R)$. Cette amélioration remplace donc $O(\ell)$ compositions $X^{q^{i-1}} \circ X^q$ par $O(\sqrt{\ell})$ compositions $X^{q^{i-1}} \circ X^q$ et $O(\sqrt{\ell})$ compositions $X^{q^{l(i-1)}} \circ X^{q^l}$.

Avec cette optimisation, la phase des précalculs de $X^{2q}, \ldots, X^{(\ell-1)q}$ et de $X^{2q^l}, \ldots, X^{(\ell-1)q^l}$ utilisée classiquement pour la composition des polynômes devient le principal coût. Pour le réduire, nous suivons Shoup et préférons substituer à cette méthode l'algorithme de Brent et Kung [15] qui est aussi du type pas de bébé et pas de géant. Brièvement, pour composer deux polynômes $g(X)$ et $h(X)$ de degré $\ell - 1$, il suffit d'écrire

$$g(X) = \sum_{0 \leqslant i \leqslant (\ell-1)/t} g_i(X) X^{it},$$

où $t \simeq \sqrt{\ell}$ et les polynômes $g_i(X)$ sont de degrés bornés par t. Il ne reste plus alors qu'à précalculer $h^2(X)$, $h^3(X), \ldots h^t(X)$ et $h^{2t}(X)$, $h^{3t}(X), \ldots h^{t^2}(X)$ pour réaliser efficacement la composition $g(h(X))$. Notons que cette méthode est d'autant plus valable lorsqu'il y a, comme dans notre cas, besoin de composer de nombreux polynômes avec un polynôme fixé et qu'alors, des algorithmes de multiplication polynomiale par transformées de Fourier discrètes sont particulièrement adaptés.

11.1.2 Algorithme d'Elkies

Avec les entiers ℓ pour lesquels $\Phi_\ell^{*,c}(X, j_E)$ possède deux racines dans $\mathbb{F}_q$, il est possible, comme expliqué au chapitre 3, de construire une extension de degré $(\ell - 1)$ de $\mathbb{F}_q$,

$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y^2 - X^3 - a_2X^2 - a_4X^4 - a_6) \text{ avec } \mathbb{X} = \mathbb{F}_q[X]/(h(X))$$

de sorte que $P = (X, Y)$ soit un point de ℓ-torsion. La phase terminale de l'algorithme d'Elkies consiste alors à trouver l'entier κ, $1 \leqslant \kappa < \ell$, tel que $\phi_E(P) = \kappa P$. En pratique nous calculons d'une part, Y^q dans $\mathbb{Y}$, et d'autre part, $[2]P$, $[3]P$, $\ldots$ jusqu'à ce que l'ordonnée d'un multiple κP de P soit égale à Y^q. Une justification de cette méthode vient d'un théorème d'Atkin [5].

Théorème 52. *Soient E une courbe elliptique définie par $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$ sur un corps fini $\mathbb{F}_q$ de caractéristique p, ℓ un nombre premier impair et $h(X) = X^d + h_1X^{d-1} + \cdots + h_d$ un facteur de degré $d = (\ell-1)/2$ de f_ℓ correspondant à une valeur propre κ et à un sous-groupe $\mathbb{E}$ d'ordre ℓ de $E[\ell]$. Supposons que Y^q est égal à l'ordonnée de $[\lambda]_E$ dans $\mathbb{Y}$ où $\lambda \in \mathbb{F}_\ell$, alors $\lambda = \kappa$ si $\ell = 2 \bmod 3$ ou si le coefficient de degré $(\ell - 3)/2$ de $h(X)$ est non nul.*

La meilleure alternative à ce schéma est un algorithme de Müller appelé "funny baby giant step" [94] complété par un résultat de Dewaghe [36]. La première idée est de remplacer la recherche de κ par celle de deux entiers $i \in [1, B]$ et $j \in [1, 2B]$ tels que

$$j\phi_E(P) = iP \tag{11.1}$$

et d'en déduire $\kappa = i/j \bmod \ell$. Il s'avère que le plus petit entier B pour lequel il existe deux tels entiers i et j est inférieur à $\sqrt{\ell}$ quand $\ell < 1000$. Par exemple, pour $\ell = 997$, nous avons $B = 22$. L'algorithme consiste alors à précalculer les points iP pour ensuite trouver lors du calcul des points $j\phi_E(P)$ l'entier j pour lequel l'équation (11.1) est satisfaite.

Si cet algorithme remplace en moyenne $(\ell - 1)/4$ sommes de points dans $E(\mathbb{Y})$ par $2\sqrt{\ell}$ sommes, il nécessite par contre le calcul de X^q dans $\mathbb{X}$. Globalement, le gain semble donc quasi nul par rapport à la méthode initiale. Cependant, comme il se trouve que les entiers i et j sont petits en pratique, il devient efficace d'utiliser les formules du théorème 10 pour obtenir dans $\mathbb{X}$ les abscisses de $i\phi_E(P)$ et jP. Or ces formules ont uniquement besoin des abscisses de $\phi_E(P)$ et P : il n'est donc plus nécessaire de calculer Y^q.

En échange de ce gain, un nouveau problème surgit. En effet, puisque nous n'avons plus Y^q, nous ne sommes plus en mesure de distinguer κ de $-\kappa$. Par bonheur, un résultat de Dewaghe nous tire d'affaire dans de nombreux cas.

Théorème 53. *Avec les notations du théorème 52, soit $\overline{h}(X) = X^s + \overline{h}_1X^{s-1} + \cdots + \overline{h}_s$ un facteur de $h(X)$ de degré le semi ordre s de κ (cf. définition 18) dans $\mathbb{F}_\ell$.*

Alors, si pour un point P non nul de $\mathbb{E}$, nous avons l'abscisse de $j\phi_E(P)$ qui est égale à celle de iP, nous obtenons en posant $\kappa_0 = i\,j^{-1} \bmod \ell$:

– *si* $\ell = 3 \bmod 4$,

$$\kappa = \left(\frac{\kappa_0}{\ell}\right)\left(\frac{\rho}{q}\right)\kappa_0 \qquad \textit{quand } p > 2,\ a_1 = a_3 = 0,$$

$$\kappa = \begin{cases} \left(\frac{\kappa_0}{\ell}\right)\kappa_0 & \textit{si } \mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}\left(h_1 + a_6\frac{h_{d-1}^2}{h_d^2}\right) = 0, \\ -\left(\frac{\kappa_0}{\ell}\right)\kappa_0 & \textit{sinon}\ , \end{cases} \qquad \textit{quand } p{=}2,\ \begin{cases} a_1 = 1, \\ a_2 = a_3 = a_4 = 0, \end{cases}$$

où $\rho = \mathrm{Resultant}(h(X), X^3 + a_2X^2 + a_4X + a_6)$ *dans* $\mathbb{F}_q$.

– *si* $\ell = 1 \bmod 4$ *et* s *impair*,

$$\kappa = \kappa_0^s\left(\frac{\rho}{q}\right)\kappa_0 \qquad \textit{quand } p > 2,\ a_1 = a_3 = 0,$$

$$\kappa = \begin{cases} \kappa_0^s\left(\frac{\kappa_0}{\ell}\right)\kappa_0 & \textit{si } \mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}\left(\overline{h}_1 + a_6\frac{\overline{h}_{s-1}^2}{\overline{h}_s^2}\right) = 0, \\ -\kappa_0^s\left(\frac{\kappa_0}{\ell}\right)\kappa_0 & \textit{sinon}\ , \end{cases} \qquad \textit{quand } p{=}2,\ \begin{cases} a_1 = 1, \\ a_2 = a_3 = a_4 = 0, \end{cases}$$

où $\rho = \mathrm{Resultant}(\overline{h}(X), X^3 + a_2X^2 + a_4X + a_6)$ *dans* $\mathbb{F}_q$.

Pour $\ell = 1 \bmod 4$ et un semi ordre s pair, nous n'avons malheureusement pas d'autre alternative que de factoriser $h(X)$ pour obtenir modulo un facteur de degré s de $h(X)$, et de calculer les ordonnées de $j\phi_E(P)$ et iP afin de pouvoir conclure.

Enfin, une optimisation supplémentaire consiste, plutôt que calculer des inverses dans $\mathbb{X}$, à conserver séparément numérateur $A_i(X)$ et dénominateur $B_i(X)$ des abscisses des points iP donnés par le théorème 10. Il est alors possible, de tester rapidement si

$$\frac{A_j(X^q)}{B_j(X^q)} = \frac{A_i(X)}{B_i(X)} \bmod h(X),$$

en vérifiant que le résultat d'une application linéaire aléatoire L appliquée à $A_i(X)B_j(X^q)$ est égal à celui de $B_i(X)A_j(X^q)$ (cf. [110]).

11.1.3 Algorithme de Schoof

Pour un point $P = (X, Y)$ de ℓ torsion construit comme expliqué au chapitre 3 dans une extension de degré $r(\ell - 1)$ de $\mathbb{F}_q$,

$$\mathbb{Y} = \mathbb{X}[Y]/(Y^2 + a_1XY + a_3Y^2 - X^3 - a_2X^2 - a_4X^4 - a_6) \text{ avec } \mathbb{X} = \mathbb{F}_q[X]/(h(X)),$$

Dewaghe [36] propose, dans le même esprit que l'algorithme "funny baby giant step" de Müller, une variante des "pas de bébé, pas de géant" ne nécessitant pas le calcul des ordonnées pour trouver l'entier θ tel que

$$\phi_E^2(P) + qP = \theta\phi_E(P).$$

Ainsi, il recherche encore deux entiers $i \in [1, B]$ et $j \in [1, 2B]$ (B déterminé comme dans la section précédente) tels que

$$j(\phi_E^2(P) + qP) = i\phi_E(P).$$

S'il ne semble pas difficile de calculer l'abscisse du point $i\phi_E(P)$ à partir de celle de $\phi_E(P)$ (cf. formules du théorème 10), un problème se pose par contre pour $j(\phi_E^2(P) + qP)$ puisque l'abscisse de la somme de $\phi_E^2(P)$ et de qP dépend des ordonnées de ces points. Dewaghe contourne cette difficulté en testant en fait s'il existe deux entiers i et j pour lesquels nous avons *simultanément*

$$j\phi_E^2(P) \pm jqP = i\phi_E(P). \tag{11.2}$$

Et il montre que si cela se produit pour un nombre premier ℓ correspondant au cas (iii) du théorème 35, alors la seule possibilité est $j\phi_E^2(P) + jqP = i\phi_E(P)$.

Pour tester l'équation (11.2), il suffit de remarquer que les fonctions symétriques des abscisses X_3 et X_4 de la différence et la somme de deux points (X_1, Y_1) et (X_2, Y_2) ne dépendent pas de Y_1 et de Y_2. Ainsi, X_3 et X_4 sont les racines en X du polynôme

$$D(X_1, X_2, X) = N(X_1, X_2)X^2 - S(X_1, X_2)X + P(X_1, X_2),$$

où

$$\begin{aligned} N(X_1, X_2) &= (X_1 - X_2)^2, \\ S(X_1, X_2) &= (X_2 + X_1)\,(a_1\,a_3 + 2\,a_4 + 2\,X_1\,X_2) + \left(4\,a_2 + {a_1}^2\right) X_1\,X_2 + 4\,a_6 + {a_3}^2, \\ P(X_1, X_2) &= (X_1\,X_2 - a_4)\,(X_1\,X_2 - a_4 - a_1\,a_3) - (X_1 + X_2 + a_2)\left({a_3}^2 + 4\,a_6\right) - {a_1}^2 a_6. \end{aligned}$$

L'équation (11.2) est donc satisfaite si et seulement si

$$D(X_{j\phi_E^2(P)}, X_{jqP}, X_{i\phi_E(P)}) = 0. \tag{11.3}$$

L'algorithme complet découle de ces remarques. Bien sûr, numérateurs et dénominateurs des abscisses des points $j(\phi_E^2(P)$, $jqP)$ et $i\phi_E(P)$ sont calculés séparément à l'aide des formules du théorème 10 et l'équation (11.3) est alors évaluée rapidement à l'aide d'une application linéaire aléatoire comme dans l'algorithme de Müller.

L'inconvénient majeur est encore une fois que nous ne sommes plus en mesure de distinguer θ de $-\theta$. Pour se tirer d'affaire, du moins pour certains ℓ, nous avons un autre résultat de Dewaghe.

Théorème 54. *Avec les notations du théorème 35 où E est donnée sur $\mathbb{F}_q$ par $E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$, soit $h(X) = X^{rd} + h_1X^{rd-1} + \cdots + h_{rd}$ un facteur de f_ℓ de degré rd où $d = (\ell-1)/2$. Soit de plus $\overline{h}(X) = X^{rs} + \overline{h}_1X^{rs-1} + \cdots + \overline{h}_{rs}$ un facteur de $h(X)$ de degré rs où s est le semi ordre de ω dans $\mathbb{F}_\ell$ avec ω, l'entier tel que $\phi^r_{E[\ell]} = [\omega]_{E[\ell]}$. Alors, si pour un point P non nul de $E[\ell]$, nous avons l'abscisse de $j\phi_E^2(P) + jqP$ qui est égale à celle de $i\phi_E(P)$, nous avons en posant $\theta_0 = i\,j^{-1} \bmod \ell$:*

– *si r est impair et $\ell = 3 \bmod 4$,*

$$\theta = \left(\frac{\omega_0}{\ell}\right)\left(\frac{\rho}{q}\right)\theta_0 \qquad \text{quand } p > 2,\ a_1 = a_3 = 0,$$

$$\theta = \begin{cases} \left(\dfrac{\omega_0}{\ell}\right)\theta_0 & \text{si } \mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}\left(h_1 + a_6\dfrac{h_{rd-1}^2}{h_{rd}^2}\right) = 0, \\ -\left(\dfrac{\theta_0}{\ell}\right)\theta_0 & \text{sinon}, \end{cases} \qquad \text{quand } p{=}2,\ \begin{cases} a_1 = 1, \\ a_2 = a_3 = a_4 = 0, \end{cases}$$

où $\rho = \mathrm{Resultant}(h(X), X^3 + a_2X^2 + a_4X + a_6)$.

– *si r est impair et $\ell = 1 \bmod 4$ et s impair,*

$$\theta = \theta_0^s\left(\frac{\rho}{q}\right)\theta_0 \qquad \text{quand } p > 2,\ a_1 = a_3 = 0,$$

$$\theta = \begin{cases} \theta_0^s\left(\dfrac{\kappa_0}{\ell}\right)\theta_0 & \text{si } \mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}\left(\overline{h}_1 + a_6\dfrac{\overline{h}_{sd-1}^2}{\overline{h}_{sd}^2}\right) = 0, \\ -\theta_0^s\left(\dfrac{\kappa_0}{\ell}\right)\theta_0 & \text{sinon}, \end{cases} \qquad \text{quand } p{=}2,\ \begin{cases} a_1 = 1, \\ a_2 = a_3 = a_4 = 0, \end{cases}$$

où $\rho = \mathrm{Resultant}(\overline{h}(X), X^3 + a_2X^2 + a_4X + a_6)$.

Pour r pair, ou $\ell = 1 \bmod 4$ et un semi ordre s pair, nous n'avons malheureusement pas d'autre alternative que de calculer, modulo un facteur de degré sd de $h(X)$, les ordonnées de $j\phi_E^2(P)$, jqP et $i\phi_E(P)$ afin de pouvoir conclure.

11.2 Algorithmes "tri-recherche"

Le nombre de points d'une courbe elliptique E définie sur un corps fini $\mathbb{F}_q$ est égal à $q+1-c$ où l'entier c satisfait $|c| \leqslant 2\sqrt{q}$. Les algorithmes du chapitre 3 ont pour objectif d'obtenir un nombre minimal de

candidats pour c modulo des entiers ℓ qui d'une part, satisfont

$$\prod_{\substack{\ell \text{ premiers} \\ \text{entre eux}}} \ell > 4\sqrt{q}, \tag{11.4}$$

et d'autre part, sont aussi petits que possible. L'idéal serait d'avoir pour chaque ℓ, un unique candidat $c \bmod \ell$, puisqu'alors une simple application du théorème chinois nous permettrait de retrouver c.

Malheureusement, ces algorithmes ne fournissent une unique valeur pour $c \bmod \ell$ que pour approximativement la moitié des ℓ, généralement les nombres premiers dits d'Elkies et leurs puissances. Pour les autres ℓ, ceux dits d'Atkin, nous avons un nombre pair de possibilités. Une solution qui ne change pas la complexité asymptotique de l'algorithme est de ne conserver que les entiers d'Elkies, quitte à utiliser des ℓ plus grands que ne l'impose la condition (11.4). En pratique, il est préférable de diminuer le nombre des entiers ℓ d'Elkies nécessaires en les remplaçant par quelques entiers ℓ d'Atkin, ceux avec un nombre de candidats pour $c \bmod \ell$ petit. En contrepartie, nous avons plusieurs candidats, disons un nombre C, à tester dans l'intervalle $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$.

Une approche naïve pour trouver l'entier c est bien sûr d'énumérer les C candidats. Elle nécessite $O(C)$ additions dans $E(\mathbb{F}_q)$. Un algorithme de type "pas de bébés, pas de géants" dû à Atkin [5] permet néanmoins de faire chuter en moyenne ce nombre d'opérations à environ $2\sqrt{3C}$ additions dans $E(\mathbb{F}_q)$ grâce à une partition astucieuse des ℓ (section 11.2.1). Une fois décrit cet algorithme (section 11.2.2), nous en proposons une variante qui nous permet de réaliser cette énumération en seulement $\sqrt{2C}$ additions dans $E(\mathbb{F}_q)$ en moyenne (section 11.2.3).

11.2.1 Préparation des données

Au début de cette étape, nous disposons d'un ensemble $\mathcal{L}$ d'entiers ℓ vérifiant la condition (11.4). Pour chaque ℓ, nous avons un ensemble $\mathcal{T}_\ell$ de candidats pour $c \bmod \ell$. Précisément avec les algorithmes des chapitres 3 et 11, nous avons soit $\#\mathcal{T}_\ell = 1$ (ℓ de type "Elkies"), soit $\#\mathcal{T}_\ell$ pair avec la propriété supplémentaire suivante,

$$\forall \theta \in \mathcal{T}_\ell,\ -\theta \bmod \ell \in \mathcal{T}_\ell. \tag{11.5}$$

Algorithme de Nicolas

Comme nous ne pouvons utiliser qu'une partie des entiers d'Atkin, le produit des $\ell \in \mathcal{L}$ est généralement bien supérieur à la borne $4\sqrt{q}$. Il est intéressant pour la suite d'en extraire un sous-ensemble nous apportant un maximum d'information sur c, c'est-à-dire tel que

$$\prod \ell > 4\sqrt{q} \text{ et } \prod \#\mathcal{T}_\ell \text{ minimal.} \tag{11.6}$$

Atkin ne donne aucun algorithme pour cela et une recherche exhaustive devient vite impraticable. Nous allons décrire pour notre part un algorithme de type "programmation dynamique" dû à Nicolas [95] qui, quoique de complexité exponentielle en $\#\mathcal{L}$ pour certains problèmes, se comporte de façon satisfaisante avec nos données.

Cette méthode nécessite une fonction de coût $C(\ell)$ que nous définissons comme étant égale à 1 pour $\ell = 1$, à $\#\mathcal{T}_\ell$ pour $\ell \in \mathcal{L}$, à $\prod C(\ell_i)$ pour $\prod \ell_i$ si les ℓ_i sont des entiers distincts de $\mathcal{L}$ et à ∞ sinon. Son but est de calculer ce que Nicolas appelle des "champions".

Définition 23. *On dit qu'un entier B est un champion pour une fonction de coût C si $C(B) < \infty$ et*

$$\forall n \in \mathbb{N}^*,\ n \geqslant B \Rightarrow C(n) \geqslant C(B).$$

Non seulement le nombre de ces champions est fini puisque leurs facteurs ne peuvent être que des entiers ℓ avec $C(\ell) < \infty$, mais encore, connaissant ces champions, il est facile de résoudre notre problème puisque le produit des ℓ de l'équation (11.6) n'est jamais que l'entier de coût minimal dans l'ensemble des champions supérieurs à $4\sqrt{q}$. Clairement, cet ensemble est non vide puisque $\prod_{\ell \in \mathcal{L}} \ell$ est l'un de ces

champions. Pour les déterminer explicitement, l'algorithme numérote tout d'abord les ℓ tels que $C(\ell) < \infty$ par ordre de coûts croissants, soit donc

$$\{\ell_0 = 1, \ell_1, \ldots, \ell_k\} = \{\ell \in \mathbb{N}, C(\ell) < \infty\}.$$

Puis il calcule ce que Nicolas appelle les j-champions.

Définition 24. *Étant donnée une fonction de coût C finie sur $\{\ell_0, \ldots, \ell_k\}$, un entier B est un j-champion pour C ($j \in \{0, \ldots, k\}$) si B est un champion pour la fonction de coût $C_j(\ell)$ définie par $C_j(\ell_i) = C(\ell_i)$ pour $i = 1, \ldots, j$ et $C_j(\ell) = \infty$ sinon.*

L'algorithme consiste à calculer les j-champions pour j croissant de 2 jusqu'à k, les k-champions étant alors les champions recherchés. Notons déjà qu'il n'y a que deux 2-champions, $\{1, \ell_1\}$. Ensuite, nous obtenons les j-champions à partir des $(j-1)$-champions avec le lemme suivant.

Lemme 9. *Avec les notations de la définition 24, si B est un j-champion, alors B/ℓ_j est un $(j-1)$-champion quand ℓ_j divise B ou B est un $(j-1)$-champion sinon.*

Démonstration : si $B \neq 0 \bmod \ell_j$, alors $\forall n \geqslant B, C(n) \geqslant B$ et donc B est un $(j-1)$-champion. Sinon, pour tout $n \geqslant B/\ell_j$, nous avons $n\ell_j \geqslant B$ et $C(n\ell_j) \geqslant C(B)$. Comme C est une fonction de coût, nous avons $C(n\ell_j) = C(n)C(\ell_j)$, d'où $C(n) \geqslant C(B)/C(\ell_j) \geqslant C(B/\ell_j)$ et donc B/ℓ_j est un $(j-1)$-champion. □

À la j-ième itération, l'algorithme extrait donc les j-champions à partir des $(j-1)$-champions $\{B_1, \ldots, B_n\}$ par un simple tri de l'ensemble

$$\{B_1, \ldots, B_n\} \cup \{B_1\ell_j, \ldots, B_n\ell_j\}.$$

L'algorithme original de Nicolas de la figure 11.1 n'est qu'une reformulation astucieuse de ce procédé.

```
procedure champions({ℓ_1, ..., ℓ_k}, C : ℓ → C(ℓ))
n := 2; B_1 := 1; B'_1 := 1; B'_2 := ℓ_1
for j := 2, ..., k do
  # B' contient les j-champions ordonnés par ordre croissant.
  i := 1; i_1 := 2; i_2 := 1
  while i_1 ⩽ n do
    if B'_{i_1} < ℓ_j B'_{i_2} then
      b := B'_{i_1}; i_1 := i_1 + 1
    else
      b := ℓ_j B'_{i_2}; i_2 := i_2 + 1
    endif
    while C(b) < C(B_i) do i := i − 1 endwhile
    i := i + 1; B_i := b
  endwhile
  while i_2 ⩽ n do
    b := ℓ_j B'_{i_2}; i_2 := i_2 + 1
    while C(b) < C(B_i) do i := i − 1 endwhile
    i := i + 1; B_i := b
  endwhile
  n := i
  for i := 1, ..., n do B'_i := B_i endwhile
endfor
return([B_2, ..., B_n])
```

FIG. 11.1 – Algorithme de Nicolas pour le calcul de champions.

Exemple : pour calculer les champions de l'ensemble $\{5,3,7\}$ associé à la fonction de coût C définie par $C(1)=1$, $C(5)=4$, $C(3)=2$ et $C(7)=2$, l'algorithme procède ainsi. Initialement, les 2-champions sont 1 et 5. Les 3-champions (coûts donnés entre parenthèses) sont donc parmi

$$\{1(1),5(4)\}\cup\{3(2),15(8)\}\,.$$

Après un tri, nous nous apercevons que les entiers de cet ensemble sont tous des 3-champions. De même, les 4-champions sont parmi

$$\{1(1),3(2),5(4),15(8)\}\cup\{7(1),21(2),35(4),105(8)\}\,.$$

Après tri, cet ensemble est égal à

$$\{1(1),3(2),5(4),7(1),15(8),21(2),35(4),105(8)\}\,.$$

Or 5 et 15 ne peuvent pas être des champions car nous avons respectivement $7\geqslant 5$ et $C(7)<C(5)$, $21\geqslant 15$ et $C(21)<C(15)$. Les champions sont donc

$$\{1(1),3(2),7(1),21(2),35(4),105(8)\}\,.$$

Partition des modulos

Comme manifestement, le sous-ensemble de $\mathcal{L}$ obtenu après l'appel de `champions` contiendrait les ℓ avec $\#\mathcal{T}_\ell=1$, nous préférons utiliser `champions` sur l'ensemble $\mathcal{L}-\mathfrak{U}$ où

$$\mathfrak{U}=\{\ell,\#\mathcal{T}_\ell=1\}.$$

En posant $M_\mathfrak{U}=\prod_{\ell\in\mathfrak{U}}\ell$, nous obtenons ainsi des ℓ tels que

$$\prod\ell>\frac{4\sqrt{q}}{M_\mathfrak{U}}\ \text{et}\ \prod\#\mathcal{T}_\ell\ \text{minimal.}$$

Les deux algorithmes que nous considérons dans les sections 11.2.2 et 11.2.3 nécessitent alors une partition de ces ℓ en deux sous-ensembles $\mathfrak{B}$ (ensemble des "bébés") et $\mathfrak{G}$ (ensemble des "géants") tels que

$$\prod_{\ell\in\mathfrak{B}}\#\mathcal{T}_\ell\simeq\eta\prod_{\ell\in\mathfrak{G}}\#\mathcal{T}_\ell$$

avec η valant respectivement $3/4$ et $1/2$. Ces algorithmes n'étant utilisables en pratique que si $\prod_{\ell\in\mathfrak{B}\cup\mathfrak{G}}\#\mathcal{T}_\ell\leqslant 10^{15}$, une simple recherche exhaustive pour déterminer $\mathfrak{B}$ et $\mathfrak{G}$ convient ici parfaitement.

11.2.2 Algorithme d'Atkin

L'algorithme d'Atkin repose sur une formulation adaptée du théorème chinois.

Reformulation du théorème chinois

À partir des trois ensembles $\mathfrak{U}$, $\mathfrak{B}$ et $\mathfrak{G}$ définis dans la section 11.2.1, nous posons

$$M_\mathfrak{U}=\prod_{\ell\in\mathfrak{U}}\ell,\ M_\mathfrak{B}=\prod_{\ell\in\mathfrak{B}}\ell\ \text{et}\ M_\mathfrak{G}=\prod_{\ell\in\mathfrak{G}}\ell.$$

Par simple application du théorème chinois, il n'est alors pas difficile d'obtenir un unique candidat u pour $c \bmod M_\mathfrak{U}$, $C_\mathfrak{B}$ $(=\prod_{\ell\in\mathfrak{B}}\#\mathcal{T}_\ell)$ candidats $\{b_1,\ldots,b_{C_\mathfrak{B}}\}$ pour $c \bmod M_\mathfrak{B}$ et $C_\mathfrak{G}$ $(=\prod_{\ell\in\mathfrak{G}}\#\mathcal{T}_\ell)$ candidats $\{g_1,\ldots,g_{C_\mathfrak{B}}\}$ pour $c \bmod M_\mathfrak{G}$. Pour déterminer c à partir de ces résidus modulaires, l'algorithme utilise la proposition suivante.

Proposition 29. *Soient $M_{\mathfrak{U}}$, $M_{\mathfrak{B}}$ et $M_{\mathfrak{G}}$ trois entiers premiers entre eux définis comme précédemment. Soit aussi*

$$c \in \left[-\frac{M_{\mathfrak{U}} M_{\mathfrak{B}} M_{\mathfrak{G}}}{2}, \frac{M_{\mathfrak{U}} M_{\mathfrak{B}} M_{\mathfrak{G}}}{2} \right] \tag{11.7}$$

tel que

$$\begin{cases} c \bmod M_{\mathfrak{U}} = u, \\ c \bmod M_{\mathfrak{B}} \in \{b_1, \ldots, b_{C_{\mathfrak{B}}}\}, \\ c \bmod M_{\mathfrak{G}} \in \{g_1, \ldots, g_{C_{\mathfrak{G}}}\}. \end{cases} \tag{11.8}$$

Il existe alors deux entiers $r_{\mathfrak{B}}$ et $r_{\mathfrak{G}}$ tels que $c = u + M_{\mathfrak{U}}(r_{\mathfrak{B}} M_{\mathfrak{G}} + r_{\mathfrak{G}} M_{\mathfrak{B}})$ ou $c = u + M_{\mathfrak{U}}(r_{\mathfrak{B}} M_{\mathfrak{G}} + (r_{\mathfrak{G}} - M_{\mathfrak{G}}) M_{\mathfrak{B}})$ et qui vérifient

$$\begin{cases} r_{\mathfrak{B}} \in \left\{ \dfrac{b_1 - u}{M_{\mathfrak{U}} M_{\mathfrak{G}}}, \ldots, \dfrac{b_{C_{\mathfrak{B}}} - u}{M_{\mathfrak{U}} M_{\mathfrak{G}}} \right\} \bmod M_{\mathfrak{B}}, \\ r_{\mathfrak{G}} \in \left\{ \dfrac{g_1 - u}{M_{\mathfrak{U}} M_{\mathfrak{B}}}, \ldots, \dfrac{g_{C_{\mathfrak{G}}} - u}{M_{\mathfrak{U}} M_{\mathfrak{B}}} \right\} \bmod M_{\mathfrak{G}}, \end{cases} \tag{11.9}$$

et

$$-\frac{M_{\mathfrak{B}}}{2} \leqslant r_{\mathfrak{B}} \leqslant \frac{M_{\mathfrak{B}}}{2}, \ 0 \leqslant r_{\mathfrak{G}} \leqslant M_{\mathfrak{G}} \tag{11.10}$$

Démonstration : par une légère variante du théorème chinois, il n'est pas difficile de voir que les entiers c vérifiant les équations (11.8) sont donnés par

$$c(r_{\mathfrak{B}}, r_{\mathfrak{G}}) = u + M_{\mathfrak{U}}(r_{\mathfrak{B}} M_{\mathfrak{G}} + r_{\mathfrak{G}} M_{\mathfrak{B}})$$

pour les entiers $r_{\mathfrak{B}}$ et $r_{\mathfrak{G}}$ satisfaisant les équations (11.9). Or, pour de tels entiers $r_{\mathfrak{B}}$ et $r_{\mathfrak{G}}$, nous avons non seulement pour tout entier λ, $r_{\mathfrak{B}} - \lambda M_{\mathfrak{B}}$ et $r_{\mathfrak{G}} + \lambda M_{\mathfrak{G}}$ qui vérifient les relations (11.9), mais aussi $c(r_{\mathfrak{B}}, r_{\mathfrak{G}}) = c(r_{\mathfrak{B}} - \lambda M_{\mathfrak{B}}, r_{\mathfrak{G}} + \lambda M_{\mathfrak{G}})$. Choisissons donc $r_{\mathfrak{B}}$, éventuellement à une translation de $\lambda M_{\mathfrak{B}}$ près, tel que

$$-\frac{M_{\mathfrak{B}}}{2} \leqslant r_{\mathfrak{B}} \leqslant \frac{M_{\mathfrak{B}}}{2},$$

et dans ce cas, la condition (11.7) implique

$$-M_{\mathfrak{G}} \leqslant r_{\mathfrak{G}} \leqslant M_{\mathfrak{G}}.$$

Restreignant $r_{\mathfrak{G}}$ à $[0, M_{\mathfrak{G}}]$, nous avons donc $c = c(r_{\mathfrak{B}}, r_{\mathfrak{G}})$ ou $c = c(r_{\mathfrak{B}}, r_{\mathfrak{G}} - M_{\mathfrak{G}})$. □

Pour trouver c, il n'est donc plus nécessaire d'énumérer les $C_{\mathfrak{B}} C_{\mathfrak{G}}$ candidats de l'intervalle $[-M_{\mathfrak{U}} M_{\mathfrak{B}} M_{\mathfrak{G}}/2, M_{\mathfrak{U}} M_{\mathfrak{B}} M_{\mathfrak{G}}/2]$, mais simplement les $C_{\mathfrak{B}}$ candidats β_r de $[-M_{\mathfrak{B}}/2, M_{\mathfrak{B}}/2]$ pour $r_{\mathfrak{B}}$ puis les $C_{\mathfrak{G}}$ candidats γ_s de $[0, M_{\mathfrak{G}}]$ pour $r_{\mathfrak{G}}$.

Algorithme

L'algorithme découle immédiatement de la proposition 29. Il est décrit dans [94, pp. 144–147] et est constitué des étapes suivantes.

Partition : nous extrayons de $\mathcal{L}$, trois sous-ensembles $\mathfrak{U}$, $\mathfrak{B}$ et $\mathfrak{G}$ comme expliqué dans la section 11.2.1. Soient $M_{\mathfrak{U}}$, $M_{\mathfrak{B}}$ et $M_{\mathfrak{G}}$, les modulos associés respectivement aux résidus ordonnés par ordre croissant u, $\{\beta_1, \ldots, \beta_{C_{\mathfrak{B}}}\}$ et $\{\gamma_1, \ldots, \gamma_{C_{\mathfrak{G}}}\}$. Puis à partir d'un point P aléatoirement choisi sur la courbe $E(\mathbb{F}_q)$, nous calculons les points $P_{\mathfrak{U}} = (q + 1 - u)P$, $P_{\mathfrak{B}} = M_{\mathfrak{U}} M_{\mathfrak{G}} P$, $P_{\mathfrak{G}} = M_{\mathfrak{U}} M_{\mathfrak{B}} P$ et $P_q = M_{\mathfrak{U}} M_{\mathfrak{B}} M_{\mathfrak{G}} P$.

Pas de bébés : nous stockons les points $P_{\mathfrak{U}} - \beta_r P_{\mathfrak{B}}$ pour r variant de 1 jusqu'à $C_{\mathfrak{B}}$. Pour cela, nous calculons tout d'abord $P_{\mathfrak{U}} - \beta_1 P_{\mathfrak{B}}$, puis $P_{\mathfrak{U}} - \beta_{r+1} P_{\mathfrak{B}}$ par

$$P_{\mathfrak{U}} - \beta_{r+1} P_{\mathfrak{B}} = P_{\mathfrak{U}} - \beta_r P_{\mathfrak{B}} - (\beta_{r+1} - \beta_r) P_{\mathfrak{B}}.$$

Plutôt que de calculer $(\beta_{r+1}-\beta_r)\,P_{\mathfrak{B}}$ par exponentiation, Atkin conseille d'écrire l'entier $\beta_{r+1}-\beta_r$ dans une base $B \simeq 1000$,

$$\beta_{r+1}-\beta_r = \tau_0 + \tau_1 B + \cdots + \tau_i B^i,$$

d'emmagasiner au préalable

$$0P_{\mathfrak{B}},\ldots,(B-1)P_{\mathfrak{B}}\ ;BP_{\mathfrak{B}},\ldots,(B-1)BP_{\mathfrak{B}}\ ;\ldots\ ;B^iP_{\mathfrak{B}},\ldots,(B-1)B^iP_{\mathfrak{B}}$$

et donc d'obtenir $(\beta_{r+1}-\beta_r)\,P_{\mathfrak{B}}$ en au plus i additions. Enfin, pour diminuer l'espace nécessaire, nous stockons dans une table, non pas les points, mais une valeur de hachage de chacun de ces points.

Pas de géants : nous calculons, avec le même procédé que pour les pas de bébés $\gamma_s P_{\mathfrak{G}}$, pour s variant de 1 à $C_{\mathfrak{G}}$, avec $j+1$ additions. Par simple examen de la table précédente, nous vérifions pour chaque s que la valeur de hachage de $\gamma_s P_{\mathfrak{G}}$ ou celle de $\gamma_s P_{\mathfrak{G}} + P_q$ correspond dans la table à celle d'un point $P_{\mathfrak{U}} - \beta_r P_{\mathfrak{B}}$. Si cela est le cas, il ne reste plus à vérifier que $(q+1-u-M_{\mathfrak{U}}(\beta_r M_{\mathfrak{G}}+\gamma_s M_{\mathfrak{B}}))Q = O_E$ ou $(q+1-u-M_{\mathfrak{U}}(\beta_r M_{\mathfrak{G}}+(\gamma_s - M_{\mathfrak{G}})M_{\mathfrak{B}}))Q = O_E$ avec un autre point Q aléatoirement choisi sur la courbe pour conclure.

Complexité

Le coût de l'étape de partition étant négligeable, nous nous concentrons sur les deux suivantes. Dans l'étape des pas de bébés, nous avons $C_{\mathfrak{B}}$ points à calculer. Pour chacun de ces points, nous devons calculer $(\beta_{r+1}-\beta_r)\,P_{\mathfrak{B}}$ avec i additions. Nous avons donc besoin de $(1+i)C_{\mathfrak{B}}$ additions. Dans l'étape des pas de géants, nous avons en moyenne $C_{\mathfrak{G}}/2$ points à calculer. Pour chacun de ces points, nous devons non seulement calculer $(\gamma_{s+1}-\gamma_s)\,P_{\mathfrak{G}}$ avec j additions, mais encore additionner P_q pour un test supplémentaire. En conséquence $(1+j/2)C_{\mathfrak{G}}$ additions en moyenne sont nécessaires.

Le coût total de cet algorithme étant $(1+i)C_{\mathfrak{B}} + (1+j/2)C_{\mathfrak{G}}$ avec $C_{\mathfrak{B}}C_{\mathfrak{G}} = C$, il est minimal pour $(1+i)C_{\mathfrak{B}} \simeq (1+j/2)C_{\mathfrak{G}}$ et nous devons donc choisir les deux ensembles $\mathfrak{B}$ et $\mathfrak{G}$ tels que $C_{\mathfrak{B}} = \sqrt{\frac{1+j/2}{1+i}C}$, $C_{\mathfrak{G}} = \sqrt{\frac{1+i}{1+j/2}C}$. Dans ce cas, le coût total de l'algorithme est $2\sqrt{(1+i)(1+j/2)C}$. En supposant $\sqrt{M_{\mathfrak{B}}}$ et $\sqrt{M_{\mathfrak{G}}}$ de taille raisonnable (hypothèse très optimiste), nous avons $i=j=1$ et ce coût est alors de

$$2\sqrt{3C} \simeq 3.4\sqrt{C} \text{ additions sur la courbe.}$$

11.2.3 Variante du "tri-recherche" d'Atkin

Notre variante de l'algorithme d'Atkin repose, d'une part sur une modification de la proposition 29, et d'autre part sur une heuristique qui nous permet diviser le nombre d'additions par deux.

Autre reformulation du théorème chinois

L'inconvénient de la proposition 29 est qu'elle n'exploite pas la propriété (11.5), ce qui nous impose l'addition systématique de P_q dans l'étape des pas de géants. Nous proposons donc une légère variante de cette proposition qui se démontre sur le même modèle et qui nous permet de combler cette lacune.

Proposition 30. *Soient c, $M_{\mathfrak{U}}$ $M_{\mathfrak{B}}$ et $M_{\mathfrak{G}}$, quatre entiers vérifiant les conditions (11.7) et (11.8). Il existe alors trois entiers $S_{\mathfrak{G}}$, $r_{\mathfrak{B}}$ et $r_{\mathfrak{G}}$ tels que $c = u + S_{\mathfrak{G}}M_{\mathfrak{U}} + M_{\mathfrak{U}}(r_{\mathfrak{B}}M_{\mathfrak{G}} \pm r_{\mathfrak{G}}M_{\mathfrak{B}})$ et qui satisfont*

$$\begin{cases} S_{\mathfrak{G}} = \dfrac{-u}{M_{\mathfrak{U}}M_{\mathfrak{B}}} \bmod M_{\mathfrak{G}} \\ r_{\mathfrak{B}} \bmod M_{\mathfrak{B}} \in \left\{ \dfrac{b_1-u}{M_{\mathfrak{U}}M_{\mathfrak{G}}},\ldots,\dfrac{b_{C_{\mathfrak{B}}}-u}{M_{\mathfrak{U}}M_{\mathfrak{G}}} \right\} \bmod M_{\mathfrak{B}}, \\ r_{\mathfrak{G}} \bmod M_{\mathfrak{G}} \in \left\{ \dfrac{g_1}{M_{\mathfrak{U}}M_{\mathfrak{B}}},\ldots,\dfrac{g_{C_{\mathfrak{G}}}}{M_{\mathfrak{U}}M_{\mathfrak{B}}} \right\} \bmod M_{\mathfrak{G}}, \end{cases}$$

et

$$-\frac{M_{\mathfrak{B}}}{2} - S_{\mathfrak{G}}\frac{M_{\mathfrak{B}}}{M_{\mathfrak{G}}} \leqslant r_{\mathfrak{B}} \leqslant \frac{M_{\mathfrak{B}}}{2} - S_{\mathfrak{G}}\frac{M_{\mathfrak{B}}}{M_{\mathfrak{G}}},\ 0 \leqslant r_{\mathfrak{G}} \leqslant M_{\mathfrak{G}}.$$

Optimisations

Nous proposons trois améliorations de l'algorithme d'Atkin.

- Nous remplaçons les β_r et les γ_s donnés après la proposition 29 par les candidats pour $r_{\mathfrak{B}}$ et $r_{\mathfrak{G}}$ des intervalles $[-\frac{M_{\mathfrak{B}}}{2} - S_{\mathfrak{G}}\frac{M_{\mathfrak{B}}}{M_{\mathfrak{G}}}, \frac{M_{\mathfrak{B}}}{2} - S_{\mathfrak{G}}\frac{M_{\mathfrak{B}}}{M_{\mathfrak{G}}}]$ et $[0, M_{\mathfrak{G}}]$ obtenus avec la proposition 30. Ainsi, à l'étape des pas de bébés et de géants, nous n'avons plus qu'à vérifier que la valeur de hachage de $\pm\gamma_s P_{\mathfrak{G}}$ correspond à celle d'une valeur de la table des pas de bébés. Comme obtenir $-\gamma_s P_{\mathfrak{G}}$ est immédiat à partir de $\gamma_s P_{\mathfrak{G}}$, nous faisons au total l'économie des $C_{\mathfrak{G}}/2$ additions de P_q.
- Nous avons remarqué en pratique que le nombre des différences $\beta_{r+1}-\beta_r$ ou $\gamma_{s+1}-\gamma_s$ est négligeable vis-à-vis de $C_{\mathfrak{B}}$ ou $C_{\mathfrak{G}}$. Nous calculons donc ces différences, puis, après un tri, enlevons les duplicats. Ainsi, pour $C_{\mathfrak{B}} \simeq 10^5$, seules quelques centaines de ces différences subsistent. Dans les étapes des pas de bébés et de géants, nous remplaçons donc le précalcul des points

$$0P_{\mathfrak{B}}, \ldots, (B-1)P_{\mathfrak{B}}\ ; BP_{\mathfrak{B}}, \ldots, (B-1)BP_{\mathfrak{B}}\ ; \ldots\ ; B^iP_{\mathfrak{B}}, \ldots, (B-1)B^iP_{\mathfrak{B}}$$

  et

$$0P_{\mathfrak{G}}, \ldots, (B-1)P_{\mathfrak{G}}\ ; BP_{\mathfrak{G}}, \ldots, (B-1)BP_{\mathfrak{G}}\ ; \ldots\ ; B^jP_{\mathfrak{G}}, \ldots, (B-1)B^jP_{\mathfrak{G}}$$

  par celui des différences $(\beta_{r+1}-\beta_r)P_{\mathfrak{B}}$ et celui des $(\gamma_{s+1}-\gamma_s)P_{\mathfrak{G}}$.
- Pour réduire encore le coût, nous effectuons plusieurs additions "simultanément" grâce une astuce calculatoire déjà utilisée avec succès par l'auteur pour factoriser des entiers [65]. En effet, comme nous pouvons le remarquer dans les formules du chapitre 2, le coût majeur lors de l'addition de deux points sur une courbe elliptique est le calcul de l'inverse d'un élément de $\mathbb{F}_q$. Or il est possible de remplacer le coût de n inversions par celui d'une inversion et $3(n-1)$ multiplications. Par exemple, pour calculer les inverses de trois éléments x, y et z de $\mathbb{F}_q$, nous calculons d'abord xy et xyz avec 2 multiplications. Ensuite $(xyz)^{-1}$ avec une inversion. Nous obtenons alors z^{-1} par le produit de xy avec $(xyz)^{-1}$. Pour x^{-1} et y^{-1}, il ne reste plus qu'à multiplier $(xyz)^{-1}$ par z pour obtenir $(xy)^{-1}$ que nous multiplions ensuite par x et y. Le cas général, est une généralisation immédiate de ce petit exemple et pour en tirer parti dans le tri-recherche, nous effectuons simplement les nombreuses additions que requiert cet algorithme par groupes de 100.

Complexité

Avec ces optimisations, le coût de l'étape des pas de bébés est de $C_{\mathfrak{B}}$ additions, et en moyenne, celui des pas de géants, $C_{\mathfrak{G}}/2$ additions. Nous choisissons donc $\mathfrak{B}$ et $\mathfrak{G}$ tels que $C_{\mathfrak{B}} \simeq \frac{1}{2}C_{\mathfrak{G}}$. Nous avons alors un coût total de $\sqrt{2C} \simeq 1.4\sqrt{C}$ additions sur la courbe.

Le gain obtenu par l'addition simultanée de points est lui plus difficile à chiffrer puisque dépendant du corps fini $\mathbb{F}_q$. Pour $\mathbb{Z}/p\mathbb{Z}$, une étude menée par l'auteur [65] montre que le gain est supérieur à 4. Pour $\mathbb{F}_q$, gageons sans risque que nous divisons le temps total par un facteur au moins égal à deux.

Résultats

Nous avons programmé pour tout corps fini $\mathbb{F}_q$ cette variante de l'algorithme d'Atkin en langage `C` à l'aide de la librairie `ZEN` (cf. chapitre 10). Même si l'objectif initial en est la recombinaison des données intermédiaires obtenues avec les algorithmes des chapitres 3 et 11, et même si probablement des algorithmes *ad hoc* du même type [27, Algo. 7.4.12] sont plus efficaces en pratique, il nous a semblé intéressant de calculer la cardinalité de courbes elliptiques avec uniquement cet algorithme par la donnée de ℓ candidats $0, 1, \ldots, \ell-1$ pour c avec un nombre suffisant de nombres premiers ℓ.

Pour savoir dans ce cadre jusqu'à quelle taille de corps fini l'utilisation seule de cet algorithme suffit, nous avons mesuré dans le tableau 11.2 le temps nécessaire à un tel calcul dans des corps à environ 10^5, 10^{10}, ..., 10^{30} éléments pour les courbes

- $E\ : Y^2+XY = X^3+a$ avec $a = t^{16}+t^{14}+t^{13}+t^9+t^8+t^7+t^6+t^5+t^4+t^3$ dans $\mathbb{F}_{2^n} = \mathbb{F}_2[t]/(X^n+f(t))$ où $f(t)$ est le plus petit polynôme à coefficients dans $\mathbb{F}_2$ tel que $X^n + f(t)$ soit irréductible.
- $E\ : Y^2 = X^3 + 4589X + 91128$ dans $\mathbb{Z}/p\mathbb{Z}$.

Corps finis taille	corps	Temps	C
10^5	$\mathbb{F}_{2^{17}}$	0 s	$2{\cdot}10^3$
	$\mathbb{F}_{10^5+3}$	0 s	
10^{10}	$\mathbb{F}_{2^{30}}$	0 s	$2{\cdot}10^5$
	$\mathbb{F}_{10^9+7}$	0 s	
10^{15}	$\mathbb{F}_{2^{46}}$	2 s	$4{\cdot}10^7$
	$\mathbb{F}_{10^{14}+31}$	1 s	

Corps finis taille	corps	Temps	C
10^{20}	$\mathbb{F}_{2^{64}}$	37 s	$3{\cdot}10^{10}$
	$\mathbb{F}_{10^{19}+51}$	9 s	
10^{25}	$\mathbb{F}_{2^{80}}$	13 mn 7 s	$4{\cdot}10^{12}$
	$\mathbb{F}_{10^{24}+7}$	5 mn 20 s	
10^{30}	$\mathbb{F}_{2^{96}}$	9 h 25 mn	10^{15}
	$\mathbb{F}_{10^{29}+319}$	5 h 3 mn	

FIG. 11.2 – Cardinalité d'une courbe avec un algorithme de tri-recherche (station DEC).

Chapitre 12

Programmation de l'algorithme SEA et résultats

Le programme SEA (Schoof, Elkies, Atkin) est un logiciel écrit en langage C dont le but est le calcul de la cardinalité d'une courbe elliptique E donnée par une équation de Weierstrass, $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$, sur un corps fini $\mathbb{F}_q$ défini par une tour d'extensions algébriques de $\mathbb{F}_p$. Il est utilisable pour toute caractéristique p tant que q est inférieur à 10^{100}. Pour le cas particulier des corps de caractéristique deux et de grande caractéristique (disons $p > 1000$), des corps bien plus gros sont envisageables. Nous avons ainsi pu calculer la cardinalité d'une courbe elliptique définie dans $\mathbb{F}_{2^{1301}}$.

À cet effet, le programme SEA[1] tire parti des routines de la librairie ZEN (cf. chapitre 10) pour mettre en œuvre en toute généralité les algorithmes des parties I et II. Une fois explicités les choix algorithmiques que nous avons été amenés à faire, nous expliquons quelle stratégie nous appliquons pour mettre en œuvre *automatiquement* ces algorithmes tout en essayant de minimiser le temps de calcul total (section 12.1). Nous donnons ensuite des temps précis obtenus avec notre implantation pour des corps de taille moyenne avant de détailler les résultats que nous avons obtenus pour des corps de taille plus grande (section 12.2). Enfin, nous expliquons comment il est possible d'utiliser cette implantation pour rechercher efficacement des courbes elliptiques à nombre de points premier (section 12.3).

12.1 Implantation

Si comme nous le rappelons tout d'abord, le choix des algorithmes à utiliser dans une implantation est clair à ce stade de la lecture, encore faut-il les mettre en œuvre efficacement.

Du point de vue algorithmique, nous avons pour cela expérimenté deux stratégies. Dans une première stratégie dite "statique", nous contrôlons le comportement de l'algorithme à l'aide de paramètres et regardons leurs influences sur le temps total. Pour un corps fini fixé, il est ainsi possible de les régler sur quelques courbes pour obtenir ensuite d'excellents résultats en moyenne. Le désavantage de cette méthode est que ce réglage est rendu difficile pour de grands corps finis puisqu'il nécessite le calcul préalable de nombreuses cardinalités. Nous préférons pour notre part appliquer une stratégie dite "dynamique". Elle consiste à découper le calcul en tâches auxquelles nous associons un coût. Le programme exécute alors les tâches par ordre de coûts croissants.

Du point de vue de la programmation, nous avons tout écrit en langage C à l'aide de la librairie ZEN (cf. chapitre 10). Nous sommes ainsi en mesure de traiter tout corps fini défini par la donnée d'une tour d'extensions algébriques. Nous renvoyons le lecteur à l'annexe A pour un exemple d'exécution du programme obtenu.

[1] Ces travaux ont fait l'objet de deux publications [70, 68].

12.1.1 Algorithmes utilisés

Le corps du programme `SEA` est composé des algorithmes des chapitres 3 et 11 à l'exception de l'algorithme de Dewaghe pour trouver l'entier θ tel que $\phi_E^2 + [q]_E = [\theta]_E \circ \phi_E$ puisque cet algorithme est finalement d'un usage assez marginal en pratique.

Précisément, nous découpons en tâches l'algorithme décrit à la fin du chapitre 3 pour déterminer $c \bmod \ell^k$ (ℓ, un nombre premier). Ainsi, pour un ℓ fixé, la première tâche est le calcul de $X^q \bmod \Phi_\ell(X, j_E)$ pour déterminer à quel cas du théorème 35 le nombre premier ℓ correspond.

Dans les cas (i) et (ii), nous utilisons ensuite l'algorithme d'Elkies complété par les cycles d'isogénies de Couveignes, Dewaghe et Morain. Schématiquement, nous calculons une isogénie de degré ℓ qui nous sert à définir un point P de ℓ-division dans une extension algébrique de $\mathbb{F}_q$ de degré $\ell-1$. Puis nous recherchons la valeur propre de ϕ_E associée au "vecteur propre" P, c'est-à-dire l'entier κ tel que $\phi_E(P) = \kappa P$ puisqu'alors $c = \kappa + q/\kappa \bmod \ell$. Il est ensuite possible de réitérer le procédé pour ℓ^k avec $k = 2,\ldots$

Dans le cas (iii), nous déterminons tout d'abord l'entier r du théorème 35 à partir duquel nous sommes en mesure d'avoir plusieurs candidats pour $c \bmod \ell$. Ensuite, nous pouvons éventuellement construire une isogénie de degré ℓ vers une courbe définie sur $\mathbb{F}_{q^r}$ qui nous sert à construire un point P de ℓ-torsion sur une extension algébrique de degré $r(\ell-1)$ de $\mathbb{F}_q$. Comme pour l'algorithme original de Schoof, nous recherchons ensuite l'entier θ tel que $\phi_E^2(P) + qP = \theta\phi_E(P)$. Il est ensuite possible de réitérer le procédé pour ℓ^k avec $k = 2,\ldots$ Notons que si $r = \ell+1$ comme cela arrive relativement souvent, plutôt que d'obtenir une isogénie de degré ℓ sur $\mathbb{F}_{q^{\ell+1}}$, il est finalement plus rapide d'utiliser directement l'algorithme original de Schoof. Cette organisation en tâches est schématisée sur la figure 12.1.

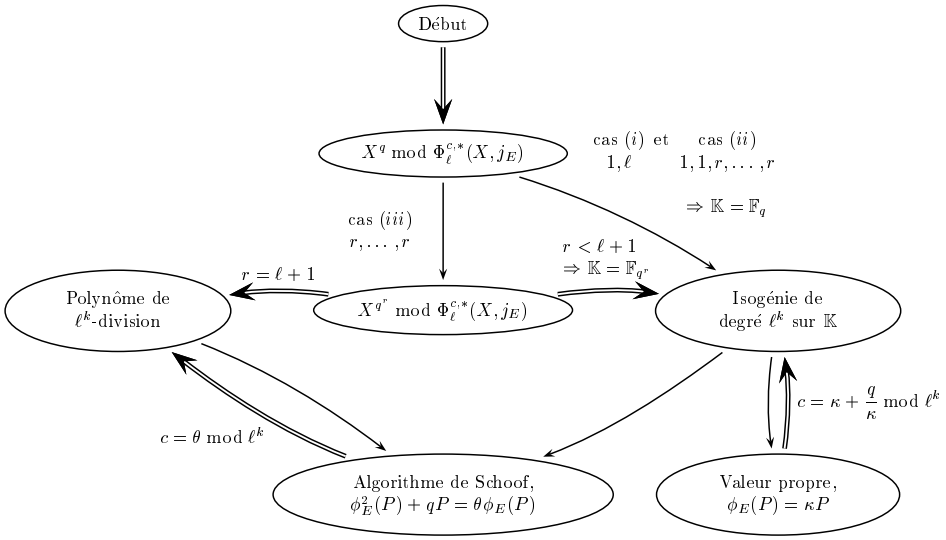


FIG. 12.1 – Division en tâches du calcul de $c \bmod \ell^k$.

L'algorithme de calcul d'isogénie utilisé dépend quant à lui de la caractéristique p du corps et du degré ℓ de l'isogénie. Pour les raisons invoquées dans la partie II, nous utilisons en fait l'algorithme `MULTBY3` pour tout ℓ quand $p = 2$ (chapitre 7), l'algorithme `CCR` quand $\ell < p$ (chapitre 4), la combinaison d'algorithme du chapitre 8 quand $\ell > p$ avec un appel au premier algorithme de Couveignes quand de plus $\ell > 2p$ (chapitre 5).

Enfin, pour les courbes particulières de la section 2.2 du chapitre 2, nous avons mis en place des procédures spécifiques qui nous permettent d'obtenir la cardinalité de telles courbes rapidement.

12.1.2 Stratégie statique

Comme les calculs dans des extensions algébriques sont d'autant plus coûteux que le degré de ces extensions est élevé, il est naturel de borner dans l'algorithme ces degrés par des constantes fixées par l'utilisateur. Nous introduisons ainsi :

- $\mathcal{E}$, le degré maximal d'une extension sur laquelle est définie un point de ℓ-torsion pour les cas (i) et (ii) du théorème 35,
- $\mathcal{C}$, le degré maximal d'une extension sur laquelle est définie un point de ℓ^k-torsion pour les cas (ii),
- $\mathcal{A}$, le degré maximal en dessous duquel nous recherchons l'entier r dans le cas (iii),
- $\mathcal{M}$, le nombre maximal de candidats pour c avant d'appeler la routine de "tri-recherche" du chapitre 3.

procedure `SEA`$(\mathbb{F}_{p^n}, E, \mathcal{E}, \mathcal{C}, \mathcal{A}, \mathcal{M})$
$\ell := 1$; $M_u := 1$; $M_l := 1$; $M := 1$
while $M_u \times M_l < 4\sqrt{p^n}$ **or** $M > \mathcal{M}$ **do**
  $\ell :=$ nextprime(ℓ);
  Trouver le nombre ν de racines de $\Phi_\ell^{c,*}(X, j_E)$ dans $\mathbb{F}_{p^n}$.
  if $\nu = 2$ **and** $(\ell-1)/2 \leqslant \mathcal{E}$ **then**
    `ElkiesCase`(ℓ); Calculer d, le semi ordre de la valeur propre κ modulo ℓ.
    for $k := 2$ **while** $\ell^{k-1}d \leqslant \mathcal{C}$ **do** Calculer c mod ℓ^k; $M_u := M_u \times \ell$ **endfor**
  else
    if $(\ell^2-1)/2 \leqslant \mathcal{S}$ **then**
      for $k := 1$ **while** $(\ell^{2k} - \ell^{2k-2})/2 \leqslant \mathcal{S}$ **do**
        Calculer c mod ℓ^k avec l'algorithme original de Schoof; $M_u := M_u \times \ell$
      endfor
    else if $\ell \leqslant \mathcal{A}$ **then** `AtkinCase`(ℓ); $M_l := M_l \times \ell$ **endif endif**
  endif
endfor
Utiliser un algorithme de tri-recherche pour finir les calculs.
endprocedure

FIG. 12.2 – Version simplifiée de l'algorithme `SEA` paramétré par $\mathcal{E}$, $\mathcal{C}$, $\mathcal{A}$ et $\mathcal{M}$.

Ainsi, l'algorithme 12.2 est une version simplifiée de l'algorithme `SEA` qui se sert de ces paramètres ($\ell \neq 2$). Nous y utilisons deux variables, M_u et M_l, qui contiennent respectivement le produit de nombres premiers ℓ pour lesquels c mod ℓ est connu de façon unique (nombres de type U) et ceux pour lesquels nous avons plusieurs candidats pour c mod ℓ (nombres de type L). Typiquement, M_u contient les nombres premiers dits d'Elkies et M_l, ceux dit d'Atkin. La variable M contient le nombre courant de candidats pour c. Enfin, les quantités $\ell^{k-1}d$ et $(\ell^{2k} - \ell^{2k-2})/2$ représentent le degré d'un facteur du polynôme $f_{\ell^k}(X)$ de ℓ^k-division. Le cœur des calculs est alors constitué par les routines `ElkiesCase` et `AtkinCase` de la figure 12.3.

procedure `AtkinCase`(ℓ)
Trouver le plus petit entier r tel que $X^{q^r} \equiv X \bmod \Phi_\ell^{c,*}(X, j_E)$ et poser $c(\ell) = \varphi(r)$.
$M := M \times c(\ell)$; $M_l := M_l \times \ell$

procedure `ElkiesCase`(ℓ)
Calculer un facteur $h_\ell(X)$ de $f_\ell(X)$ de degré $(\ell-1)/2$ par un calcul d'isogénie.
Trouver une valeur propre de ϕ_E relative à h_ℓ, c'est-à-dire, $1 \leq \kappa < \ell$ tel que $(X^q, Y^q) = \kappa(X, Y)$ pour un point (X, Y) de ℓ-torsion, en déduire que $c \equiv \kappa + q/\kappa \bmod \ell$.
$M_u := M_u \times \ell$
endprocedure

FIG. 12.3 – Algorithmes d'Atkin et d'Elkies.

Cette présentation a l'avantage de regrouper plusieurs sous-stratégies possibles. Par exemple, fixer $\mathcal{E} = \mathcal{A} = 0$ redonne l'algorithme orignal de Schoof alors que fixer $\mathcal{E} = 0$ conduit au premier algorithme

d'Atkin. Enfin, l'introduction de $\mathcal{C}$ rend possible l'utilisation des cycles d'isogénies. Il s'avère que, quoique fastidieux, un réglage fin de ces paramètres conduit en pratique à d'excellents résultats (cf. tableau 12.1).

$n=65$ $\mathcal{E}=2,\mathcal{C}=2,\mathcal{S}=0$				$n=89$ $\mathcal{E}=3,\mathcal{C}=4,\mathcal{S}=0$				$n=105$ $\mathcal{E}=3,\mathcal{C}=4,\mathcal{S}=24$			
	min	max	avg		min	max	avg		min	max	avg
$\ell_{\max}$	29	31	29	$\ell_{\max}$	37	41	41	$\ell_{\max}$	43	47	43
$\#U$	0	5	1	$\#U$	1	5	3	$\#U$	1	5	3
$\#L$	5	10	9	$\#L$	7	12	10	$\#L$	9	13	11
$\#M$	$5{\cdot}10^2$	$8{\cdot}10^5$	$4{\cdot}10^4$	$\#M$	10^4	$9{\cdot}10^6$	10^6	$\#M$	$5{\cdot}10^4$	10^9	10^8
X^q	2.2	3.5	2.8	X^q	4.1	7.3	5.9	X^q	7.5	10.6	8.8
X^{q^r}	0.2	0.7	0.4	X^{q^r}	0.9	2.5	1.7	X^{q^r}	2.2	3.9	3
Schoof	0	0	0	Schoof	0	0.5	0.2	Schoof	0	0.6	0.2
h	0	0.1	0	h	0	0.2	0.1	h	0	0.3	0.1
κ	0.1	1.1	0.5	κ	0.6	1.8	1.1	κ	0.5	2.3	1.3
$\mathrm{T}-\mathrm{R}$	0.5	1.1	0.7	$\mathrm{T}-\mathrm{R}$	0.9	2.4	1.5	$\mathrm{T}-\mathrm{R}$	1.7	28.2	5.8
Total	3.7	6	4.5	Total	7.6	13	10.5	Total	14	41.6	19.1

TAB. 12.1 – Statistiques avec une stratégie statique pour de petits corps $\mathbb{F}_{2^n}$.

Nous avons mesuré le temps moyen d'exécution pour 50 courbes aléatoires $Y^2+XY=X^3+a$ définies sur $\mathbb{F}_{2^{65}} \simeq \mathbb{F}_2[t]/(t^{65}+t^4+t^3+t+1)$, $\mathbb{F}_{2^{89}} \simeq \mathbb{F}_2[t]/(t^{89}+t^6+t^5+t^3+1)$ et $\mathbb{F}_{2^{105}} \simeq \mathbb{F}_2[t]/(t^{105}+t^4+1)$. Nous donnons : ℓ_{max}, le plus grand nombre premier utilisé ; le nombre de nombres premiers de type U (resp. L) ; M, le nombre de candidats pour $c \bmod \ell$; le temps cumulé pour X^q, X^{q^r} et l'algorithme original de Schoof ; le temps pour calculer les isogénies de degré ℓ (h), le temps pour trouver la valeur propre κ associée à l'isogénie (κ) et enfin le temps pris par le "tri-recherche" ($T-R$). Pour chaque corps fini, nous donnons de plus temps minimaux, maximaux et moyens en secondes.

12.1.3 Stratégie dynamique

Le gros désavantage de la stratégie statique est l'obligation de fixer au préalable les paramètres. Pour des corps plus gros que ceux traités précédemment, cela devient difficile et nous préférons pour notre part utiliser une stratégie tout autre.

À chaque tâche de la figure 12.1, nous associons un coût qui est le degré de l'extension algébrique dans laquelle ont lieu les calculs. Notre politique est alors d'effectuer en priorité la tâche de coût le plus bas. Par exemple au tout début du programme, nous devons calculer $X^q \bmod \Phi_\ell(X, j_E)$. Pour un ℓ donné, le coût de cette tâche est $\ell+1$. Le programme va donc réaliser ce calcul pour $\ell=2$. Si, par exemple, 2 correspond au cas (iii) du théorème 35, nous calculons $c \bmod 2$ avec l'algorithme de Schoof car le coût de cette phase est 1, ce qui est moindre que de calculer $X^q \bmod \Phi_\ell(X, j_E)$ pour $\ell=3$ qui est de coût 4. Par contre nous n'effectuons le calcul de $c \bmod 2^2$ qu'une fois traité $\ell=3$ puisque le coût d'un algorithme de Schoof appliqué à 2^2 est 6.

Ainsi, pour un ℓ donné, le calcul de $c \bmod \ell^k$ dépend fortement de k (les transitions qui font croître le coût sont représentées par une double flèche sur le schéma de la figure 12.1). C'est pourquoi, le programme `SEA` préférera calculer c modulo un nombre premier immédiatement supérieur à ℓ plutôt que de calculer $c \bmod \ell^2$. Par contre, il est fort possible que bien plus tard dans l'exécution, ℓ^2 soit traité, par ce qu'alors ce sera la tâche de coût minimal.

Notons que l'implantation de cette stratégie nécessite le stockage des données intermédiaires afin d'être en mesure, pour un ℓ donné, de reprendre les calculs. Néanmoins, ce coût est négligeable en pratique.

Nous avons effectué des statistiques identiques à celles de la section précédente avec cette stratégie. Les résultats sont donnés sur le tableau 12.2, ils sont proches de ceux obtenus avec une stratégie statique.

$\mathbb{F}_{2^{65}}$	min	max	avg
$\ell_{\max}$	31	31	31
$\#U$	0	5	1
$\#L$	6	11	10
$\#M$	10^3	$3{\cdot}10^6$	$2{\cdot}10^5$
X^q	2.2	4.2	3.3
X^{q^r}	0, 3	0.9	0.6
Schoof	0	0	0
h	0	0	0
κ	0	0	0
T − R	0.6	1.5	1
Total	3.8	5.9	4.9

$\mathbb{F}_{2^{89}}$	min	max	avg
$\ell_{\max}$	41	41	41
$\#U$	0	4	2
$\#L$	9	13	11
$\#M$	$6{\cdot}10^3$	$8{\cdot}10^7$	$6{\cdot}10^6$
X^q	5.1	7.8	6.4
X^{q^r}	0.8	2.6	1.9
Schoof	0	0.4	0
h	0	0.2	0
κ	0.2	0.8	0.6
T − R	1.1	5.2	2.2
Total	9.2	14.6	11.2

$\mathbb{F}_{2^{105}}$	min	max	avg
$\ell_{\max}$	41	47	42
$\#U$	1	6	3
$\#L$	8	13	10
$\#M$	$5{\cdot}10^3$	$2{\cdot}10^8$	10^7
X^q	6.6	11.5	8.8
X^{q^r}	1	3.8	2.6
Schoof	0	3.8	0.3
h	0	1.7	0.5
κ	0.8	3.8	2.4
T − R	1.1	9.8	2.9
Total	13.4	24.5	17.3

TAB. 12.2 – Statistiques avec une stratégie dynamique pour de petits corps $\mathbb{F}_{2^n}$.

12.2 Résultats

Les performances du programme `SEA` dépendent du corps fini de définition. C'est pourquoi, nous donnons dans une première partie des temps précis pour des corps finis de taille moyenne ($q < 10^{100}$) et de caractéristique variée. Puis, comme nous avons optimisé notre programme pour des corps finis de caractéristique deux, nous donnons dans une deuxième partie les résultats que nous avons obtenus avec ce programme pour de grands corps de cette famille.

12.2.1 Corps de taille moyenne

Puisque les corps finis $\mathbb{F}_{2^n}$ avec $n \simeq 200$ sont de première importance pour les applications cryptographiques, nous avons réalisé la même étude que précédemment pour trois corps finis de ce type ; c'est-à-dire 50 courbes elliptiques dans $\mathbb{F}_{2^{155}} \simeq \mathbb{F}_2[t]/(t^{155}+t^7+t^5+t^4+1)$, $\mathbb{F}_{2^{196}} \simeq \mathbb{F}_2[t]/(t^{196}+t^3+1)$ et $\mathbb{F}_{2^{300}} \simeq \mathbb{F}_2[t]/(t^{300}+t^5+1)$. Les résultats sont donnés sur le tableau 12.3 en secondes.

$\mathbb{F}_{2^{155}}$	min	max	avg
$\ell_{\max}$	59	71	60
$\#U$	4	11	7
$\#L$	7	15	10
$\#M$	$3{\cdot}10^4$	$7{\cdot}10^8$	$7{\cdot}10^7$
X^q	30.4	56.1	40.6
X^{q^r}	4.4	13.9	7.8
Schoof	0	14.8	4.3
h	1.5	21.6	7.1
κ	7.4	31.9	20.1
T − R	2.9	20.4	6.5
Total	58.8	132	86.5

$\mathbb{F}_{2^{196}}$	min	max	avg
$\ell_{\max}$	73	79	74
$\#U$	7	13	10
$\#L$	8	15	11
$\#M$	10^5	$7{\cdot}10^9$	$8{\cdot}10^8$
X^q	113	475	147
X^{q^r}	8.8	31.8	21.6
Schoof	0	55.5	17.9
h	9.9	419	40.6
κ	29.2	90.1	58.9
T − R	5	86.9	22.9
Total	212	1029	308

$\mathbb{F}_{2^{300}}$	min	max	avg
$\ell_{\max}$	9	15	11
$\#U$	11	19	16
$\#L$	9	20	14
$\#M$	$5{\cdot}10^6$	$5{\cdot}10^{11}$	$5{\cdot}10^{10}$
X^q	744	1761	996
X^{q^r}	46	387	119
Schoof	0	551	199
h	76	568	287
κ	354	961	601
T − R	14	1510	230
Total	1519	3686	2434

TAB. 12.3 – Statistiques pour des corps $\mathbb{F}_{2^n}$ de taille moyenne (stratégie dynamique).

Ainsi, s'il y a seulement cinq ans, il fallait plusieurs heures à Menezes, Vanstonne et Zuccherato pour obtenir la cardinalité d'une courbe définie sur $\mathbb{F}_{2^{196}}$ [80], seules quelques minutes sont aujourd'hui nécessaires.

Jusqu'à présent, nous nous sommes uniquement préoccupés de corps finis de caractéristique deux. Pour étudier l'influence de la caractéristique, nous avons d'autre part calculé la cardinalité d'une courbe elliptique $E : Y^2 + a_1XY = X^3 + a_4X + a_6$ dans six corps finis $\mathbb{F}_{p^n} \simeq \mathbb{F}_p[t]/(P(t))$ où p est un nombre premier dont le logarithme en base deux est sensiblement égal à 2, 8, 16, 64, 128 et 266 et où $P(t)$ est

un polynôme irréductible de degré le plus petit entier n tel que $p^n > 10^{80}$. Les résultats sont donnés en secondes dans le tableau 12.4 (où $\overline{91128} = t^{16} + t^{14} + t^{13} + t^9 + t^8 + t^7 + t^6 + t^5 + t^4 + t^3$).

p	$10^{80} + 129$	$2^{128} - 159$	$2^{64} - 59$	$2^{16} - 15$	251	2
$P(t)$	–	$t^3 + 2$	$t^5 + t + 4$	$t^{17} + t$ $+33892$	$t^{34} + t + 52$	$t^{266} + t^6 +$ $t^3 + t^2 + 1$
a_1	0	1	1	1	1	1
a_4	4589	$4589t + 4792$	$4589t + 4792$	4589	$18t + 71$	0
a_6	91128	$91128(t + 1)$	$91128(t + 1)$	$t + 25607$	$t^2 + 112t + 15$	$\overline{91128}$
$\ell_{\max}$	131	167	139	107	107	103
$\#U$	14	19	20	14	12	13
$\#L$	18	20	13	13	15	14
$\#M$	$4{\cdot}10^{10}$	$3{\cdot}10^{11}$	$7{\cdot}10^{10}$	10^9	$9{\cdot}10^{10}$	$6{\cdot}10^9$
X^q	1645	10248	2330	6025	12759	556
X^{q^r}	363	1953	299	817	1421	106
Schoof	204	841	81	2668	9044	43
h	49	514	119	286	505	91
κ	154	3066	505	1181	2258	242
T – R	55	154	160	168	878	139
Total	2481	17437	3500	11148	26867	1179

TAB. 12.4 – Temps pour des corps finis à environ 10^{80} éléments (stratégie dynamique).

De nombreuses fluctuations sont notables. Nous y voyons deux causes. D'une part, l'influence de l'arithmétique utilisée est importante (cf. la figure 10.1). D'autre part, si effectivement nous avons cherché à avoir $p^n \simeq 10^{80}$, il n'en demeure pas moins vrai que certain des corps finis utilisés ont bien plus que 10^{80}éléments et sont donc pénalisés (par exemple $3 \log_{10}(2^{128} - 159) \simeq 115$).

12.2.2 Corps de grande taille

Notre expérience des corps finis de grande taille a commencé avec ceux de grande caractéristique, puisque, suite à l'utilisation de routines de multiplications polynomiales basées sur des algorithmes de transformées de Fourier discrètes et développées par l'auteur [65], Morain a été en mesure de déterminer que la courbe

$$Y^2 = X^3 + 4589X + 91128$$

définie dans $\mathbb{F}_{10^{499}+153}$ a $10^{499} + 153 + 1 - c$ points avec

$$\begin{aligned} c = -&55317125053606569162976530203184874598723974032546865680659963170\\ &11274137992945744442611560616258650142223797293405315503589938880\\ &32237207379679849162325347608624510817409606791818935212167258043\\ &61067332068304349539659492265105944069081498646941789 69 \end{aligned}$$

(87 jours sur une station DEC à 266 MHz).

Depuis, nous n'avons eu de cesse que de calculer la cardinalité de la courbe elliptique E_X définie par

$$E_X : Y^2 + XY = X^3 + t^{16} + t^{14} + t^{13} + t^9 + t^8 + t^7 + t^6 + t^5 + t^4 + t^3.$$

sur des corps $\mathbb{F}_{2^n}$ avec n aussi grand que possible. Nous en rappelons dans le tableau 12.5 la chronologie. Bien sûr, sur une période aussi longue, notre implantation a subi de nombreux remaniements avant d'aboutir au programme que nous décrivons dans ce document. C'est pourquoi, nous donnons dans le tableau, pour chaque exposant n, non seulement la date et les temps (ramenés à la DEC 266 MHz qui

nous sert de référence) correspondants à la première fois où nous avons été en mesure d'effectuer le calcul, mais aussi les temps qui sont nécessaires pour le réaliser avec notre implantation actuelle (notons que cette implantation dispose d'une arithmétique polynomiale pour $\mathbb{F}_2$ deux fois plus rapide que pour nos précédents programmes). De plus, comme nos premières implantations souffraient d'un calcul d'isogénies peu performant, nous donnons pour ces deux implantations le plus grand nombre premier ℓ utilisé ($\ell_{\max}$) et le temps cumulé des calculs d'isogénies.

n	Vieux programme				Programme actuel		
	Date	$\ell_{\max}$	Temps isogénies	Temps total	$\ell_{\max}$	Temps isogénies	Temps total
300	30/09/94	109	3j 22h	9j 4h	109	3mn 19s	47mn 42s
400	18/10/94	173	4j 23h	14j 13h	157	4mn 20s	2h 7mn
500	9/11/94	179	3j	6j 11h	193	6mn 2s	8h
601	5/1/95	271	20j 10h	30j 3h	317	50mn 50s	1j 6h
701	13/3/95	337	28j	45j 17h	359	2h 37mn	3j 4h
1009	29/8/95	577	78j 21h	121j 15h	547	1j 2h	19j 11h
1301	15/4/96	673	3j 12h	103j 37h	673	3j 12h	51j 18h

TAB. 12.5 – Temps pour de grands corps finis de caractéristique deux.

Comme à la date d'aujourd'hui, nous sommes probablement les seuls à avoir été en mesure de calculer la cardinalité d'une courbe elliptique dans un corps fini de caractéristique deux aussi large que $\mathbb{F}_{2^{1301}}$, nous détaillons ce calcul sur la figure 12.4.

Soient

$$\mathbb{F}_{2^{1301}} \simeq \mathbb{F}_2[t]/(t^{1301} + t^{11} + t^{10} + t + 1)$$

et

$$a_6 = t^{16} + t^{14} + t^{13} + t^9 + t^8 + t^7 + t^6 + t^5 + t^4 + t^3.$$

Alors le nombre de points de la courbe elliptique

$$E_X \ : Y^2 + XY = X^3 + a_6$$

est égal à

$$2^{1301} + 1 - c$$

où

$$\begin{aligned} c = -&1252020466045855099434238134741436458098159760255154193747525 \\ &1045091682467090516587665003925037001223752821603011654792837 \\ &2408257614374530929254301621371725666282688807678760468051211 \\ &77132601520639. \end{aligned}$$

FIG. 12.4 – Cardinalité d'une courbe elliptique définie sur $\mathbb{F}_{2^{1301}}$.

12.3 Application à la cryptographie

Puisque les meilleures attaques contre le problème du logarithme discret sur une courbe elliptique sont

1. la réduction par couplage de Weil pour des courbes supersingulières (attaque [77]),
2. les algorithmes de "pas de bébé, pas de géant", Pollard-ρ et Pohlig Helman pour les autres courbes,

de “bonnes courbes” pour des applications cryptographiques doivent, d'une part, être définies dans un corps fini $\mathbb{F}_q$ suffisamment grand et être de cardinalité “presque première” (pour éviter le point 2.) et, d'autre part, de cardinalité différente de q, $q+1\pm\sqrt{q}$, $q+1\pm\sqrt{2q}$, $q+1\pm\sqrt{3q}$ et $q+1\pm 2\sqrt{q}$ (pour éviter le point 1.).

Nous décrivons dans la section 12.3.1 une stratégie de type “ early abort” suggérée par Morain pour tirer parti de l'algorithme `SEA` afin de détecter rapidement la plupart des courbes qui ne satisfont pas cette condition. Par soucis de simplification, nous ne l'expliquons que pour des courbes elliptiques E_a : $Y^2+XY=X^3+a$ définies sur $\mathbb{F}_{2^n}$, mais cette stratégie est bien sûr valide pour tout corps fini. Nous donnons ensuite dans la section 12.3.2 des temps et des exemples de “bonnes courbes” calculées avec cette stratégie.

12.3.1 Stratégie “Early abort”

Algorithme

Une courbe elliptique E_a : $Y^2+XY=X^3+a$ est non supersingulière et a toujours un point $Q_a=(\sqrt[4]{a},\sqrt{a})$ d'ordre 4. Ainsi, la condition précédente peut être reformulée comme suit, “une bonne courbe elliptique E_a est une courbe définie sur $\mathbb{F}_{2^n}$ avec $n\geqslant 60$ dont la cardinalité est quatre fois un nombre premier”.

Pour trouver de telles “bonnes courbes”, nous procédons comme suit :

1. choisir au hasard un élément $a\in\mathbb{F}_{2^n}^*$.
2. calculer $c \bmod \ell$ avec l'algorithme `SEA` en vérifiant durant le calcul pour chaque nombre premier d'Elkies ℓ distinct de 2 que
$$2^n+1-c \bmod \ell \neq 0.$$
Sinon, cela signifie que le nombre de points sur la courbe est divisible par ℓ. Dans ce cas, aller à l'étape 1.
3. vérifier que la cardinalité de la courbe est 4 fois un nombre premier, sinon, aller à l'étape 1.

Tout d'abord, remarquons que chaque fois que $2^n+1-t \bmod \ell = 0$ pour un nombre premier ℓ, cela signifie qu'il y a un point d'ordre ℓ sur E_a, en particulier, il existe une isogénie de degré ℓ définie à partir de E_a et ℓ est nécessairement un nombre premier d'Elkies.

Remarquons aussi qu'il est préférable de tester la primalité de la cardinalité à l'étape 3. par un test de pseudo primalité complété par un test de primalité exact (par exemple le programme ECPP [88]), mais pour des raisons pratiques, nous utilisons uniquement le logiciel MAPLE [25] dans nos expériences.

Remarquons enfin que, comme nous le verrons dans la section 12.3.1, un nombre premier ℓ a d'autant plus de chance de diviser la cardinalité d'une courbe qu'il est petit. Comme nous utilisons des nombres premiers ℓ aussi petits que possible dans l'algorithme `SEA`, nous écartons de mauvaises courbes rapidement, ce qui explique les bonnes performances de cet algorithme.

Analyse

Un théorème de Howe [49] qui étend le travail de Lenstra [64] (voir aussi [58]) décrit le comportement asymptotique de la probabilité qu'une courbe elliptique aléatoire définie sur un corps fini $\mathbb{F}_q$ ait ℓ^k ($k\in\mathbb{N}^*$) qui divise le nombre N de ces points quand $q\to\infty$.

Théorème 55. *Il existe une constante $C\leqslant 1/12+5\sqrt{2}/6\simeq 1.262$ telle que l'énoncé suivant soit vrai. Étant donnée une puissance q d'un nombre premier, soit r la fonction arithmétique multiplicative définie pour tout nombre premier ℓ et tout entier strictement positif k par*

$$r_q(\ell^k)=\begin{cases}\dfrac{1}{\ell^{k-1}(\ell-1)} & \text{si } q\neq 1 \bmod \ell^\mu,\\[2ex] \dfrac{\ell^{\nu+1}+\ell^\nu-1}{\ell^{\nu+\mu-1}(\ell^2-1)} & \text{si } q=1 \bmod \ell^\mu,\end{cases}$$

où $\mu = \lceil k/2 \rceil$ *et* $\nu = \lfloor k/2 \rfloor$. *Soit aussi pour tout entier strictement positif* N, $\pi_{q,N}$, *la probabilité qu'une courbe aléatoire* E *sur* $\mathbb{F}_q$ *ait* N *qui divise* $\#E(\mathbb{F}_q)$. *Alors la probabilité* $\pi_{q,N}$ *satisfait*

$$|\pi_{q,N} - r_q(N)| \leqslant \frac{CN\chi(N)2^{\sigma(N)}}{\sqrt{q}},$$

où $\chi(N) = \prod_{\lambda|N}(\lambda+1)/(\lambda-1)$ *et* $\sigma(N)$ *est le nombre des diviseurs premiers de* N.

Soit maintenant $g_q(\ell)$ la probabilité que le plus petit facteur premier de N soit ℓ. Cette probabilité est égale à

$$g_q(\ell) = r_q(\ell) \prod_{\text{nombres premiers } \lambda<\ell} (1 - r_q(\lambda)).$$

En ce qui nous concerne, nous testons des courbes aléatoires E_a non supersingulières définies sur $\mathbb{F}_{2^n}$ avec des cardinalités toujours divisibles par 4. Nous faisons donc l'hypothèse supplémentaire que le théorème de Howe s'applique, excepté pour $\ell = 2$. Les probabilités $r_{2^n}(\ell^k)$ deviennent alors

$$\rho_n(\ell^k) = \begin{cases} 1 & \text{pour } \ell = 2 \text{ et } k = 1, \\ \frac{1}{2^{k-2}} & \text{pour } \ell = 2 \text{ et } k > 1, \\ r_{2^n}(\ell^k) & \text{pour } \ell > 2. \end{cases}$$

Par conséquent, la probabilité $\gamma_n(\ell)$ que nous détections à l'étape 2. de l'algorithme qu'un nombre premier ℓ impair divise la cardinalité de E_a est égale à

$$\gamma_n(\ell) = \rho_n(\ell)(1 - \rho_n(2^3)) \prod_{\text{nombres premiers impairs } \lambda<\ell} (1 - \rho_n(\lambda)).$$

Cette quantité peut être facilement calculée pour n fixé, mais pour tout n, nous pouvons seulement affirmer que

$$\rho_n(2^3) = \frac{1}{2} \text{ et } \frac{\ell}{\ell^2 - 1} \leqslant \rho_n(\ell) \leqslant \frac{1}{\ell - 1},$$

et donc $3/16 \leqslant \gamma_n(3) \leqslant 1/4$, $5/96 \leqslant \gamma_n(5) \leqslant 5/64$, $7/256 \leqslant \gamma_n(7) \leqslant 95/2304\ldots$

12.3.2 Résultats

Nous avons été capables de calculer un grand nombre de telles "bonnes courbes" définies sur $\mathbb{F}_{2^{65}}$, $\mathbb{F}_{2^{89}}$, $\mathbb{F}_{2^{105}}$, $\mathbb{F}_{2^{155}}$ et $\mathbb{F}_{2^{196}}$ en un temps raisonnable. Des statistiques sont données dans le tableau 12.6.

Il apparaît dans ce tableau que les estimations théoriques que nous avons faites dans la section 12.3.1 sont la plupart du temps vérifiées en pratique, excepté peut-être pour des cardinalités divisibles par 5 dans $\mathbb{F}_{2^{196}}$ (150 au lieu de $1000\frac{25}{394} \simeq 65$). Dans tous les cas, la probabilité qu'une courbe elliptique ait son nombre de points divisible par un petit nombre premier ℓ est plutôt élevée et nous avons besoin de "pousser" plus en avant les calculs pour seulement quelques courbes. Certaines de ces "bonnes courbes" sont données dans le tableau 12.7 avec la notation $\overline{\tau_0 + \tau_1 2 + \cdots + \tau_{n-1}2^{n-1}} = \tau_0 + \tau_1 t + \cdots + \tau_{n-1}t^{n-1}$.

	$\mathbb{F}_{2^{65}}$	$\mathbb{F}_{2^{89}}$	$\mathbb{F}_{2^{105}}$	$\mathbb{F}_{2^{155}}$	$\mathbb{F}_{2^{196}}$
# courbes testées	1000	1000	1000	1000	1000
$1000\gamma_n(8)$ # cardinalités divisibles par 8	500 491	500 507	500 500	500 509	500 490
$1000\gamma_n(3)$ # cardinalités divisibles par 3	250 255	250 253	250 256	250 236	187.5 177
$1000\gamma_n(5)$ # cardinalités divisibles par 5	62.5 63	62.5 73	62.5 74	62.5 68	65.1 150
$1000\gamma_n(7)$ # cardinalités divisibles par 7	31.2 28	31.2 28	27.4 59	31.2 25	41.2 34
# cardinalités divisibles par $\ell \geqslant 11$ et détectées à l'étape 2. de l'algorithme	29	52	43	61	57
# cardinalités divisibles par $\ell \geqslant 11$ et détectées à l'étape 3. de l'algorithme	116	77	62	96	90
Nombre de "bonnes courbes"	18	10	6	5	2
Temps total (s)	1277	1733	2231	14112	30254

TAB. 12.6 – Statistiques de la stratégie "early abort".

	a	Cardinalité
$\mathbb{F}_{2^{65}}$	$\overline{2108463510029530717}$	$2^2 \times 9223372038308612213$
$\mathbb{F}_{2^{65}}$	$\overline{15004298573160993787}$	$2^2 \times 9223372035176356667$
$\mathbb{F}_{2^{89}}$	$\overline{362244896591784868971148794}$	$2^2 \times 154742504910673945144969913$
$\mathbb{F}_{2^{89}}$	$\overline{578529593362960704292414 68}$	$2^2 \times 154742504910669983358163303$
$\mathbb{F}_{2^{105}}$	$\overline{65439354054004780257172904324 15}$	$2^2 \times 10141204801825837522175841 2\backslash$ 06867
$\mathbb{F}_{2^{105}}$	$\overline{22959897163766013073560510397 9}\backslash$ $\overline{54}$	$2^2 \times 10141204801825834287526670 3\backslash$ 03267
$\mathbb{F}_{2^{155}}$	$\overline{83879504358878917332366108654 1}\backslash$ $\overline{2790131341725747}$	$2^2 \times 11417981541647679048466281 9\backslash$ 27805319915233345669
$\mathbb{F}_{2^{155}}$	$\overline{11002722068779168584174718059 7}\backslash$ $\overline{77371906785324958}$	$2^2 \times 11417981541647679048466299 2\backslash$ 30130487707830550127
$\mathbb{F}_{2^{196}}$	$\overline{25033470175959423539310828379 4}\backslash$ $\overline{64907961567696239688511281965}$	$2^2 \times 25108406941546723055343157 6\backslash$ $92759220570140916154347737737 983$
$\mathbb{F}_{2^{196}}$	$\overline{40428481881214303633178804345 8}\backslash$ $\overline{37154824320382480200588296980}$	$2^2 \times 25108406941546723055343157 6\backslash$ $92813473113492187155697729606263$

TAB. 12.7 – Courbes avec une cardinalité presque première.

Annexe A

Exemple d'exécution du programme SEA

Le programme SEA est d'une utilisation aisée, puisqu'une fois rentré un corps fini de définition et les cinq coefficients $(a_1, a_2, a_3, a_4, a_6)$ d'une courbe elliptique E, il met en œuvre *automatiquement* les algorithmes des parties I et II nécessaires à l'obtention de la cardinalité de E. Pour s'en convaincre, nous commentons ci-dessous l'exécution de ce programme pour calculer dans $\mathbb{F}_{101}[t]/(t^5+2)$ le nombre de points de la courbe E définie par

$$E\ : Y^2 + XY + t^2Y = X^3 + tX^2 + t^3X + t^4.$$

À l'exécution de SEA, s'affiche tout d'abord la bannière :

```
> sea
sea 1.0 (3/2/1997)
Copyright (C) 1994-1997 Reynald LERCIER.
LIX, Ecole Polytechnique, FRANCE. All rights reserved.
This software comes with NO WARRANTY: see the file MANIFEST for details.

Computation of the cardinality of elliptic curves
        Y^2+a1*X*Y+a3*Y=X^3+a2*X^2+a4*X+a6
defined over any finite field.
```

Puis, le programme demande la caractéristique p du corps fini et initialise $\mathbb{F}_p$.

```
Enter the prime p which defines GF(p) (1 once completed).
101
We have p = 101
% Computing the initialization of R=GF(p)
Tabulation has succeeded...
% the initialization of R=GF(p) computed in 0.02 s
```

Ensuite, il demande, soit le degré de l'extension algébrique sur $\mathbb{F}_p$ souhaitée (auquel cas il calcule un polynôme irréductible de degré spécifié), soit directement un polynôme irréductible. Dans notre cas, nous rentrons $X^5 + 2$.

```
Enter the degree d or the polynomial P(X) that will define the
extension (1 once completed).
(1)*X^5+(2)
We have P(X) = (1)*X^5+(2)
```

Le programme effectue alors quelques précalculs pour déterminer la méthode de multiplication d'éléments de la librairie ZEN la plus rapide pour ce corps.

```
% Computing the initialization of R=R[X]/(P(X))
90 Usual squaring : 0.000000 ms/ops
90 Reciprocal squaring : 0.185185 ms/ops
Usual squaring chosen
89 Usual multiplication : 0.187266 ms/ops
89 Reciprocal multiplication : 0.374532 ms/ops
Usual multiplication chosen
% the initialization of R=R[X]/(P(X)) computed in 0.15 s
```

Le programme réitère alors sa demande pour une éventuelle extension algébrique de $\mathbb{F}_{101^5}$. Ici, ce n'est pas nécessaire, nous rentrons donc 1.

```
Enter the degree d or the polynomial P(X) that will define the
extension (1 once completed).
1
```

Il est ensuite nécessaire de rentrer les cinq coefficients de la courbe E.

```
Reading the parameters of the curve Y^2+a1*X*Y+a3*Y=X^3+a2*X^2+a4*X+a6.
Enter a1.
(1)
We have a1=(1)

Enter a2.
(1)*t
We have a2=(1)*t

Enter a3.
(1)*t^2
We have a3=(1)*t^2

Enter a4.
(1)*t^3
We have a4=(1)*t^3

Enter a6.
(1)*t^4
We have a6=(1)*t^4
```

Pour des raisons d'efficacité, le programme travaille non pas sur la courbe spécifiée, mais sur une courbe isomorphe d'équation l'une des équations canoniques du tableau 2.1 de la page 21.

```
The original elliptic curve is
Y^2 + ((1))*X*Y + ((1)*t^2)*Y =
X^3 + ((1)*t)*X^2 + ((1)*t^3)*X + (1)*t^4;
It is isomorphic to
y^2 = x^3 + ((84)*t^3+(14)*t^2+(87)*t+(74))*x +
(45)*t^4+(74)*t^3+(42)*t^2+(42)*t+(54);
via X = ((87))*x + ((67)*t+(42));
    Y = ((7))*x + ((65))*y + ((50)*t^2+(17)*t+(80));
or via x = ((36))*X + ((12)*t+(3));
       y = ((7))*X + ((14))*Y + ((7)*t^2);
```

Le programme détecte ensuite que la courbe rentrée n'est pas une courbe à multiplication complexe par un ordre dans un corps quadratique de petit nombre de classes.

```
It seems this curve is neither supersingular nor with ``easy'' complex
multiplication.
```

Exemple d'exécution du programme ***SEA***

Nous rentrons maintenant dans l'algorithme `SEA`. Le programme calcule tout d'abord le polynôme $\text{pgcd}(X^q - X, \Phi_2(X, j_E))$ dans $\mathbb{F}_q$.

```
%% l = 2^1
%%% Computing the extension defined by the modular equation for l = 2
%%% the extension defined by the modular equation for l = 2 computed in 0 s
%%% Computing precomputations for this ring
%%% precomputations for this ring computed in 0.15 s
%%% Computing X^q-X mod phi
%%% X^q-X mod phi computed in 0.02 s
%%% Computing gcd(X^q-X, phi)
%%% gcd(X^q-X, phi) computed in 0 s
%%% Computing the splitting
%%%% Computing precomputations for this ring
%%%% precomputations for this ring computed in 0.18 s
%%% splitting found in 0.18 s
%% l = 2^1 computed in 0.35 s
```

Le programme en déduit ensuite que nous sommes dans le cas (i) du théorème 35 :

```
%% Computing the type of l = 2^1
Splitting [1 2], 2 is a double eigenvalue prime
%% type of l = 2^1 found in 0 s
```

Dans ce cas, il est facile de conclure.

```
%% Computing c for l = 2^1
% c mod 2 is 0
%% c for l = 2^1 found in 0 s
```

Le programme traite ensuite le cas $\ell = 3$.

```
%% l = 3^1
%%% Computing the extension defined by the modular equation for l = 3
%%% the extension defined by the modular equation for l = 3 computed in 0 s
%%% Computing precomputations for this ring
%%% precomputations for this ring computed in 0.05 s
%%% Computing X^q-X mod phi
%%% X^q-X mod phi computed in 0.02 s
%%% Computing gcd(X^q-X, phi)
%%% gcd(X^q-X, phi) computed in 0 s
%%% Computing the splitting
%%%% Computing precomputations for this ring
%%%% precomputations for this ring computed in 0.18 s
%%% splitting found in 0.18 s
%% l = 3^1 computed in 0.25 s
```

Nous avons ainsi 3 qui se rattache au cas (iii) du théorème 35 :

```
%% Computing the type of l = 3^1
Splitting [1 1 r .. r ], 3 is an Elkies prime
%% type of l = 3^1 found in 0 s
```

Nous calculons donc une isogénie de degré 3.

```
%% l = 3^1
%%% Computing F
%%% F computed in 0 s
```

```
%%% Computing an isogeny of degree 3
There is 1 Js to test
%%%% Computing the isogeneous curve(s)
A 3-isogeneous curve is among:
y^2 = x^3 + ((35)*t^4+(83)*t^3+(99)*t^2+(27)*t+(56))*x +
(65)*t^4+(45)*t^3+(28)*t^2+(72)*t+(70)
%%%% the isogeneous curve(s) computed in 0.02 s
%%%% Computing an isogeny of degree 3
Testing p1 = (56)*t^4+(43)*t^3+(67)*t^2+(40)*t+(58)
Good check for this factor.
((1))*X+((45)*t^4+(58)*t^3+(34)*t^2+(61)*t+(43))
%%%% an isogeny of degree 3 found in 0 s
%%% an isogeny of degree 3 found in 0.02 s
%% l = 3^1 computed in 0.02 s
```

Et nous trouvons la valeur propre associée.

```
%% Computing the eigenvalue for l = 3^1
I am going to work modulo a polynomial of degree 1
%%% Computing precomputations for this ring
%%% precomputations for this ring computed in 0.13 s
%%% Computing precomputations for this ring
%%% precomputations for this ring computed in 0.20 s
%%% Computing Y^q
%%% Y^q computed in 0 s
%%% Computing 0*[X, Y, 1]
%%% 0*[X, Y, 1] computed in 0 s
%%% Computing R == 0*[X, Y, 1]
%%% R == 0*[X, Y, 1] computed in 0 s
%%% Computing R == -0*[X, Y, 1]
%%% R == -0*[X, Y, 1] computed in 0 s
%%% Computing 1*[X, Y, 1]
%%% 1*[X, Y, 1] computed in 0 s
%%% Computing 1*[X, Y, 1]
%%% 1*[X, Y, 1] computed in 0 s
%%% Computing R == 1*[X, Y, 1]
%%% R == 1*[X, Y, 1] computed in 0 s
%%% Computing R == -1*[X, Y, 1]
%%% R == -1*[X, Y, 1] computed in 0 s
AOUHH k = 2
%% the eigenvalue for l = 3^1 found in 0.35 s
c mod 3 is 0
```

Le programme traite alors successivement les nombres premiers $5, 7, 11, 13$ et 17. Nous détaillons juste $\ell = 7$ qui correspond au cas (iii) du théorème 35.

```
%% l = 7^1
%%% Computing the extension defined by the modular equation for l = 7
%%% the extension defined by the modular equation for l = 7 computed in 0 s
%%% Computing precomputations for this ring
%%% precomputations for this ring computed in 0.10 s
%%% Computing X^q-X mod phi
%%% X^q-X mod phi computed in 0.03 s
%%% Computing gcd(X^q-X, phi)
%%% gcd(X^q-X, phi) computed in 0 s
```

Exemple d'exécution du programme ***SEA***

```
%%% Computing the splitting
%%% splitting found in 0 s
%% l = 7^1 computed in 0.13 s

%% Computing the type of l = 7^1
Splitting [ r .. r ], 7 is an Atkin prime
%% type of l = 7^1 found in 0 s

%% l = 7^1
% The r's to test are : 8

% The good r is 8
% t mod 7 is among [2 3 4 5 ]
%% l = 7^1 computed in 0 s
```

Finalement, nous avons les résultats suivants.

```
I have these lists of moduli.
t mod 2 is in [ 0 ]
t mod 3 is in [ 0 ]
t mod 5 is in [ 2, 3 ]
t mod 7 is in [ 2, 3, 4, 5 ]
t mod 11 is in [ 1, 4, 7, 10 ]
t mod 13 is in [ 1, 3, 6, 7, 10, 12 ]
t mod 17 is in [ 7, 10 ]

The logarithm of U-primes is 0.78
The logarithm of L-primes is 4.93
The logarithm of all primes is 5.71
The logarithm of 4*sqrt(q) is 5.61
```

Il est ensuite nécessaire de combiner ces données par l'algorithme de "tri-recherche" du chapitre 11. Pour cela, le programme divise les modulos en trois ensembles $\mathfrak{G}$, $\mathfrak{B}$ et $\mathfrak{U}$ qui correspondent ci-dessous respectivement aux "listes" 1, 2 et 3.

```
There are 5 champions.
17 (2), 5 (2), 11 (4), 13 (6), 7 (4)
n := 3.840000e+02
I splitted them in 3 lists.
n := 3.84e+02
The list 1 has got 2 modulis.
17 (2), 13 (6)
M1 := 221;
n1 := 12
The list 2 has got 3 modulis.
5 (2), 11 (4), 7 (4)
M2 := 385;
n2 := 32
The list 3 has got 2 modulis.
2, 3
M3 := 6;
```

Puis, viennent quelques précalculs.

```
%%% Computing t3 = t mod M3
%%% t3 = t mod M3 computed in 0 s
```

```
%%% Computing s2 = -t3/M1/M3 mod M2
%%% s2 = -t3/M1/M3 mod M2 computed in 0 s
%%% Computing s3 = t3+s2*M1*M3 mod M2
%%% s3 = t3+s2*M1*M3 mod M2 computed in 0 s
%%% Computing a point P on the curve
%%% a point P on the curve found in 0.02 s
%%% Computing (q+1-s3)*P
%%% (q+1-s3)*P computed in 0.02 s
%%% Computing M3*P
%%% M3*P computed in 0 s
%%% Computing M2*M3*P
%%% M2*M3*P computed in 0 s
%%% Computing M1*M3*P
%%% M1*M3*P computed in 0 s
```

Les points de la phase “pas de géants” sont alors calculés et stockés.

```
%%% Computing R1
%%%% Computing R1 modulo the list 1
%%%% R1 modulo the list 1 computed in 0 s
%%%% Computing R1 modulo M1
%%%% R1 modulo M1 computed in 0 s
%%%% Computing the shift of R1 in [-M1/2-S2*M1/M2, M1/2-S2*M1/M2]
%%%% the shift of R1 in [-M1/2-S2*M1/M2, M1/2-S2*M1/M2] computed in 0 s
%%%% sorting R1
%%%% sorting R1 computed in 0 s
%%% R1 computed in 0 s
%%% Computing the differences D1 of R1
%%% the differences D1 of R1 computed in 0 s
%%% sorting D1
%%% sorting D1 computed in 0 s
There remains 5 D1
%%% Computing D1*M2*M3*P
%%%% Computing 0*P,...,199*P,...,200^0*P,...,199*200^0*P
%%%% 0*P,...,199*P,...,200^0*P,...,199*200^0*P computed in 0.03 s
%%%% Computing D*P
%%%% D*P computed in 0 s
%%% D1*M2*M3*P computed in 0.05 s
%%% Computing P1 = (q+1-s3)*P-R1*M2*M3*P
%%%% Computing the first step
%%%% the first step computed in 0 s
%%% P1 = (q+1-s3)*P-R1*M2*M3*P computed in 0 s
%%% sorting the abcissae of P1
%%% sorting the abcissae of P1 computed in 0 s
```

Commence alors le calcul des points de la phase “pas de bébé”.

```
%%% Computing R2
%%%% Computing R2 modulo the list 2
%%%% R2 modulo the list 2 computed in 0 s
%%%% Computing R2 modulo M2
%%%% R2 modulo M2 computed in 0 s
%%%% sorting R2
%%%% sorting R2 computed in 0 s
%%% R2 computed in 0 s
```

Exemple d'exécution du programme ***SEA***

```
%%% Computing the differences D2 of R2
%%% the differences D2 of R2 computed in 0 s
%%% sorting D2
%%% sorting D2 computed in 0 s
There remains 9 D2
%%% Computing D2*M1*M3*P
%%%% Computing 0*P,...,199*P,...,200^0*P,...,199*200^0*P
%%%% 0*P,...,199*P,...,200^0*P,...,199*200^0*P computed in 0.03 s
%%%% Computing D*P
%%%% D*P computed in 0 s
%%% D2*M1*M3*P computed in 0.05 s
```

Ce qui conduit au résultat espéré.

```
%%% Computing P2 = R2*M1*M3*P
%%%% Computing the second step
AHOUH, I found a pseudo match ...
%%%%% Computing R1
%%%%% R1 computed in 0 s
We have R1 = -5
%%%%% Computing R2
%%%%% R2 computed in 0 s
We have R2 = 52
%%%%% Computing c
So, we have probably c = -80502
Do we have (q+1-c)*P = 0 for a random point P ?
YAOUH, we have (q+1-c)*P = 0.
%%%%% c computed in 0.02 s
%%%% the second step computed in 0.02 s
%%% P2 = R2*M1*M3*P computed in 0.02 s
%% the Match & Sort step computed in 0.15 s
The number of points on this curve is :
10510100501+1+80502=10510181004
% computed in 3.02 s
```

Bibliographie

[1] M. L. Abell and J. P. Braselton. *The Mathematica HandBook*. Academic Press, 1992.

[2] L. M. Adleman, R. L. Rivest, and A. Shamir. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2) :120–126, 1978.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Reading. Addison–Wesley, 1974.

[4] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime, 1988. Email on the Number Theory Mailing List.

[5] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime, 1991. Email on the Number Theory Mailing List.

[6] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61(203) :29–68, July 1993.

[7] C. Batut, D. Bernardi, H. Cohen, and M. Olivier. *PARI-GP, User's guide*, 1995. Available at `ftp ://megrez.math.u-bordeaux.fr/pub/pari`.

[8] A. Bender and G. Castagnoli. On the implementation of elliptic curve cryptosystems. In G. Brassard, editor, *Advances in Cryptology*, volume 435 of *Lecture Notes in Comput. Sci.*, pages 186–192. Springer-Verlag, 1989. Proc. Crypto '89, Santa Barbara, August 20–24.

[9] F. Bergeron, J. Berstel, S. Brlek, and C. Duboc. Addition chains using continued fractions. *Journal of Algorithms*, 10(3) :403–412, September 1989.

[10] I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian. *Applications of finite fields*. Kluwer Academic Publishers, 1993. Editor, A. J. Menezes.

[11] A. Borel, S. Chowla, C. S. Herz, K. Iwasawa, and J. P. Serre. *Seminar on complex multiplication*. Number 21 in Lecture Notes in Math. Springer, 1966.

[12] J. Bos and M. Coster. Addition chain heuristics. In *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Comput. Sci.* Springer-Verlag, 1989.

[13] A. Brauer. On addition chains. *Bull. AMS*, 10(3) :403–412, September 1989.

[14] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solutions of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, pages 259–295, 1980.

[15] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25 :581–595, 1978.

[16] R. P. Brent and H. T. Kung. Systolic VLSI arrays for linear time GCD computation. In *VLSI'83*, pages 145–154, 1983.

[17] B. Buchberger and T. Jebelean. Parallel rational arithmetic for computer. Algebra systems : Motivating experiments. Technical report, RISC-LINZ, april 1992.

[18] J.J. Cannon. The magma system for algebra, number theory and geometry. Available at `http ://www.maths.usyd.edu.au :8000/comp/magma/Overview.html`, 1996.

[19] D. G. Cantor. On arithmetical algorithms over finite fields. *Journal of Combinatorial Theory*, 50(Series A) :285–300, 1989.

[20] D. G. Cantor and E. Kaltofeln. On fast polynomials over arbtrary algebras. *Acta Informatica*, 28 :693–701, 1991.

[21] F. Chabaud. Sécurité des crypto-systèmes de McEliece. Mémoire de DEA, École polytechnique, 91128 Palaiseau CEDEX, France, July 1993.

[22] F. Chabaud. *Recherche de performance dans l'algorithmique des corps finis. Applications à la cryptographie.* thèse, École polytechnique, October 1996.

[23] F. Chabaud and R. Lercier. *ZEN, User Manual.* Laboratoire d'informatique de l'École polytechnique (LIX), 1996. Available at `http ://lix.polytechnique.fr/~zen/`.

[24] J. Chao, K. Tanada, and S. Tsujii. Design of elliptic curves with controllable lower boundary of extension degree for reduction attacks. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Comput. Sci.*, pages 50–55. Springer-Verlag, 1994. Proc. 14th Annual International Cryptology Conference, Santa Barbara, Ca, USA, August 21–25.

[25] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, 1991.

[26] L. S. Charlap, R. Coley, and D. P. Robbins. Enumeration of rational points on elliptic curves over finite fields. Draft, 1991.

[27] H. Cohen. *A course in computational algebric number theory*, volume 138 of *Graduate Texts in Mathematics.* Springer Verlag, 1993.

[28] G. E. Collins. Lecture notes on arithmetic algorithms, 1980.

[29] L. Comtet. Calcul pratique des coefficients de Taylor d'une fonction algébrique. *Enseignement Math.*, pages 267–270, 1964.

[30] J. M. Couveignes. *Quelques calculs en théorie des nombres.* thèse, Université de Bordeaux I, July 1994.

[31] J. M. Couveignes. Computing l-isogenies with the p-torsion. In H. Cohen, editor, *ANTS-II*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 59–65. Springer-Verlag, 1996.

[32] J. M. Couveignes. Isomorphisms between towers of artin-schreier exetensions over a finite fields. Draft, 1997.

[33] J.-M. Couveignes, L. Dewaghe, and F. Morain. Isogeny cycles and the Schoof-Elkies-Atkin algorithm. Research Report LIX/RR/96/03, LIX, April 1996.

[34] J.M. Couveignes and F. Morain. Schoof's algorithm and isogeny cycles. In L. Adleman and M. D. Huangs, editors, *ANTS-I*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 43–58. Springer-Verlag, May 1994.

[35] M. Deuring. Die Typen der Multiplikatorenringe elliptischer Funktionenkörper. *Abh. Math. Sem. Hamburg*, 14 :197–272, 1941.

[36] L. Dewaghe. Remarques sur l'algorithme SEA. In preparation, December 1995.

[37] P. Downey, B. Leony, and R.Sethi. Computing sequences with addition chains. *SIAM J. Comput.*, 3 :638–696, 1981.

[38] P. Dubois. *Software portability with imake.* Nutshell handBook. O'Reilly & Associates, Inc, 1993.

[39] N. D. Elkies. Explicit isogenies. Draft, 1991.

[40] N. D. ELkies. Elliptic and modular curves over finite fields and related computational issues. In *Computational Perspectives On Number Theory*, 1997. To appear. In honor of A. O. L. Atkin. Held September, 1995 in Chicago.

[41] Free Software Foundation. CVS library. Available at `ftp ://prep.ai.mit.edu/pub/gnu/cvs-1.9.tar.gz`, 1996.

[42] Free Software Foundation. GNU MP library. Available at `ftp ://prep.ai.mit.edu/pub/gnu/gmp-2.0.2.tar.gz`, 1996.

[43] A. Fröhlich. *Formal groups*, volume 74 of *Lecture Notes in Math.* Springer-Verlag, 1968.

[44] Numerical Algorithms Group. Axiom. Available at `http ://www.nag.co.uk :80/symbolic/AX.html`, 1996.

[45] H. Gunji. The hasse invariant and p-division points of an elliptic curve. *Arch. Math.*, 27(2), 1976.

[46] G. Harper, A. Menezes, and S. Vanstone. Public-key cryptosystems with very small key length. In R. A. Rueppel, editor, *Advances in Cryptoloy – EUROCRYPT '92*, volume 658 of *Lecture Notes in Comput. Sci.*, pages 163–173. Springer-Verlag, 1993. Workshop on the Theory and Application of Cryptographic Techniques, Balatonfüred, Hungary, May 24–28, 1992, Proceedings.

[47] H. Hasse. Beweis des Analogons der Riemannschen Vermutung für die Artinschen und F. K. Smidtschen Kongruenzzetafunktionen in gewissen elliptischen Fällen. *Ges. d. Wiss. Narichten. Math.-Phys. Klasse*, pages 253–262, 1933.

[48] J.-C. Hervé, B. Serpette, and J. Vuillemin. BigNum : A portable and efficient package for arbitrary-precision arithmetic. Technical Report 2, Digital Paris Research Laboratory, May 1989. Available at `doc-server@prl.dec.com`.

[49] E. W. Howe. On the group orders of elliptic curves over finite fields. *Compositio Mathematica*, 85 :229–247, 1993.

[50] D. Husemöller. *Elliptic curves*, volume 111 of *Graduate Texts in Mathematics*. Springer, 1987.

[51] T. Jebelean. Improving the multiprecision euclidean algorithm. *Lecture Notes in Comput. Sci.*, 722 :45–58, 1992.

[52] T. Jebelean. A generalization of the binary GCD algorithm. In *Proc. ISSAC'93, Kiev, Ukraine*, July 1993.

[53] B. S. Kaliski, Jr. A pseudo-random bit generator based on elliptic logarithms. In *Proc. Crypto 86*, volume 263 of *Lecture Notes in Comput. Sci.*, 1986. Proceedings Crypto '86, Santa Barbara (USA), August 11–15, 1986.

[54] S. A. Katre and A. R. Rajwade. Jacobsthal sums of prime order. *Indian J. pure appl. Math.*, 17(12) :1345–1362, Dec 1986.

[55] S. A. Katre and A. R. Rajwade. Resolution of sign ambiguity in the determination of the cyclotomic numbers of order 4 and the corresponding jacobsthal sum. *Math. Scand.*, 60 :52–62, 1987.

[56] D. E. Knuth. *The Art of Computer Programming : Seminumerical Algorithms*. Addison-Wesley, 2nd edition, 1981.

[57] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177) :203–209, January 1987.

[58] N. Koblitz. Primality of the number of points on an elliptic curve over a finite field. *Pacific Journal of Mathematics*, 131(1) :157–165, 1988.

[59] N. Koblitz. Elliptic curve implementation of zero-knowledge blobs. *Journal of Cryptology*, 4(3) :207–213, 1991.

[60] K. Koyama and Y. Tsuroka. Speeding up elliptic cryptosystems by using a signed binary window method. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Comput. Sci.*, pages 345–357. Springer-Verlag, 1992.

[61] G. J. Lay and H. G. Zimmer. Constructing elliptic curves with given group order over large finite fields. In L. Adleman and M.-D. Huang, editors, *ANTS-I*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 250–263. Springer-Verlag, 1994. 1st Algorithmic Number Theory Symposium - Cornell University, May 6-9, 1994.

[62] D. H. Lehmer. Euclid's algorithm for large numbers. *American Mathematical Monthly*, 45 :227–233, 1938.

[63] A. Lenstra and P. Leyland. Free version of the LIP package. Available at `ftp ://pub/math/freelip/freelip_1.0.tar.gz`, 1996.

[64] H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Math.*, 126 :649–673, 1987.

[65] R. Lercier. Factoriser des entiers par la méthode des courbes elliptiques. Mémoire de DEA, École polytechnique, 91128 Palaiseau CEDEX, France, July 1993.

[66] R. Lercier. Optimisation de l'algorithme de factorisation d'entier découvert par H. W. Lenstra. Mémoire de super-projet, École Nationale Supérieure des Techniques Avancées, Paris, France, June 1994.

[67] R. Lercier. Computing isogenies in GF(2^n). In H. Cohen, editor, *ANTS-II*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 197–212. Springer-Verlag, 1996.

[68] R. Lercier. Finding good random elliptic curves for cryptosystems definied over $\mathbb{F}_{2^n}$. In *Advances in Cryptoloy – EUROCRYPT '97*, LNCS. Springer-Verlag, 1997. To appear.

[69] R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields : strategies and performances. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT '95*, number 921 in Lecture Notes in Comput. Sci., pages 79–94, 1995. International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 1995, Proceedings.

[70] R. Lercier and F. Morain. Counting the number of points on elliptic curves over F_{p^n} using Couveignes's algorithm. Rapport de Recherche LIX/RR/95/09, Laboratoire d'Informatique de l'École polytechnique (LIX), 1995. Available at `http://lix.polytechnique.fr/~morain/Articles`.

[71] R. Lercier and F. Morain. Counting the number of points on elliptic curves over F_{p^n} using Couveignes's algorithm. Submitted at Math. Comp., 1995.

[72] R. Lercier and F. Morain. Algorithms for computing isogenies between elliptic curves. In *Computational Perspectives On Number Theory*, 1997. To appear. In honor of A. O. L. Atkin. Held September, 1995 in Chicago.

[73] Group LiDIA. *LiDIA Manual. A library for computational number theory.*, 1996. Available at `http ://www.umbc.edu/pdsrc/docs/lidiahtml/lidiahtml.html`.

[74] R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison–Wesley, 1983.

[75] J. L. Massey. Shift-register and BCH decoding. *IEEE Trans. on Information Theory*, IT-15(1) :122–127, January 1969.

[76] R. McEliece. *Finite fields for computer scientists and engineers*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1988.

[77] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEETIT*, 39(5) :1639–1646, 1993.

[78] A. Menezes and S. A. Vanstone. The implementation of elliptic curve cryptosystems. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology*, number 453 in Lecture Notes in Comput. Sci., pages 2–13. Springer–Verlag, 1990. Proceedings Auscrypt '90, Sysdney (Australia), January 1990.

[79] A. J. Menezes. *Elliptic curve public key cryptosystems*. Kluwer Academic Publishers, 1993.

[80] A. J. Menezes, S. A. Vanstone, and R. J. Zuccherato. Counting points on elliptic curves over F_{2^m}. *Math. Comp.*, 60(201) :407–420, January 1993.

[81] S. M. Meyer and J. P. Sorenson. Efficient algorithms for computing the Jacobi symbol. In H. Cohen, editor, *ANTS-II*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 225–239. Springer-Verlag, 1996.

[82] V. Miller. Use of elliptic curves in cryptography. In A. M. Odlyzko, editor, *Advances in Cryptology*, volume 263 of *Lecture Notes in Comput. Sci.*, pages 417–426. Springer-Verlag, 1987. Proceedings Crypto '86, Santa Barbara (USA), August11–15, 1986.

[83] A. Miyaji. On ordinary elliptic curve cryptosystems. In *Advances in Cryptology – ASIACRYPT '91*, volume 739 of *Lecture Notes in Comput. Sci.*, pages 50–55. Springer-Verlag, 1991.

[84] A. Miyaji. Elliptic curves over F_p suitable for cryptosystems. In J. Seberry and Y. Zheng, editors, *Advances in cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Comput. Sci.*, pages 479–491. Springer-Verlag, 1993. Workshop on the theory and application of cryptographic techniques, Gold Coast, Queensland, Australia, December 13-16, 1992.

[85] T. Miyake. *Modular forms*. Springer-Verlag, 89.

[86] R. Moenck. Fast computation of GCD's. In *5-th annual ACM Symposium on Theory of computing*, pages 142–151, 1973.

[87] P. L. Montgomery. *An FFT extension of the Elliptic Curve Method of factorization*. PhD thesis, University of California – Los Angeles, 1992.

[88] F. Morain. *Courbes elliptiques et tests de primalité*. thèse, Université Claude Bernard–Lyon I, September 1990.

[89] F. Morain. Building cyclic elliptic curves modulo large primes. In D. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Comput. Sci.*, pages 328–336. Springer–Verlag, 1991. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Brighton, United Kingdom, April 8–11, 1991.

[90] F. Morain. *Modular arithmetic : the BigMod library*, 1992.

[91] F. Morain. Calcul du nombre de points sur une courbe elliptique dans un corps fini : aspects algorithmiques. *J. Théor. Nombres Bordeaux*, 7 :255–282, 1995.

[92] F. Morain. Classes d'isomorphismes des courbes elliptiques supersingulières en caractéristique ≥ 3. To appear in Utilitas Mathematica. Preprint, available at `http://lix.polytechnique.fr/~morain/`, March 1996.

[93] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *R.A.I.R.O. Theoretical Informatics and Applications*, 24(6) :531–543, 1990.

[94] V. Müller. *Ein Algorithmus zur bestimmung der Punktanzahl elliptisher kurven über endlichen körpen der charakteristik größer drei*. PhD thesis, Technischen Fakultät der Universität des Saarlandes, February 1995.

[95] J. L. Nicolas. Calcul des champions, 1994. Email to F. Morain with a program written in MAPLE.

[96] A. Nitaj. L'algorithme de Cornacchia. *Expositiones Mathematicae*, 13(4) :358–365, 1995.

[97] G. Norton. A shift remainder GCD algorithm. *Lecture Notes in Comput. Sci.*, 356 :350–356, 1987.

[98] M. E. Pohst. Kash : Kant shell (komputational algebraic number theory). Available at `ftp ://ftp.math.tu-berlin.de/pub/algebra/Kant/Kash/Binaries`, 1996.

[99] B. Salvy and P. Zimmermann. GFUN, a MAPLE package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2) :163–177, 1994.

[100] B. Schoeneberg. *Elliptic modular functions*, volume 203 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen*. Springer–Verlag, 1974.

[101] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44 :483–494, 1985.

[102] R. Schoof. Nonsingular plane cubic curves over finite fields. *J. Combin. Theory Ser.*, 46 :183–211, 1987.

[103] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7 :219–254, 1995. Available at `http://www.emath.fr/Maths/Jtnb/jtnb1995-1.html`.

[104] J. P. Serre. *Cours d'arithmétique*. PUF, 1970.

[105] T. Setz. *Integration von mechanismen zur unterstützung der fehlertoleranz in LiPS*. PhD thesis, Universität des Saarlandes, Saarbrücken, January 1996.

[106] J. Shallit and J. Sorenson. Analysis of a left-shift binary GCD. In *Algorithmic Number Theory Symposium*, May 1994.

[107] D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math. vol. 20*, pages 415–440. AMS, 1971.

[108] A. Shönhage. Schnelle berechnung von kettenbruchentwicklungen. *Acta Informatica*, pages 139–144, 1971.

[109] A. Shönhage. A lower bound on the length of addition chains. *Theoretical Computer Science*, 1 :1–12, 1975.

[110] V. Shoup. A new polynomial factorization algorithm and its implementation. *JSC*, 20 :363–397, 1995.

[111] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 1986.

[112] C. Small. *Arithmetic of finite fields*. Marcel Dekker, 1991.

[113] J. Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus de l'Académie des Sciences de Paris*, 273 :238–241, 1971. Série A.

[114] J. F. Voloch. Explicit p-descent for elliptic curves in characteristic p. *CM*, 74 :247–258, 1990.

[115] J. F. Voloch. An analogue of the weierstrass ζ-function in characteristic p. *AA*, LXXIX.1 :1–6, 1997.

[116] E. Waterhouse. Abelian varieties over finite fields. *Ann. Sci. École Norm. Sup.*, 2 :521–560, 1969.

[117] K. Weber. The accelerated integer GCD algorithm. *ACM Transactions on Mathematical Software*, 21 :111–122, 1995.

ÉCOLE POLYTECHNIQUE

Année universitaire 19.96 19..97.

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

TITRE DE LA THESE: ALGORITHMIQUE DES COURBES ELLIPTIQUES DANS LES CORPS FINIS

NOM PRENOM DE L'AUTEUR: LERCIER, Reynald.

MEMBRES DU JURY: J.-J. QUISQUATER; B. VALLÉE; J.-M. COUVEIGNES; F. MORAIN; J.-M. STEYAERT

PRESIDENT DU JURY: J.-J. QUISQUATER

DATE DE LA SOUTENANCE: 16/06/97

REPRODUCTION DE LA THESE SOUTENUE:

- [x] THESE POUVANT ÊTRE REPRODUITE EN L'ETAT
- [ ] THESE NE POUVANT PAS ÊTRE REPRODUITE
- [ ] THESE POUVANT ÊTRE REPRODUITE APRES CORRECTIONS

Signature du Président du Jury,

J.J. Quisquater

CET IMPRIME DOIT ÊTRE RENVOYE APRES SIGNATURE
A LA DIVISION DE L'ETUDIANT AVEC LE RAPPORT DE SOUTENANCE

91128 PALAISEAU CEDEX - Tél (33) 1 01 69 33 45 81- 45 82 - Fax : (33) 01 69 33 30 59

RÉSUMÉ : cette thèse est consacrée au calcul du nombre de points d'une courbe elliptique définie sur un corps fini. Nous étudions dans une première partie l'algorithme de Schoof et ses variantes dues à Atkin et à Elkies. Nous montrons ainsi en quelle mesure ces algorithmes, initialement prévus pour des corps de grande caractéristique, peuvent s'appliquer à ceux de petite caractéristique.

Il s'avère que la majeure partie des idées d'Atkin et Elkies sont applicables à cette dernière famille de corps, à quelques modifications mineures près, excepté le calcul d'isogénies entre courbes elliptiques. C'est pourquoi, nous étudions dans une deuxième partie cinq algorithmes de calcul d'isogénies. Le premier algorithme est l'algorithme original d'Atkin pour le cas des corps de grande caractéristique. Les deuxième et troisième algorithme sont ceux de Couveignes pour les corps de petite caractéristique. Enfin, nous proposons à notre tour un quatrième algorithme pour le cas spécifique de la caractéristique deux et montrons dans un cinquième algorithme comment ces idées peuvent se généraliser à des corps de caractéristique p impaire pour des isogénies de degré ℓ borné par $2p$.

D'un point de vue plus pratique, nous explicitons dans une troisième partie les méthodes de programmation mises en œuvre pour implanter les algorithmes précédents. Nous y présentons notamment `ZEN`, une librairie de calcul qui permet une programmation efficace en langage `C` de toute extension algébrique d'un anneau fini. Enfin, nous expliquons comment nous utilisons le programme obtenu pour calculer efficacement le nombre de points de courbes définies dans tout corps finis à moins de 10^{100} éléments. En particulier, nous montrons comment trouver ainsi des courbes elliptiques ayant de bonnes propriétés cryptographiques.

MOTS CLÉS : corps finis, courbes elliptiques, algorithme de Schoof, isogénies, équations modulaires, groupes formels.

ABSTRACT : This thesis deals with computations of cardinality of elliptic curves which are defined over a finite field. In a first part, we study Schoof's algorithm and variants due to Atkin and Elkies. We show how these algorithms, initially designed for finite fields of large characteristic, can be applied to fields of small characteristic.

It turns out that most of Atkin's and Elkies' ideas can be used in the last case, except for computing isogenies between elliptic curves. We therefore study five algorithms for computing isogenies in part II. First algorithm is Atkin's original algorithm for fields of large characteristic. Second and the third are Couveignes's algorithms for finite fields of small characteristic. Finally, we propose a fourth algorithm specially designed for finite fields of characteristic two, and we show in fifth algorithm, how we can extend these ideas for finite fields of odd characteristic p and isogenies of degree ℓ smaller than $2p$.

From a practical point of view, we explain how we have programmed the previous algorithms in a third part. In particular, we introduce `ZEN`, a programming library written in `C`-language which efficiently computes in every finite extension over a finite ring. Then, we explain how we used the obtained program for efficiently computing number of points of curves defined over any finite fields whose number of points is smaller than 10^{100}. Moreover, we describe how we can find elliptic curves with good cryptographic properties.

KEYWORDS : Finite fields, elliptic curves, Schoof's algorithm, isogenies, modular equations, formal groups.