Algorithmique des courbes algébriques pour la cryptologie

Pierrick Gaudry

Algorithmique des courbes algébriques pour la cryptologie

Pierrick Gaudry

Mémoire présenté et soutenu publiquement le 8 octobre 2008
pour l'obtention de l'

Habilitation à diriger des recherches

de l'Université Henri Poincaré – Nancy 1

Composition du jury :

Rapporteurs :	Alfred Menezes, professeur à l'Université de Waterloo
	Jean-François Mestre, professeur à l'Université de Paris 7
	Brigitte Vallée, directrice de recherche CNRS à l'Université de Caen
Examinateurs :	Daniel Bernstein, professeur à l'University of Illinois at Chicago
	Didier Galmiche, professeur à l'Université de Nancy 1
	Arjen Lenstra, professeur à l'École polytechnique fédérale de Lausanne
	Paul Zimmermann, directeur de recherche INRIA, LORIA

Elegance is an algorithm.
Iain M. Banks, *The Algebraist*

Table des matières

Chapitre 1

Introduction

1.1 Un bref résumé de mon parcours de chercheur

Je suis entré dans le monde de la recherche en 1995, lors de mon stage de DEA que j'ai effectué au laboratoire LIX de l'École polytechnique, encadré par F. Morain. J'ai par la suite continué sur le même sujet de recherche lors d'une thèse que j'ai soutenue en décembre 2000, toujours sous la direction de F. Morain. Durant cette première période, j'ai découvert la thématique de l'algorithmique des courbes appliquée à la cryptologie. J'en ai exploré différents aspects, en particulier autour des problèmes de comptage de points et de logarithme discret.

Après des visites de quelques mois dans des laboratoires à l'étranger, j'ai été embauché au CNRS en 2001, avec une affectation toujours au LIX. J'y ai poursuivi mes recherches, approfondissant des aspects que j'avais étudiés en thèse, et en découvrant de nouveau. Au fur et à mesure des collaborations, je me suis ouvert à des problématiques relevant plus du calcul formel que de la théorie algorithmique des nombres ou de la cryptographie. C'est ainsi que j'ai pris goût à l'algorithmique asymptotiquement rapide pour les polynômes et les entiers.

En 2005, j'ai effectué une mutation vers le LORIA, pour rejoindre le projet SPACES, devenue depuis l'équipe-projet CACAO. Ceci, m'a permis de concrétiser une certaine volonté d'infléchir mes recherches : en m'appuyant sur les compétences de l'équipe, j'ai commencé à travailler sur des problématiques d'arithmétique rapide. Mes propres compétences en programmation efficace, que j'avais acquises au cours de mes recherches sur l'utilisation des courbes algébriques en cryptographie, ont ainsi pu être mises en valeur.

Les deux premières parties de ce mémoire sont consacrées aux thématiques qui restent mes recherches principales : l'utilisation des courbes algébriques en cryptographie, soit du point de vue de la conception de cryptosystèmes, soit du point de vue de la cryptanalyse. Dans une troisième partie, plus courte, je décris quelques travaux récents, qui concernent des problèmes d'arithmétique efficace.

Les orientations futures que je souhaite donner à ma recherche seront discutées en conclusion de document, lorsqu'il me sera possible d'évoquer telle ou telle partie de mes travaux de manière plus précise.

1.2 Les problèmes algorithmiques sur les courbes en lien avec la cryptographie

1.2.1 La cryptographie

L'information est précieuse. Lors de son stockage ou de sa transmission, il est nécessaire de la protéger. Deux grands types de protection se distinguent : la protection contre les altérations dues à des problèmes physiques (typiquement une perturbation par un bruit gaussien), et la protection contre des ennemis malicieux. La théorie des codes correcteurs d'erreurs et la cryptologie sont les domaines de recherche associés à ces problématiques. La cryptologie, qui nous intéresse plus particulièrement ici, se divise en deux disciplines complémentaires et indissociables : la cryptographie, la conception de systèmes de protection, et la cryptanalyse, l'étude des attaques des systèmes connus.

Les propriétés que la cryptographie doit garantir sont variées. La confidentialité est celle qui vient en premier à l'esprit, mais l'authenticité et l'intégrité sont largement aussi cruciales (personne ne peut écouter ma conversation, mais suis-je bien sûr de l'identité de mon interlocuteur et que ce que j'envoie est bien ce qui est reçu ?) Des propriétés plus complexes comme lorsqu'on souhaite effectuer un vote électronique sont aussi largement étudiées.

Un produit cryptographique tel que le logiciel SSH qui permet de se connecter à distance à un ordinateur de manière sécurisée est le résultat d'une longue chaîne dont aucun maillon ne doit présenter de faiblesse afin d'assurer la sécurité de l'ensemble. Au plus bas niveau, se trouvent les primitives cryptographiques élémentaires. Par exemple un algorithme de chiffrement par bloc, paramétré par une clef ; ou bien une fonction de hachage ; ou encore un algorithme de signature. Ces primitives présentent des propriétés de sécurité bien définies, mais insuffisantes pour la plupart des utilisations. Par exemple un chiffrement par bloc, de par sa nature déterministe, chiffre un même texte toujours de la même manière, ce qui ne convient pas à une discussion très formatée avec des réponses oui/non. Pour certaines primitives, la sécurité se ramène directement à un problème algorithmique bien identifié (comme la factorisation d'entiers), pour d'autres, elle est conjecturée suite à des considérations liées aux grandes familles de cryptanalyses connues. Au niveau immédiatement supérieur, les primitives cryptographiques sont combinées en des protocoles effectuant des tâches plus complexes. À ce niveau là, en général une preuve de sécurité est exigée. On définit un modèle de sécurité très strict donnant d'énormes pouvoirs à l'attaquant (non seulement des capacités de calcul, mais aussi la possibilité d'intervenir activement dans les communications). Ensuite, il s'agit d'établir une preuve mathématique qu'un attaquant parvenant à briser le système aura en fait le pouvoir de contredire une des hypothèses de sécurité d'une des primitives. Ainsi la couche protocole ne rajoute pas de faiblesse. À partir de ces protocoles, on peut définir des standards précis précisant les différents choix de paramètres. Et pour finir, au dernier niveau, il faut implanter ce standard. Lors de cette dernière phase, selon le domaine d'application on peut ou non exiger une preuve formelle de l'exactitude de l'implantation (voire de la compilation).

Dans ce mémoire nous nous intéressons au tout premier niveau : la conception de primitives cryptographiques et l'étude de leur sécurité élémentaire. Les systèmes à base de courbes algébriques nous occupent plus particulièrement. Dans ce cadre, il s'agit de primitives dont la sécurité est liée au problème du logarithme discret dans des groupes finis liés aux courbes. Certaines primitives ont une sécurité élémentaire exactement équivalente à celle du logarithme discret, mais la plupart vont être équivalentes à des problèmes légèrement différents comme le problème de Diffie-Hellman. Les liens de ces problèmes avec celui du logarithme discret sont suffisamment étroits pour que l'on se concentre uniquement sur ce dernier. Par ailleurs, les groupes associés aux courbes viennent naturellement équipés d'une forme bilinéaire non dégénérée qui est parfois calculable efficacement (c'est notamment le cas pour les courbes supersingulières). Il existe de nombreuses primitives s'appuyant sur l'existence de cette structure bilinéaire, et dont la sécurité repose sur des problèmes difficiles associés. Nous ne parlerons pas ou peu de cet aspect de la cryptographie à base de courbes dans ce mémoire.

1.2.2 Les courbes algébriques et leur jacobienne

Cette section contient quelques rappels ; des livres de référence contenant des résultats plus complets, ainsi que les preuves sont par exemple [113, 161, 55]. Pour les résultats sur les courbes hyperelliptiques, on pourra consulter [124] ou [25].

Soit k un corps. Les objets qui nous intéressent sont les courbes algébriques définies sur k. Il s'agit des variétés algébriques projectives lisses de dimension 1 absolument irréductibles. En fait on travaille presque toujours en pratique avec un modèle plan, affine, éventuellement singulier, tout en gardant à l'esprit que c'est le modèle complété et désingularisé qui est le véritable objet étudié.

Soit donc $\mathcal{C}$ une courbe définie sur k. Un diviseur de $\mathcal{C}$ est une somme formelle finie de

points de $\mathcal{C}$ sur une clôture algébrique $\bar{k}$ de k. Ainsi un diviseur D de $\mathcal{C}$ s'écrit

$$D = \sum_{P \in \mathcal{C}} n_P P,$$

où les n_P sont des entiers presque tous nuls. L'entier $\sum n_P$ est appelé le degré de D. Les diviseurs forment un groupe abélien, et les diviseurs de degré zéro forment un sous-groupe du groupe des diviseurs. Un diviseur dont tous les coefficients sont positifs est appelé diviseur effectif, et dans le même ordre d'idée on dit que $D \geqslant E$ si et seulement si $D - E$ est effectif. Le corps de fonctions $k(\mathcal{C})$ de la courbe $\mathcal{C}$ est l'ensemble des fonctions rationnelles de $\mathcal{C}$ vers k. À toute fonction f de $k(\mathcal{C})$, on peut associer un diviseur noté $\mathrm{div} f$ en mettant un coefficient nul pour chaque point de $\mathcal{C}$ où f est bien définie et non nulle, le degré d'annulation lorsque f s'annule, et le degré du pôle lorsque $1/f$ s'annule. On peut montrer que le diviseur ainsi formé est bien défini dès que f est non nulle, et qu'il est de degré 0. Un tel diviseur provenant d'une fonction est appelé diviseur principal. Les diviseurs principaux forment un sous-groupe des diviseurs de degré 0. Le groupe quotient des diviseurs de degré 0 par le groupe des diviseurs principaux est appelé le groupe de Picard de $\mathcal{C}$. Il s'agit d'un objet auquel on peut adjoindre une structure de variété algébrique projective ; on parle alors de la variété jacobienne de $\mathcal{C}$. C'est cette dernière terminologie qui sera utilisée dans ce mémoire, même lorsque la structure géométrique n'est pas mentionnée.

Soit k' une extension de k incluse dans $\bar{k}$. Un diviseur est défini sur k' si et seulement s'il est invariant sous l'action du groupe de Galois de $\bar{k}$ sur k'. Cela ne signifie pas que les points composant le diviseur sont eux-mêmes définis sur k', mais que le diviseur est une somme d'orbites de points conjugués sous l'action de ce groupe de Galois. Cette définition est compatible avec la notion de diviseur principal : si une fonction est définie sur k', alors son diviseur associé l'est aussi. Un élément de la jacobienne est défini sur k' s'il existe un représentant dans sa classe qui est défini sur k'. Sous des conditions peu contraignantes[1], les différentes notions que l'on peut imaginer sont compatibles : on peut prendre le quotient par les diviseurs principaux définis sur k', ou bien par les diviseurs principaux définis sur $\bar{k}$ et ensuite requérir que la classe contienne un élément défini sur k', sans changer le résultat.

La jacobienne d'une courbe est l'objet qui va nous intéresser. Dans le cas où k est un corps fini, la jacobienne est un groupe abélien fini, dans lequel on va disposer d'algorithmes efficaces pour la loi de groupe. Il s'agit donc d'un candidat naturel pour fabriquer des primitives reposant sur le problème du logarithme discret.

L'élément essentiel pour montrer le caractère fini de la jacobienne et pour représenter ses points de manière unique est le théorème de Riemann-Roch. L'énoncé fait intervenir les espaces vectoriels suivants. Soit D un diviseur défini sur k, l'espace $L(D)$ est l'ensemble des fonctions sur la courbe dont le diviseur est plus grand que $-D$:

$$L(D) = \{f \in k(\mathcal{C});\ f \neq 0, \mathrm{div}(f) \geqslant -D\} \cup \{0\}.$$

On adjoint la fonction nulle de sorte que $L(D)$ forme un espace vectoriel sur k. La dimension de cet espace vectoriel est notée $l(D)$. Cette dimension est finie, et est mesurée plus précisément par le théorème de Riemann-Roch.

[1] Il suffit que $\mathcal{C}$ admette un point défini sur k. On s'intéressera surtout aux courbes de genre petit sur des corps finis suffisamment gros pour que ça soit toujours le cas.

Théorème 1 (Riemann-Roch) *Soit $\mathcal{C}$ une courbe définie sur un corps k. Il existe un entier g et un diviseur W tels que pour tout diviseur D,*

$$l(D) = \deg(D) + 1 - g + l(W - D).$$

L'entier g est unique et est appelé le genre de $\mathcal{C}$. Le diviseur W n'est pas unique ; un W tel que dans le théorème est appelé diviseur canonique. Nous allons maintenant montrer comment le théorème de Riemann-Roch permet de trouver un représentant petit dans chaque classe de diviseurs de degré 0. Pour cela, on se fixe un point de $\mathcal{C}$ défini sur k que l'on note P_∞ ; si un tel point n'existe pas, on peut toujours prendre à la place un diviseur de degré 1. Soit D un diviseur de degré 0. Considérons le diviseur $D' = D + gP_\infty$; le théorème donne $l(D') = 1 + l(W - D) \geqslant 1$. Ainsi il existe une fonction f telle que $\mathrm{div}(f) + D' \geqslant 0$. Notons E ce diviseur $\mathrm{div}(f) + D'$ qui est effectif, de degré g ; par définition D et $E - gP_\infty$ sont dans la même classe. Ainsi chaque élément de la jacobienne peut être représenté par un diviseur effectif de degré g. La finitude de la jacobienne s'en déduit immédiatement. On peut même pousser plus loin la construction en prenant pour E un diviseur effectif de degré m minimal tel que D est équivalent à $E - mP_\infty$. On a alors un représentant unique de la classe de D, et le diviseur $E - mP\infty$ est appelé diviseur réduit.

Le cas des courbes hyperelliptiques

Une courbe hyperelliptique est une courbe $\mathcal{C}$ tel qu'il existe un morphisme de degré 2 de $\mathcal{C}$ vers $\mathbb{P}^1$. De manière plus concrète, cela signifie que $\mathcal{C}$ admet une équation de la forme $y^2 + h(x)y = f(x)$, où f et h sont deux polynômes. Si de plus $\mathcal{C}$ admet un point de Weierstrass rationnel, on peut imposer que $\deg f = 2g + 1$ et que $\deg h \leqslant g$ et l'on parle d'un modèle imaginaire pour l'équation de $\mathcal{C}$. Dans le cas où la caractéristique du corps est différente de 2, on peut même forcer $h = 0$. L'application $\iota : \mathcal{C} \to \mathcal{C}$ défini par $\iota(x, y) = (x, -y - h(x))$ est une involution appelée involution hyperelliptique.

La notion de diviseur réduit définie à l'aide du théorème de Riemann-Roch se spécialise dans le cas hyperelliptique imaginaire de manière très simple : c'est ce qu'on appelle la représentation de Mumford. On choisit pour point de base P_∞ l'unique point à l'infini de $\mathcal{C}$. Soit D un diviseur de degré 0 et $E - mP_\infty$ le diviseur réduit correspondant. On forme alors le polynôme $u(x)$ de degré m dont les racines sont précisément les abscisses des points formant le diviseur E, et l'on code les ordonnées dans un polynôme d'interpolation $v(x)$. On obtient ainsi le résultat suivant :

Proposition 2 *Soit $\mathcal{C}$ une courbe hyperelliptique de genre g représentée par un modèle imaginaire $y^2 + h(x)y = f(x)$. Les éléments de la jacobienne de $\mathcal{C}$ sont en bijection avec les couples de polynômes $\langle u(x), v(x)\rangle$ tels que u est unitaire,* $\deg v < \deg u \leqslant g$, *et $u | v^2 + hv - f$.*

Notons que cette représentation respecte la notion de corps de définition : un élément de la jacobienne est définie sur un corps k si et seulement si les polynômes u et v sont définis sur k. L'algorithme de Cantor [14], qui suit les algorithmes de composition et de réduction des formes quadratiques est un algorithme permettant de calculer efficacement dans la jacobienne : étant donnés deux éléments représentés sous forme de Mumford, on peut calculer la représentation de leur somme à l'aide de quelques opérations sur les polynômes.

Les coniques et les courbes elliptiques

Les courbes de genre 0 sont les coniques dont l'arithmétique est très simple. Ensuite viennent les courbes elliptiques qui sont les courbes de genre 1. Dans ce cas la jacobienne est isomorphe à la courbe elle-même, ce qui simplifie bien les choses ; en particulier la loi de groupe est définie directement sur les points de la courbe. Les courbes hyperelliptiques peuvent être vues comme le cas le plus simple après les courbes elliptiques.

Les points de torsion

Soit n un entier supérieur ou égal à 2. Le sous-groupe des éléments de n-torsion de la jacobienne est l'ensemble des éléments définis sur le clôture algébrique qui s'annulent lorsqu'on les multiplie par n. On note cet ensemble $\mathrm{Jac}(\mathcal{C})[n]$. Il s'agit d'un sous-groupe fini, dont la structure est donnée par le théorème suivant.

Théorème 3 *Soit $\mathcal{C}$ une courbe de genre g définie sur un corps k. Si n est un entier premier à la caractéristique de k, alors*

$$\mathrm{Jac}(\mathcal{C})[n] \cong (\mathbb{Z}/n\mathbb{Z})^{2g}.$$

De plus, si p^t est une puissance de la caractéristique de k,

$$\mathrm{Jac}(\mathcal{C})[p^t] \cong (\mathbb{Z}/p^t\mathbb{Z})^r,$$

où r est un entier compris entre 0 et g qui ne dépend pas de t. Il est appelé le p-rang de la jacobienne de $\mathcal{C}$.

1.2.3 Problèmes algorithmiques

Les questions algorithmiques liées à l'utilisation des jacobiennes de courbes en cryptographie peuvent avoir deux finalités : soit on se place du côté de l'utilisateur, et l'on cherche à construire des paramètres et à effectuer les opérations cryptographiques le plus rapidement possible ; soit on se place du côté de l'attaquant et l'on cherche à accélérer les algorithmes qui permettent de casser le système, en particulier le logarithme discret. Ce dernier point de vue est important car les meilleurs attaques connues fixent les tailles de paramètres nécessaires pour garantir une sécurité donnée. L'algorithmique du côté constructif est moins cruciale d'un point de vue académique, mais tout aussi importante en pratique, car c'est bien l'efficacité d'un système qui va faire sa renommée et inciter les gens à l'utiliser.

Étude de la sécurité

La sécurité des cryptosystèmes à base de courbes que l'on étudie réside en premier lieu sur la difficulté présumée du problème du logarithme discret. Celui-ci peut s'énoncer dans n'importe quel groupe cyclique.

Définition 4 *Soit G un groupe cyclique noté additivement, engendré par g d'ordre N. Le problème du logarithme discret dans G est le suivant : connaissant g et N, et étant donné un élément h dans G, trouver un entier x tel que $h = xg$.*

On peut définir des variantes de ce problème, où l'on ne suppose pas que g engendre tout le groupe, ou bien que l'ordre du groupe n'est pas connu, mais cette définition est en général suffisante pour couvrir les applications pratiques.

Historiquement, le premier algorithme cryptographique s'appuyant sur le problème du logarithme discret est le protocole de Diffie et Hellman [35] que nous rappelons maintenant. Il s'agit d'un protocole d'échange de clef : deux protagonistes, Alice et Bob, veulent communiquer de manière sécurisée, alors qu'ils ne disposent pas de secret commun pouvant leur servir de clef pour un algorithme de chiffrement classique. C'est typiquement le genre de situation qui se produit lors du paiement en ligne : je souhaite que mon numéro de carte bleue ne circule pas en clair sur Internet, mais je veux tout de même payer sur un site que je n'ai jamais visité avant, et donc avec lequel je ne partage aucun secret qui puisse servir de clef.

Le protocole de Diffie-Hellman fonctionne de la manière suivante : les participants se mettent d'accord sur un groupe G et un générateur g d'ordre N dans lequel le logarithme discret est supposé difficile. Ces données n'ont aucune raison d'être secrète et peuvent en fait être définies par des standards. Alice et Bob suivent alors la procédure suivante : Alice tire un entier aléatoire x_A entre 0 et N, calcule $h_A = x_A g$ et envoie h_A à Bob. De manière symétrique, Bob tire un entier x_B aléatoire, calcule $h_B = x_B g$ et envoie h_B à Alice. Ensuite Alice calcule $x_A h_B$ et Bob calcule $x_B h_A$. Ces deux quantités sont identiques, puisqu'elles sont égales à $x_A x_B g$, et c'est cette quantité qui va constituer la nouvelle clef secrète commune à Alice et Bob. Un espion qui écoute la ligne entre Alice et Bob voit passer seulement h_A et h_B, et son but est de calculer $x_A x_B g$. Ce problème, appelé problème de Diffie-Hellman est lié au problème du logarithme discret : si l'espion sait résoudre des logarithmes discrets dans G, il peut retrouver x_A à partir de h_A et des paramètres du système, et donc dispose d'autant d'information qu'Alice. Il n'y a alors aucune sécurité. La réciproque est plus subtile. Un résultat de Maurer et Wolf [119] affirme que si l'on sait trouver une courbe elliptique auxiliaire adaptée au cardinal de G, on peut résoudre le logarithme discret dans G à l'aide d'un oracle résolvant le problème de Diffie-Hellman dans G. Ainsi, sous réserve de la connaissance d'une courbe elliptique auxiliaire adaptée à G, le problème de Diffie-Hellman est polynomialement équivalent au problème du logarithme discret. Maintenant, si l'on ne connaît pas de courbe auxiliaire, on peut tenter d'en trouver une. Pour ce problème, il n'existe pas d'algorithme polynomial connu à ce jour. Cependant sous des heuristiques très raisonnables, pour tout G, on peut trouver en temps sous-exponentiel une courbe elliptique auxiliaire qui va permettre la réduction en temps sous-exponentiel. Dans un contexte où le meilleur algorithme connu pour le logarithme discret est de complexité sous-exponentielle (comme c'est le cas pour les groupes multiplicatifs de corps finis), ce résultat ne dit pas grand chose. Par contre, dans le contexte des courbes elliptiques ou des courbes de genre 2, les groupes sont plus petits et il n'est effectivement pas très difficile de construire des courbes auxiliaires qui permettent de réduire un problème à l'autre en un nombre modéré d'appels à l'oracle. Muzereau, Smart et Vercauteren [133] ont d'ailleurs proposés de telles courbes auxiliaires pour les courbes elliptiques des standards.

En fait, la notion de sécurité que l'on souhaite dans un protocole de type Diffie-Hellman est plus forte que juste l'impossibilité de calculer la clef de session. La clef commune obtenue se doit d'être indistinguable d'un élément purement aléatoire même pour quelqu'un qui a vu passer toutes les communications. En termes formels, on traduit cela par l'hypothèse qu'il n'existe pas d'algorithme polynomial probabiliste qui parvienne à distinguer de manière significative un triplet de la forme $(x_A g, x_B g, x_A x_B g)$ d'un triplet $(x_A g, x_B g, x_C g)$ où x_C est aléatoire uniforme dans $[0, N-1]$. Le problème associé s'appelle le problème de Diffie-Hellman

décisionnel. C'est un problème qui est a priori plus facile que la version calculatoire décrite ci-dessus. Et en effet, il existe des groupes pour lesquels on connaît des algorithmes efficaces pour résoudre le problème décisionnel, sans pour autant que l'on sache résoudre la version calculatoire [95].

Dans ce mémoire, nous ne nous intéresserons qu'au problème du logarithme discret dans les jacobiennes de courbes. Comme indiqué plus haut, cela couvre aussi essentiellement le problème de Diffie-Hellman dans sa version calculatoire. Par contre, nous ne discuterons pas plus du problème décisionnel sur lequel peu de choses sont connues.

Les approches pour attaquer le problème du logarithme discret sur les courbes peuvent se classer dans les grandes catégories suivantes :

- Les attaques génériques : il s'agit d'algorithmes qui n'utilisent pas spécifiquement le groupe dans lequel on calcule. On peut les définir dans un modèle boîte noire dans lequel chaque opération de groupe est effectuée via un oracle.
- Les attaques utilisant le couplage : pour certaines courbes très particulières (en particulier les courbes supersingulières), la jacobienne peut être équipée d'une forme bilinéaire non-dégénérée calculable rapidement. On peut utiliser celle-ci pour se ramener à un problème de logarithme discret dans un corps fini pour lequel il existe des algorithmes sous-exponentiels.
- Les attaques par relèvement : on peut tenter de relever le problème du logarithme discret en considérant des courbes sur les p-adiques ou sur un corps de nombres. Sauf dans un cas très particulier, ces attaques n'ont pour l'instant abouti à rien de probant.
- Les attaques par calcul d'index en genre grand : pour les courbes dont le genre est suffisamment grand par rapport au corps de base, on peut concevoir un algorithme de logarithme discret sous-exponentiel similaire à ceux qui existent pour les corps finis. Bien que ne s'appliquant pas telles quelles aux courbes elliptiques, l'étude de ces approches a permis de grandement réduire le champ des courbes utilisables en cryptographie.
- Les attaques par descente de Weil : dans certains contextes, on peut ramener un problème de logarithme discret sur une courbe elliptique à un problème de logarithme discret dans la jacobienne d'une courbe de grand genre. Les attaques précédentes peuvent alors appliquées.

Nous avons contribué sur les deux deniers aspects, ce qui sera résumé dans le chapitre 2 de ce mémoire.

Construction de paramètres

La première étape dans la création d'un cryptosystème est de choisir un corps fini et une courbe définie sur ce corps. La principale difficulté ici est de garantir que le cardinal de la jacobienne contienne un grand facteur premier. La difficulté du logarithme discret est en effet directement liée à la taille du plus grand facteur premier (cf ci-dessous). Idéalement, donc, on demande que le cardinal soit un nombre premier.

Soit $\mathbb{F}_q$ un corps fini à q éléments, et soit $\mathcal{C}$ une courbe sur $\mathbb{F}_q$. L'outil principal pour les études de cardinalité est l'endomorphisme de Frobenius π qui envoie un point $P = (x, y)$ de $\mathcal{C}$ défini sur $\overline{\mathbb{F}_q}$ sur $\pi(P) = (x^q, y^q)$ qui est aussi un point de $\mathcal{C}$. Les points fixes de π sont exactement les points définis sur $\mathbb{F}_q$. Cet endomorphisme de Frobenius s'étend facilement aux diviseurs puis aux éléments de la jacobienne, et l'on note toujours π ces extensions. Un des résultats majeurs du XX$^{\text{e}}$ siècle est que π vérifie une équation polynomiale dans l'anneau des

endomorphismes de la forme

$$\chi(t) = t^{2g} - s_1 t^{2g-1} + s_2 t^{2g-2} + \cdots + q^{g-2} s_2 t^2 - s_1 q^{g-1} t + q^g,$$

où les s_i sont des entiers, et dont les racines complexes sont toutes de module $\sqrt{q}$. Le cardinal de la jacobienne est alors donné par $\chi(1)$, et les bornes sur les racines fournissent les inégalités :

$$(\sqrt{q} - 1)^{2g} \leqslant \#\mathrm{Jac}(\mathcal{C}) \leqslant (\sqrt{q} + 1)^{2g}.$$

À partir de cet encadrement, on comprend l'intérêt qu'il peut y avoir à travailler avec des courbes de genre grand : en effet si l'on souhaite disposer d'un groupe d'une taille donnée, plus le genre est grand plus on pourra avoir un corps de base petit. Par exemple pour avoir un groupe de cardinal environ 2^{200}, avec une courbe elliptique il faudra choisir un corps de base ayant environ 2^{200} éléments, alors qu'un corps de base ayant environ 2^{100} éléments suffira si l'on considère une courbe de genre 2. En contrepartie, la complexité de la loi de groupe croît avec le genre, et trouver le meilleur compromis entre la taille du corps de base et la complexité de la loi de groupe est une question encore ouverte.

Les bornes de Hasse-Weil sur le cardinal de la jacobienne ne donnent qu'un ordre de grandeur. Cela ne suffit pas pour les applications cryptographiques. Voici une liste des stratégies possibles pour obtenir le cardinal exact.

- Les algorithmes de complexité exponentielle : mis à part les algorithme naïfs qui énumèrent les éléments de la jacobienne, on peut utiliser des algorithmes génériques «en racine carrée», c'est-à-dire des algorithmes qui utilisent uniquement la structure abstraite de groupe. Un autre algorithme de complexité exponentielle est le calcul de l'opérateur de Cartier-Manin, ce qui ne donne qu'une information partielle, mais parfois très utile.
- La méthode de la multiplication complexe (CM) : il s'agit de construire des courbes pour lesquelles le nombre de points de la jacobienne est relativement facile à calculer. Les courbes ainsi construites sont très particulières, si bien que l'on peut craindre d'avoir des attaques sur le logarithme discret qui soient spécifiques à ces courbes.
- L'algorithme de Schoof et ses variantes : pour cette approche, on calcule le nombre de points modulo de nombreux petits premiers en considérant les éléments de torsion. On reconstruit ensuite par le théorème des restes chinois. Historiquement, cet algorithme fut le premier à fournir une complexité polynomiale.
- Les algorithmes p-adiques : il s'agit de différentes approches (principalement dues à Satoh et Kedlaya) où l'on relève la courbe sur un corps p-adique afin d'y appliquer des résultats d'analyse p-adique. Là encore on obtient une complexité polynomiale, mais seulement lorsque la caractéristique du corps de base est petite.

L'étude et l'amélioration de ces méthodes constituent une grande part de ce mémoire. Celles-ci seront donc détaillées au fil du chapitre 3. Notons aussi que dans le cas des courbes elliptiques le problème du comptage de points est essentiellement résolu, du moins pour les tailles cryptographiques.

Efficacité du système

Dans tous les algorithmes cryptographiques à clef publique s'appuyant sur le problème du logarithme discret, l'opération la plus coûteuse est la multiplication d'un point du groupe par un scalaire. Ceci est vrai pour les algorithmes de chiffrement, de signature, d'échange de clefs, et de toutes les variantes ; en effet, cette opération de multiplication par un scalaire est

celle qui produit précisément un problème de logarithme discret (si le scalaire est secret), et va donc garantir la sécurité.

Dans le cas qui nous intéresse, la loi de groupe est complexe puisqu'il s'agit de l'addition de classes de diviseurs dans la jacobienne. Cela dit, une fois mise à plat, on peut voir la loi de groupe comme donnée par des fractions rationnelles en les coordonnées des éléments à additionner. La complexité (au sens du nombre d'opérations nécessaires à l'évaluation de ces fractions rationnelles) dépend fortement du choix des coordonnées choisies, et même pour des coordonnées fixées, déterminer la meilleure chaîne d'opérations est hors d'atteinte avec les techniques actuelles. On doit donc s'appuyer sur des astuces algorithmiques et des heuristiques afin de tenter de diminuer le coût de la loi de groupe. De plus, le problème n'admet pas de réponse universelle : cela dépend énormément du contexte d'utilisation des formules. Par exemple, si l'on utilise du matériel dédié, les opérations dans un corps fini de caractéristique 2 sont bien plus faciles que dans un corps premier. En revanche sur un ordinateur personnel, ce sont les corps premiers qui sont les plus rapides. En fonction des coûts relatifs des opérations dans le corps fini, et en particulier de l'inversion par rapport à la multiplication, on doit aussi choisir ou non des coordonnées projectives où l'on troque les inversions contre un certain nombre de multiplications. Un autre aspect est la lutte contre les attaques utilisant des fuites dues à l'unité de calcul pendant l'opération cryptographique, typiquement la variation de consommation électrique lors du calcul, mais aussi les problèmes de prédiction de branches et autres effets de cache dans les processeurs modernes. Certaines formules se prêtent mieux à la lutte contre ce type de fuites, et on peut les préférer, même s'il y a un surcoût qui sera compensé par l'économie sur les contre-mesures à mettre en œuvre avec d'autres formules.

Par ailleurs, tous les protocoles ne nécessitent pas le même type de calcul : dans certains cas, le scalaire par lequel on multiplie les points est toujours le même, dans d'autres cas on va toujours multiplier simultanément deux points par le même scalaire. Une formule qui a priori ne paraissait pas intéressante pourra donc se révéler imbattable pour un contexte particulier. Dans le cas des courbes elliptiques, la quantité de formules disponibles dans la littérature est assez impressionnante (plusieurs douzaines), et le domaine est encore actif. Pour le cas des courbes de genre supérieur, la situation n'en est pas encore là, mais on dispose déjà de nombreux choix, et il y a encore un certain potentiel d'améliorations.

Cette étude de l'efficacité des systèmes est cruciale du point de vue pratique. Il est des contextes où la vitesse n'est pas un problème : typiquement, lorsque l'on butine sur Internet, notre ordinateur personnel n'a aucun mal à gérer les calculs cryptographiques. Seule une petite fraction de la puissance de calcul sera requise, et les économies de temps ou d'énergie seront complètement négligeables. À l'autre bout du tuyau, la situation est bien différente : un serveur est soumis à des centaines voire des milliers de requêtes simultanées qui vont lui demander à chaque fois une opération cryptographique. Si l'on parle d'un serveur bancaire où les contenus échangés sont très faible en quantité, mais où la sécurité est cruciale, la part de la couche cryptographique dans l'occupation des processeurs (ou des coprocesseurs dédiés) va être prépondérante. Dans ce genre de contexte, gagner 20% sur le temps de calcul signifie que l'on n'achètera que 4 serveurs au lieu de 5, et que l'on gagne autant sur la consommation électrique et la climatisation. Un autre contexte où la vitesse d'exécution est critique est le monde embarqué : si la puissance de calcul est limitée, le temps d'exécution peut redevenir problématique (par exemple sur une carte à puce) ; le plus souvent il s'agira aussi de gérer la faible ressource en énergie. Par exemple les passeports biométriques tirent leur courant du mouvement de l'antenne qui y est moulée à l'intérieur d'un champ magnétique : on parle alors de milli-Watts, voire de micro-Watts disponibles.

Ce thème de l'efficacité sera évoqué dans ce mémoire à la section 3.7.2, où nous parlerons de nouvelles formules que nous avons proposées pour les courbes de genre 2, mais cela reste pour l'instant une petite partie de notre activité de recherche.

Chapitre 2

Cryptanalyse de certains cryptosystèmes (hyper-)elliptiques

Ce chapitre est consacré à l'étude du problème du logarithme discret dans les jacobiennes de courbes. On y décrit les différents algorithmes connus, en insistant sur ceux que nous avons nous-mêmes développés ou auxquels nous avons contribués.

Après un survol rapide des algorithmes génériques, des attaques par relèvement et de l'attaque utilisant les couplages, on y trouve une description d'un algorithme de calcul d'index adapté aux courbes de genre pas trop grand [64], ainsi que sa variante utilisant des «large primes» [77], développée en collaboration avec Thomé, Thériault et Diem. Toujours sur le calcul d'index, avec Enge [41], nous avons exhibé une classe de courbes pour lesquelles on a un algorithme de complexité sous-exponentielle en $L(1/3)$.

Une deuxième classe d'algorithmes sur lesquels nous nous étendrons dans ce chapitre concerne la descente de Weil : avec Heß et Smart [70], nous avons proposé la première attaque sur des courbes elliptiques en ramenant le logarithme discret à un problème dans une jacobienne de courbe hyperelliptique. Nous avons ensuite proposé une approche plus directe [62], s'appuyant directement sur un calcul d'index dans une variété abélienne.

2.1 Généralités sur le logarithme discret et le calcul d'index

Pour évaluer la sécurité d'un cryptosystème, il est usuel de calculer le temps que nécessiterait une attaque utilisant le meilleur algorithme connu. Il nous paraît tout d'abord utile de donner quelques ordres de grandeurs sur ce qu'est capable de faire un ordinateur d'aujourd'hui. Les processeurs ont actuellement une fréquence de quelques GHz, ce qui signifie qu'ils peuvent effectuer quelques milliards d'opérations élémentaires par secondes, disons $5\ 10^9$, ce qui fait environ $1,5\ 10^{17}$ opérations par an. Si un attaquant dispose d'un million de processeurs qu'il peut alimenter pendant 1 an, il peut donc effectuer environ $1,5\ 10^{23}$ opérations élémentaires. En prenant un petit peu de marge, on obtient le premier niveau de sécurité accepté en général, soit 2^{80} opérations. Il est clair que dans ce genre de calcul, on n'est pas à une petite constante près.

Un niveau de sécurité «standard» qui est plus raisonnable est 2^{128} opérations. Celui-ci nous met à l'abri de toute attaque s'appuyant sur les algorithmes connus. Prendre un niveau de sécurité supérieur peut se justifier si l'on souhaite se prémunir contre la découverte d'éventuelles faiblesses dans le système (mais si le système s'avère finalement très faible, cela peut s'avérer insuffisant).

2.1.1 Algorithmes génériques

La première famille d'algorithmes pour s'attaquer au problème du logarithme discret est la famille des algorithmes dits génériques. Ceux-ci sont des algorithmes qui fonctionnent quel que soit le groupe considéré pourvu que l'on ait à sa disposition des algorithmes pour effectuer quelques opérations de base, comme calculer la somme de deux éléments, calculer l'opposé d'un élément, ou bien tirer un élément aléatoire dans le groupe. On peut bien évidemment formaliser ceci à l'aide d'oracles, mais cette formalisation est surtout nécessaire lorsque l'on souhaite travailler sur des bornes inférieures.

Fixons donc le cadre dans lequel nous allons décrire brièvement ces algorithmes génériques. Soit G un groupe fini. Soit P un élément de G d'ordre *connu* N et soit Q un élément du sous-groupe engendré par P. Le but est de trouver un entier x tel que $Q = x \cdot P$.

La première remarque, connue sous le nom d'algorithme de Pohlig-Hellman [137] est que l'on peut se ramener à des calculs de logarithmes discrets dans des sous-groupes d'ordre premier. En effet, si N est sans facteur carré, il s'agit d'un simple lemme chinois qui permet de calculer x à partir de ses valeurs modulo chaque facteur premier de N. Et si N a des facteurs multiples, un relèvement du type Hensel permet de trouver x modulo des puissances d'un nombre premier.

Nous supposerons donc le plus souvent dans la suite que G est un groupe cyclique d'ordre N premier.

Pour aller au-delà de l'algorithme naïf en $O(N)$ opérations qui teste toutes les valeurs possibles de x, on peut appliquer un principe de compromis temps-mémoire nommé «pas de bébé, pas de géant» (aussi appelée méthode de Shanks [150]). Soit B un entier valant environ $\sqrt{N}$. La division euclidienne de x par B donne $x = x_0 + Bx_1$, où x_0 est un entier entre 0 et $B-1$, et x_1 est borné par environ $\sqrt{N} \approx N/B$. L'équation $Q = x \cdot P$ se récrit alors $Q - x_0 \cdot P = x_1 \cdot (B \cdot P)$. Il est possible d'énumérer tous les membres gauches possibles de cette équation en environ $\sqrt{N}$ opérations, et de même pour tous les membres de droite possibles. Il reste alors à chercher un élément commun dans ces deux listes, ce qui se fait grâce à des méthodes classiques de tri/recherche en temps $O(\sqrt{N} \log N)$. Ainsi, il existe un algorithme

qui calcule le logarithme discret en $O(\sqrt{N})$ opérations dans le groupe. En général, seules ces dernières sont comptées, et non la partie tri-recherche qui ne fait pas appel au modèle «groupe générique» et est en général très rapide en pratique, malgré le facteur logarithmique supplémentaire.

Le problème majeur de l'algorithme que l'on vient de voir est qu'il nécessite le stockage de tous les membres gauches potentiels (ou tous les membres droites, si l'on préfère), ce qui fait une complexité en espace de $O(\sqrt{N})$ éléments du groupe.

Si l'on s'autorise un algorithme probabiliste, la complexité en mémoire peut-être réduite, tout en conservant la même complexité en temps, grâce à des algorithmes du type «méthode Rho». Nous renvoyons à [25, Chapitre 19] pour une description de ces variantes.

Les algorithmes génériques s'appliquent évidemment au problème du logarithme discret dans les jacobiennes de courbes. De plus dans le cas très important des courbes elliptiques et hyperelliptiques, calculer l'opposé d'un point est essentiellement gratuit. Il est alors assez facile de modifier les algorithmes pour diviser le temps de calcul par $\sqrt{2}$. Plus généralement, si le groupe est équipé d'un automorphisme d'ordre ℓ que l'on peut calculer très efficacement par rapport à la loi de groupe, on peut espérer gagner un facteur $\sqrt{\ell}$.

2.1.2 Attaque MOV

Les jacobiennes de courbes (et plus généralement les variétés abéliennes) peuvent être munies d'une forme bilinéaire non-dégénérée, appelée le couplage de Weil. La définition et les première propriétés peuvent se trouver par exemple dans [153]. Nous les rappelons brièvement.

Soit $\mathcal{C}$ une courbe sur un corps fini $\mathbb{F}_q$. Soit P et Q deux éléments d'ordre N de la jacobienne de $\mathcal{C}$ sur une extension $\mathbb{F}_{q^k}$ de $\mathbb{F}_q$. Le couplage de Weil de P et de Q est défini de la manière suivante : soient D_P et D_Q deux diviseurs à support disjoints tels que D_P soit dans la même classe que P et D_Q soit dans la même classe que Q. Comme D_P est d'ordre N, il existe une fonction f_P telle que $\mathrm{div} f_P = ND_P$. De manière similaire, on peut définir une fonction f_Q telle que $\mathrm{div} f_Q = ND_Q$. Le couplage de Weil est alors défini par

$$e_N(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)}.$$

Le fait que la quantité définie a bien un sens est due au choix de prendre D_P et D_Q à supports disjoints. On peut aussi montrer les résultats suivants : la définition de e_N ne dépend pas des représentants D_P et D_Q choisis ; la fonction ainsi définie est une forme bilinéaire, antisymétrique ; la valeur de $e_N(P,Q)$ est une racine N-ième de l'unité dans $\mathbb{F}_{q^k}$; considérée sur l'ensemble des points de N-torsion (sur une clôture algébrique de $\mathbb{F}_q$), le couplage de Weil est non-dégénéré.

Si N est premier à la caractéristique, l'ensemble des points de N-torsion sur une clôture algébrique est isomorphe (en tant que groupe) à $(\mathbb{Z}/N\mathbb{Z})^{2g}$. On peut montrer que le couplage de Weil est trivial lorsqu'on l'applique à deux fois le même point, si bien que pour obtenir de l'information via le couplage de Weil, il est nécessaire de coupler deux points linéairement indépendants.

Il existe un algorithme très efficace pour calculer le couplage de Weil de deux points, pourvu que les coordonnées de ceux-ci soit dans une extension pas trop grande du corps de base. Cet algorithme dû à Miller [128, 129] consiste à calculer les fonctions f_P et f_Q de proche en proche par une méthode d'exponentiation binaire. Comme le degré de ces fonctions est de

complexité exponentielle, on ne peut pas les stocker sous forme développée, mais il n'y a pas de problème pour les évaluer respectivement en D_Q et D_P au fur et à mesure du calcul, si bien que l'on ne manipule jamais l'expression complète d'une fonction.

Dans une situation typique du problème du logarithme discret, les données sont deux points P et Q d'ordre divisible par N, et qui sont linéairement dépendants. De plus, en général P est un générateur de la jacobienne, si bien qu'il n'existe pas de points de N-torsion indépendant de P et Q qui soit défini sur le corps de base. Il est donc nécessaire de passer dans une extension. Pour une courbe aléatoire, ce degré est très grand (de l'ordre de N), si bien que l'on n'a aucun espoir de calculer un couplage non-trivial. Dans le cas où l'on peut trouver un point de N-torsion R indépendant de P et Q dans une extension de degré k suffisamment petit, on peut monter l'attaque suivante, due à Menezes, Okamoto et Vanstone [120]. On calcule $e_N(P,R)$ et $e_N(Q,R)$. Par linéarité, ces deux quantités forment un problème de logarithme discret dans le corps fini $\mathbb{F}_{q^k}$ dont la solution est la même que pour le problème du logarithme discret initial entre P et Q. Ainsi, on appliquant un algorithme sous-exponentiel pour résoudre le logarithme discret dans $\mathbb{F}_{q^k}$, on obtient un algorithme global qui peut s'avérer plus rapide qu'un algorithme générique dans la jacobienne.

Il existe des familles de courbes pour lesquelles k est petit, et ces courbes doivent donc être évitées. D'un autre côté, si l'on contrôle bien le degré k (ni trop gros, ni trop petit), on peut garder un couplage non-trivial calculable efficacement, sans pour autant compromettre la sécurité. Partant de cette primitive, de nombreux protocoles cryptographiques ont été inventés. Des centaines d'articles forment ainsi ce qu'on peut appeler la cryptographie à base de couplage [25, Chapitre 24].

2.1.3 Attaques par relèvement

Dans le cadre des corps finis, les algorithmes sous-exponentiels (du type calcul d'index) font appel de manière plus ou moins claire à un relèvement vers $\mathbb{Z}$ ou $\mathbb{F}_p[x]$. Ceci permet en effet de retrouver une notion de «taille» des éléments qui est essentiellement absente dans un corps fini.

Partant de ce constat, de nombreuses approches ont été tentées pour attaquer le logarithme discret dans les jacobiennes de courbes en utilisant un passage par les rationnels [94, 152], les corps de fonctions ou les p-adiques. Pour l'instant, toutes les tentatives ont échoué. Seul un cas extrêmement particulier a été cassé par relèvement p-adique, à savoir le cas où l'on veut résoudre un problème du logarithme discret dans un sous-groupe d'ordre égal à la caractéristique du corps de base [155, 148, 142, 140].

Nous ne nous étendrons pas plus sur ces approches infructueuses, même si leur étude est cruciale pour acquérir de la confiance en les cryptosystèmes à base de courbes.

2.2 Logarithme discret sur les courbes de genre supérieur ou égal à 3

Dans cette section, nous allons donner un survol des algorithmes de calcul d'index pour le logarithme discret dans les jacobiennes, en insistant plus particulièrement sur le comportement de ces algorithmes lorsque le genre est petit.

2.2.1 Bref historique

La première attaque sous-exponentielle contre le logarithme discret dans les jacobiennes de courbes a été portée par Adleman, DeMarrais et Huang en 1994 [2]. L'idée majeure est l'introduction d'une notion de friabilité sur les diviseurs : les diviseurs qui vont jouer le rôle des nombres premiers dans un calcul d'index classique sont les diviseurs formés d'une unique orbite sous l'action du groupe de Galois ; dit autrement, un diviseur premier est un diviseur formé d'un point de la courbe défini sur une extension de degré k du corps de base (et pas dans une sous-extension), ainsi que de ses k conjugués. Les diviseurs friables sont alors les diviseurs qui s'écrivent comme somme de diviseurs premiers de petit degré. À partir de cette notion de friabilité, Adleman, DeMarrais et Huang proposent de fabriquer des relations entre les diviseurs premiers au sein de la jacobienne. On considère le diviseur d'une fonction sur la courbe. Si celui-ci est friable, on peut en déduire qu'une certaine combinaison d'éléments de la jacobienne va être nulle. Ces éléments correspondent précisément aux diviseurs premiers de degré petit. En combinant suffisamment de relations, on peut en déduire par de l'algèbre linéaire les logarithmes de ces éléments. Nous passons sous silence les détails, et en particulier la manière dont à partir des logarithmes d'une partie des éléments on peut en déduire les logarithmes de tous les éléments. La complexité obtenue par Adleman, DeMarrais et Huang est de la forme $L_{q^g}(1/2, c)$, pour une certaine constante c, où L est la fonction sous-exponentielle classique qui intervient dans les questions de friabilité :

$$L_N(\alpha, c) = \exp\left(c \log(N)^\alpha \log\log(N)^{1-\alpha}\right).$$

Précisons toutefois que la complexité obtenue était seulement heuristique, et restreinte au cas hyperelliptique. Par ailleurs, une contrainte fondamentale est que le genre est grand, c'est à dire $g > \log q$.

La méthode d'Adleman, DeMarrais et Huang a été amélioré aussi bien d'un point de vue pratique que théorique par diverses personnes [48, 132, 43, 39, 60]. En 1999, lorsque nous avons commencé à travailler sur le sujet, on considérait qu'il était prudent de se contenter de courbe de genre au plus 6 pour un usage cryptographique.

Nos travaux de 2000 [64] et les améliorations qui suivirent [166, 77, 32, 34] démontrèrent qu'en fait, dès le genre 3 il existe une attaque du type calcul d'index qui sera plus efficace que les attaques génériques, et que de fait les tailles des paramètres doivent être augmentés en conséquence afin de maintenir un niveau de sécurité satisfaisant. Nous donnons une idée des algorithmes mis en jeu ci-dessous.

Parallèlement à ces études à genre fixé petit, des progrès ont aussi été effectué du côté théorique, lorsque le genre tend vers l'infini. En particulier, une complexité en $L(1/2)$ rigoureuse est désormais prouvée, pour toutes les courbes [27, 87]. Bien évidemment, on conserve une condition du type $g > \log q$. De plus des classes de courbes ont été découvertes pour lesquelles la complexité peut être ramenée à $L(1/3)$ (heuristiquement). Nous évoquons aussi ces travaux récents ci-dessous.

2.2.2 Calcul d'index à genre fixé

Le cadre que nous allons nous fixer est le suivant : soit $\mathcal{C}$ une courbe hyperelliptique de genre g définie sur un corps fini $\mathbb{F}_q$ à q éléments. On suppose que le cardinal de la jacobienne de $\mathcal{C}$ est un nombre premier N. Soit D_1 un générateur de la jacobienne, et soit D_2 un autre élément de $\mathrm{Jac}(\mathcal{C})$. On cherche le logarithme discret de D_2 en base D_1.

Suivant l'idée d'Adleman-DeMarrais-Huang, on introduit une base de facteurs $\mathcal{F}$ formée de tous les diviseurs réduits de poids 1 que l'on manipule sous forme de Mumford :

$$\mathcal{F} = \{\langle x - x_0, y_0\rangle, \text{ tels que } (x_0, y_0) \in \mathcal{C}\}.$$

Plutôt que de chercher à écrire des diviseurs principaux sur cette base, nous préférons étudier la friabilité des diviseurs réduits :

Définition 5 *Un diviseur réduit D de* $\mathrm{Jac}(\mathcal{C})$ *est dit friable s'il est égal à la somme d'au plus g éléments de la base de facteurs $\mathcal{F}$.*

Cette notion est intéressante car elle s'accompagne d'algorithmes efficaces. En effet si la représentation de Mumford de D est $\langle u(x), v(x)\rangle$, le diviseur D est friable si et seulement si le polynôme $u(x)$ est scindé ; la décomposition correspondante est alors

$$D = \sum_{u(x_0)=0} \langle x - x_0, v(x_0)\rangle.$$

Un algorithme du type calcul d'index pour le logarithme discret découle de cette notion de friabilité. On forme des combinaisons linéaires aléatoires de D_1 et D_2, et l'on teste si les diviseurs obtenus sont friables. Chaque fois qu'une combinaison linéaire donne un diviseur friable on obtient ce qu'on appelle une relation qui est alors stockée comme un ligne d'une matrice dont les colonnes sont étiquetées par les éléments de la base de facteurs. Notons $R_i = \alpha_i D_1 + \beta_i D_2$ la i-ème relation découverte. Comme R_i est friable, il s'écrit comme une somme d'au plus g éléments de $\mathcal{F}$. Numérotons les éléments de $\mathcal{F}$: $\mathfrak{p}_1$, $\mathfrak{p}_2$, …, $\mathfrak{p}_k$. Ainsi $R_i = \sum_{1\leqslant i\leqslant k} a_{ij}\mathfrak{p}_j$, où les a_{ij} sont des entiers, tous nuls sauf au plus g d'entre eux. Lorsque l'on a construit suffisamment de relations (c'est-à-dire plus que le cardinal de la base de facteurs), il existe une combinaison linéaire des lignes de la matrice (a_{ij}) qui s'annule. Ainsi il existe un vecteur γ_i tel que $\sum_i \gamma_i R_i = 0$, ce qui se récrit $\sum_i \alpha_i\gamma_i D_1 = -\sum_i \beta_i\gamma_i D_2$. Si $\sum_i \beta_i\gamma_i$ est inversible modulo N, on obtient le logarithme discret cherché :

$$\log_{D_1}(D_2) \equiv -\frac{\sum_i \alpha_i\gamma_i}{\sum_i \beta_i\gamma_i} \mod N.$$

Nous allons effectuer l'analyse de cet algorithme à g fixé. Si l'on ajoute un multiple aléatoire de D_1 à un multiple aléatoire de D_2, on obtient un élément aléatoire de la jacobienne, que l'on espère friable. La quantité d'éléments friables dans la jacobienne peut être évaluée assez facilement : à tout ensemble de g éléments de $\mathcal{F}$, on peut associer le diviseur friable, somme de ces composants. Par le théorème de Weil, le cardinal de $\mathcal{F}$ est environ q. Le nombre d'ensembles de g éléments est donc de l'ordre de $q^g/g!$. On peut montrer que le nombre de collisions est un ordre de grandeur en dessous, si bien que le nombre de diviseurs friables est aussi de l'ordre de $q^g/g!$. La probabilité qu'une combinaison linéaire aléatoire de D_1 et D_2 est friable tend donc vers $1/g!$ quand q tend vers l'infini. Le calcul d'une combinaison linéaire aléatoire ainsi que le test de friabilité prennent un temps polynomial en $\log q$, et il faudra répéter ces opérations environ $g!q$ fois en moyenne afin d'avoir assez de relations. Ensuite il s'agit d'un calcul d'un vecteur du noyau d'une matrice creuse, ce qui se fait en temps quadratique, grâce par exemple à l'algorithme de Wiedemann, soit $O(q^2)$. Finalement, la complexité totale de l'algorithme est en $O(q^2)$, avec une dépendance en g qui est en $g!$, mais qui est absorbée par le $O()$, puisque g est supposé constant.

Cet algorithme est tout à fait pratique ; nous l'avons implanté et calculé des logarithmes discrets dans des jacobiennes de courbes de genre 6 d'ordre environ 10^{40} en un temps bien moindre que si un algorithme générique avait été utilisé.

2.2.3 Variante avec deux “large primes”

Dans l'algorithme de calcul d'index précédent, il est frustrant d'avoir deux phases très déséquilibrées : l'algèbre linéaire est en $O(q^2)$ alors que la recherche de relations est en $O(q)$. Une première approche pour améliorer cette situation a été trouvée par Harley. Celle-ci fut ensuite étendue en une variante single large prime par Thériault, puis, en collaboration avec Thomé, Thériault et Diem, nous avons introduit une variante double large prime. Dans toutes ces approches, on commence par ne garder dans la base de facteurs qu'une partie des diviseurs de poids 1 : soit r un paramètre tel que $0 < r < 1$ qui sera fixé par la suite. On choisit arbitrairement environ q^r diviseurs de poids 1 pour former la base de facteurs $\mathcal{F}$. Les diviseurs de poids 1 restant sont appelés «large primes». La définition de diviseur friable est alors identique.

Dans la variante de Harley, on ignore complètement les diviseurs qui ne sont pas friables, tout comme dans l'algorithme original. La probabilité de trouver un diviseur friable se réduit à $q^{(rg-g)}/g!$. Si bien que le calcul de q^r relations coûte $O(g!q^{r+g(1-r)})$. Ensuite l'algèbre linéaire coûte $O(q^{2r})$. La valeur de r permettant de minimiser le temps total est $r = 1 - 1/(g+1)$, pour une complexité finale de $O(q^{2-2/(g+1)})$.

L'idée de Thériault est d'utiliser aussi les diviseurs presque friables, c'est-à-dire ceux qui se décomposent en $g-1$ éléments de la base de facteurs et d'un large prime. De telles relations incomplètes sont bien plus faciles à trouver, et deux telles relations peuvent se recombiner pour former une écriture uniquement sur la base de facteurs sous réserve que les large primes qui interviennent sont les mêmes. Au cœur de l'analyse se trouve le paradoxe des anniversaires qui permet d'estimer qu'après avoir calculé k relations incomplètes, on aura la possibilité de les recombiner pour former environ $k^2/2q$ relations complètes. Il nous faut donc calculer de l'ordre de $q^{(r+1)/2}$ relations incomplètes. Chacune d'entre elles requiert environ $q^{(1-r)(g-1)}$, si bien que la valeur optimale de r est $1-1/(g+1/2)$, pour une complexité finale de $(q^{2-2/(g+1/2)})$.

La variante avec deux large primes consiste à pousser l'idée de Thériault plus loin, en conservant aussi les relations qui sont doublement incomplètes, c'est-à-dire celles qui font intervenir $g-2$ éléments de la base de facteurs et 2 large primes. Aussi bien du point de vue algorithmique que du point de vue de l'analyse, il est pratique d'introduire un graphe G associé au calcul : les sommets du graphe sont étiquetés par les large primes. Les arêtes sont étiquetées par des relations doublement incomplètes dont les large primes sont précisément les extrémités des arêtes concernées. Un cycle dans le graphe correspond essentiellement à la possibilité de combiner les relations incomplètes qui étiquettent le cycle en une relation ne faisant plus intervenir de large prime. Ainsi, un algorithme de détection de cycle du type union-find sera utilisé au cours de l'algorithme ; la complexité de cet algorithme est quasi-linéaire en le nombre d'arêtes du graphe, si bien que la recombinaison des relations doublement incomplètes se fait essentiellement sans surcoût.

L'analyse précise du nombre d'arêtes qu'il faut calculer avant d'atteindre un nombre prescrit de cycles indépendants est complexe. Cela dit, on peut le borner de manière élémentaire par le nombre de sommets plus le nombre de cycles souhaités. Comme on veut calculer q^r cycles et qu'il y a environ q arêtes, on a besoin de $O(q)$ relations doublement incomplètes, chacune requérant de l'ordre de $q^{(1-r)(g-2)}$ tentatives. En mettant tout ceci ensemble, on obtient une valeur optimale pour r de $1 - 1/g$, et une complexité finale de $(q^{2-2/g})$. Telle que décrite, cette analyse est approximative, et de fait s'appuie sur des heuristiques masquées. Par exemple dans le calcul de la valeur optimale pour r on suppose que la phase d'algèbre linéaire prend un temps quadratique, ce qui est vrai uniquement si la matrice est creuse, ce qui n'est

plus garanti, car chaque ligne correspond à un cycle, et le poids de la ligne est proportionnel à la longueur du cycle, ce qui est délicat à borner.

Dans l'article [77] une analyse rigoureuse est effectuée, qui aboutit au théorème suivant. Pour cela, l'algorithme est modifié afin de faciliter l'analyse. Notons aussi qu'en pratique, on a aussi un algorithme plus complexe que celui esquissé ci-dessus, car on tire parti des relations complètes ou simplement incomplètes que l'on pourrait trouver au cours de la recherche. Pour modéliser cela dans le graphe, on rajoute un sommet singulier étiqueté par 1.

Théorème 6 (Gaudry,Thomé,Thériault,Diem) *Soit $g \geqslant 3$ fixé. Soit $\mathcal{C}$ une courbe hyperelliptique de genre g sur $\mathbb{F}_q$ telle que la jacobienne* $\mathrm{Jac}(\mathcal{C})$ *est cyclique. Alors le problème du logarithme discret dans* $\mathrm{Jac}(\mathcal{C})$ *peut être résolu par un algorithme probabiliste de complexité*

$$\tilde{O}(q^{2-2/g}).$$

Le tableau suivant résume les complexités des différents algorithmes pour les courbes de petits genres :

g	3	4	5	6
algo générique	$q^{3/2}$	q^2	$q^{5/2}$	q^3
index calculus	q^2	q^2	q^2	q^2
réduction de base	$q^{3/2}$	$q^{8/5}$	$q^{5/3}$	$q^{12/7}$
Single large prime	$q^{10/7}$	$q^{14/9}$	$q^{18/11}$	$q^{22/13}$
Double large prime	$q^{4/3}$	$q^{3/2}$	$q^{8/5}$	$q^{5/3}$

Après ce travail, Diem [32] a montré que l'on pouvait revenir à l'esprit de l'algorithme original de Adleman-DeMarrais-Huang qui cherche des relations à l'aide de diviseurs principaux plutôt qu'avec des diviseurs réduits. Ceci se fait de manière particulièrement efficace lorsque la courbe considérée admet une équation de petit degré par rapport à son genre. En particulier, les courbes hyperelliptiques sont exclues car une courbe hyperelliptique de genre g a un degré au moins $g+2$. Le résultat (heuristique) de Diem est le suivant : pour une courbe admettant un modèle plan de degré d, il existe un algorithme qui calcule un logarithme discret en une complexité de $\tilde{O}(q^{2-2/d})$.

Le cas le plus intéressant couvert par ce résultat est celui des courbes non-hyperelliptiques de genre 3, pour lequel l'algorithme de Diem donne une complexité heuristique de $\tilde{O}(q)$, à comparer à la complexité $\tilde{O}(q^{3/2})$ auparavant. Ce cas particulier des courbes de genre 3 est particulièrement intéressant pour les applications cryptographiques, et a été étudié plus en détail par Diem et Thomé [34] : ils ont prouvé certaines des heuristiques pour ce cas particulier, et ont mené des expériences numériques pour justifier la dernière hypothèse restante. Ces expériences confirment que les courbes de genre 3 non hyperelliptiques sont bien plus facile à attaquer à l'aide de l'algorithme de Diem que par un algorithme générique.

Il peut être surprenant que pour le genre 3 les courbes hyperelliptiques paraissent plus résistantes que les non-hyperelliptiques. En effet, en général l'algorithmique est plus aisée dans le cas hyperelliptique. Il est cependant probable que d'ici peu de temps on disposera aussi d'un algorithme en $\tilde{O}(q)$ pour les courbes hyperelliptiques de genre 3. Une première étape a été franchie par Smith [157] : partant d'une courbe hyperelliptique de genre 3, il parvient dans certains cas a construire une isogénie explicite vers une courbe non-hyperelliptique de genre 3, et donc d'y transférer le problème du logarithme discret. Ces calculs d'isogénies explicites

sont monnaie courante dans le cas des courbes elliptiques, et les travaux de Smith semblent être le début d'une extension complète au genre supérieur.

Si cette prédiction se confirme, les courbes de genre 3 perdraient tout intérêt par rapport au genre 2, car le corps de base devrait être essentiellement le même pour garantir une sécurité équivalente. Il est donc plus raisonnable de se concentrer sur les courbes elliptiques et de genre 2.

2.2.4 Une classe de courbes avec un algorithme en $L(1/3)$

L'idée que si une courbe admet un modèle de degré petit, alors l'algorithme de calcul d'index peut être amélioré est aussi valide dans un contexte sous-exponentiel où l'on fait tendre le genre vers l'infini. On peut même aller plus loin : pour une certaine classe de courbes très particulières, la complexité de calcul du logarithme discret peut-être de l'ordre de $L(1/3)$ au lieu des complexités classiques en $L(1/2)$. Il s'agit d'un travail commun avec Enge que nous allons expliquer maintenant, sans rentrer dans les détails techniques qui deviennent vite assez lourds.

Pour faire simple, nous allons nous concentrer sur les courbes de type C_{ab}, dont les degrés en x et y sont essentiellement le carré l'un de l'autre : il s'agit des courbes qui admettent une équation de la forme

$$\mathcal{C} : y^n + x^d + f(x, y) = 0$$

sans point singulier autre qu'à l'infini, avec de plus les conditions que $\gcd(n, d) = 1$ et que tout monôme $x^i y^j$ intervenant dans $f(x, y)$ vérifie $ni + dj < nd$. Le genre d'une telle courbe est $g = \frac{(n-1)(d-1)}{2}$; notre hypothèse sur les degrés signifie que $n \approx g^{1/3}$ et $d \approx g^{2/3}$ (nous utilisons le symbole $\approx$ pour signifier «environ de la même taille», sans définition précise). Le corps de base $\mathbb{F}_q$ est supposé fixé ; là encore il s'agit d'une simplification pour l'exposition : l'analyse requiert juste que q ne croisse pas trop vite par rapport à g.

Choisissons une base de facteurs $\mathcal{F}$ formée des $L_{q^g}(1/3)$ diviseurs premiers de plus petit degré. Ceci correspond à choisir les diviseurs premières jusqu'au degré $B \approx \log_q L(1/3)$. Pour construire des relations entre les éléments de la base de facteurs, on considère les fonctions linéaires en y de la forme

$$\varphi = a(x) + b(x)y$$

avec $a,\ b \in \mathbb{F}_q[x]$, $\gcd(a, b) = 1$ et $\deg a,\ \deg b = \delta \approx g^{1/3}$. Soit N la fonction norme de l'extension de corps de fonctions $\mathbb{F}_q(x)[y]/(y^n + x^d + f(x, y))$ par rapport à $\mathbb{F}_q(x)$. La norme de φ se calcule comme

$$\begin{aligned} \mathrm{N}(\varphi) &= \mathrm{N}(b)\mathrm{N}\left(y + \frac{a}{b}\right) \\ &= b^n \left(\left(-\frac{a}{b}\right)^n + x^d + f\left(x, -\frac{a}{b}\right)\right) \\ &= (-a)^n + b^n x^d + f^*(x), \end{aligned}$$

où chaque monôme $x^i y^j$ apparaissant dans f est transformé en le monôme $x^i(-a)^j b^{n-j}$ dans f^*. La norme de φ est B-friable (en tant que polynôme de $\mathbb{F}_q[x]$), si et seulement si le diviseur de la fonction φ est friable par rapport à la base de facteurs que l'on s'est fixée. De plus ceci est algorithmique : à chaque facteur irréductible de $\mathrm{N}(\varphi)$, on peut associer un diviseur premier du même degré qui intervient dans le diviseur de φ. La décomposition en diviseurs friables se ramène donc, comme dans le cas des courbes hyperelliptiques à un problème de factorisation de polynômes univariés.

Nous allons nous appuyer sur l'hypothèse heuristique que les normes des fonctions que l'on calcule se comporte de manière similaire à des polynômes aléatoires de même degré vis-à-vis des propriétés de friabilité. Comme les normes construites ont un degré de l'ordre de $g^{2/3}$, on va donc estimer que celles-ci sont B-friables avec probabilité $1/L_{q^g}(1/3, O(1))$. Ainsi le temps moyen pour obtenir $|\mathcal{F}| = L_{q^g}(1/3, O(1))$ relations est en $L_{q^g}(1/3, O(1))$. On peut ensuite effectuer un calcul de forme normale de Smith pour déduire la structure de groupe de la jacobienne (si celle-ci n'était pas connue). Là encore le coût est en $L_{q^g}(1/3, O(1))$. Cette phase est en fait un pré-calcul qu'il faut effectuer pour chaque courbe avant d'y résoudre des logarithmes discrets. Avant de détailler quelque peu cette phase, il reste à s'assurer que l'espace de recherche annoncé est suffisamment grand pour produire la quantité de relations souhaitée. Le nombre de possibilité pour les polynômes a et b est de l'ordre de

$$q^{2\delta} = q^{2g^{1/3}} = \exp(2 \log q g^{1/3}) < \exp(2(g^{1/3}(\log q)^{1/3})(\log(g \log q))^{2/3}) = L(1/3).$$

Cette inégalité est dans le sens inverse du sens souhaité ; pour s'en sortir, il est nécessaire de mener une analyse plus rigoureuse, sans se contenter des constantes $O(1)$ non explicites que nous avons volontairement laissées pour garder une certaine lisibilité. Nous ne rentrerons pas dans ces détails et renvoyons pour ceux-ci à l'article [41]. Le résultat qu'on obtient est effectivement du même ordre de grandeur que celui obtenu avec l'analyse approximative.

Nous allons maintenant décrire la phase de calcul des logarithmes discrets. Cette étape va se faire par une stratégie de «descente par spécial-Q». Malheureusement, il est nécessaire d'autoriser un temps de calcul un peu plus grand que $L_{q^g}(1/3, O(1))$: on se fixe un réel positif ε et l'on va s'autoriser une complexité de $L_{q^g}(1/3 + \varepsilon, O(1))$. Soit P et Q les éléments de la jacobienne de $\mathcal{C}$ qui définissent le problème de logarithme discret que l'on cherche à résoudre. Ces deux points vont être traités séparément, chacun étant récrit par la procédure suivante sous forme d'une somme d'éléments de la base de facteurs. On part de l'élément P. En passant un temps de $L_{q^g}(1/3 + \varepsilon, O(1))$, on peut récrire P comme une somme d'éléments de degré $\log_q L_{q^g}(2/3 - \varepsilon, O(1))$. Pour cela, on procède de manière classique en ajoutant à P des éléments aléatoires de la base de facteurs : le diviseur réduit obtenu est ensuite testé pour sa friabilité. Comme le diviseur réduit fait intervenir un diviseur effectif de degré au plus g, les probabilités de friabilité sont similaires à celle des polynômes de degré g, d'où le résultat. Ainsi P est égal, dans la jacobienne, à $\sum R_i$, où chaque R_i est un diviseur premier de degré au plus $\log_q L_{q^g}(2/3-\varepsilon, O(1))$. La technique de descente commence alors : chaque R_i va lui-même être récrit en une somme de diviseurs de degré au plus $\log_q L_{q^g}(2/3 - 2\varepsilon, O(1))$. On continue ensuite à récrire chaque nouvel élément comme une somme de diviseurs de degré inférieur, si bien que l'on crée un arbre de décomposition enraciné en P, tel que P vaut la somme des feuilles. Décrivons comment l'on passe de R_i à une somme d'éléments de degré plus petit : supposons que R_i a degré $\log_q L_{q^g}(1/3+t, O(1))$, avec $\varepsilon < t < \frac{1}{3} - \varepsilon$. L'ensemble des fonctions de la forme $a(x) + b(x)y$ dont le diviseur contient R_i est un réseau. Sans rentrer dans les détails, on peut trouver facilement deux vecteurs indépendants de petite taille formant une base de ce réseau. On va donc tester la friabilité des diviseurs des fonctions formées comme petite combinaison linéaire des vecteurs de base. Tout calculs faits, on peut montrer qu'en y passant un temps $L_{q^g}(1/3 + \varepsilon, O(1))$, on a de bonnes chances d'obtenir un diviseur qui soit composé d'éléments de degré au plus $\log_q L_{q^g}(1/3 + t - \varepsilon, O(1))$.

Comme pour la première phase, il est nécessaire d'expliciter toutes les constantes omises dans les $O(1)$, afin de pouvoir s'assurer qu'il est possible de régler les paramètres de la descente pour que la complexité heuristique globale tienne bien en $L_{q^g}(1/3 + \varepsilon, O(1))$. Il faut aussi

vérifier que la taille de l'arbre obtenu est suffisamment contrôlée pour ne pas perturber cette complexité. Là encore, nous renvoyons à [41] pour de plus amples détails. Mentionnons aussi que la classe de courbes concernées est un peu plus vaste que celle des courbes C_{ab} que nous avons considérées ici.

2.2.5 Perspectives

Mentionnons tout d'abord que Diem [30] a développé un algorithme très similaire au nôtre, avec une complexité en $L_{q^g}(1/3, c + o(1))$ pour une classe de courbes algébriques. Bien que l'approche semble différente au premier abord, il nous semble qu'il s'agit principalement d'un changement de terminologie pour les objets manipulés. Il reste cependant à unifier clairement ces deux algorithmes.

Dans cet esprit, nous travaillons, en collaboration avec Enge et Thomé sur une amélioration de notre algorithme, afin de n'avoir plus d'ε dans la complexité, et d'étendre le champ d'application à une classe de courbes plus vaste. Cela engloberait alors les courbes attaquées par Diem.

Se pose ensuite la question de la rareté de ces courbes «faibles». En effet, si l'on considère une courbe aléatoire elle n'aura, a priori, pas de modèle de petit degré par rapport à son genre, et encore moins de la forme exigée par notre algorithme. On peut tenter, comme nous le faisons actuellement, d'étendre la classe de courbes pour lesquelles on a une bonne complexité pour le logarithme discret. Cela dit, il paraît peu vraisemblable que l'on parviendra à attaquer ainsi une grande proportion des courbes. Une approche plus prometeuse, mais pour l'instant à l'état d'ébauche, serait de suivre la voie ouverte par Smith pour les calculs d'isogénies explicites. En effet, le fait d'admettre ou non un modèle plan de petit degré n'est pas invariant par isogénie, et l'on peut imaginer de parcourir plus ou moins aléatoirement la classe d'isogénie de la courbe cible jusqu'à trouver une courbe qui admette un modèle plan de petit degré. Comme une isogénie permet de transporter un problème de logarithme discret, cela permettrait donc de retrouver une complexité en $L(1/3)$ pour des courbes qui n'y sont a priori pas sujettes.

Un problème théorique apparaît alors : quelle est la proportion des classes d'isogénies admettant une courbe ayant un modèle plan de petit degré ?

2.3 Descente de Weil

Nous allons décrire le principe de l'attaque par descente de Weil dans le cas de courbes elliptiques, sachant que ce principe s'applique de manière totalement similaire à des courbes de genre supérieur. Soit donc E une courbe elliptique sur un corps fini non premier $\mathbb{F}_{q^n}$. On suppose que E est utilisé dans un contexte cryptographique, si bien que l'on peut supposer que E est cyclique d'ordre premier N. La restriction de Weil est une construction classique de géométrie algébrique donnée par une propriété universelle et qui fournit une variété abélienne A définie sur $\mathbb{F}_q$ associée à E. La construction de cette variété A a en fait une interprétation pratique très intuitive. On se donne une représentation du corps $\mathbb{F}_{q^n}$ comme $\mathbb{F}_q[t]/(f(t))$ où $f(t)$ est un polynôme irréductible de degré n sur $\mathbb{F}_q$. Partant d'une équation $y^2 = x^3 + ax + b$ pour E, on écrit les coordonnées x et y à l'aide de cette représentation de $\mathbb{F}_{q^n}$ comme $x = x_0 + x_1 t + \cdots + x_{n-1} t^{n-1}$ et de même pour y. Ainsi on remplace deux coordonnées sur $\mathbb{F}_{q^n}$ par $2n$ coordonnées sur $\mathbb{F}_q$. L'équation de E devient une équation polynomiale en t de degré $n-1$, si bien que l'on a n équations reliant les nouvelles coordonnées. Le bilan est donc une

description de E comme une variété de dimension n sur $\mathbb{F}_q$. De plus, la correspondance entre les deux descriptions est simple, si bien que la loi de groupe sur E se transcrit en une loi de groupe sur cette nouvelle variété. Cette construction terre-à-terre est purement affine, et pour pouvoir affirmer que l'on obtient bien une variété abélienne, il est nécessaire d'utiliser un peu plus de théorie. Toutefois, c'est bien avec ces équations que l'on va faire les calculs pratiques.

Une manière plus intrinsèque de voir la restriction de Weil est de considérer la variété obtenue en prenant le produit de tous les courbes conjuguées galoisiennes de E. Il y a n conjuguées, donc la variété obtenue est de dimension n. A priori cette variété est définie sur le même corps que ses composants : $\mathbb{F}_{q^n}$, mais par symétrie elle peut en fait être définie sur $\mathbb{F}_q$. Là encore, la loi de groupe est héritée des lois de groupe sur E et ses conjuguées, si bien que l'on va obtenir une variété abélienne sur $\mathbb{F}_q$.

L'utilisation de la restriction de Weil dans un cadre cryptographique remonte à 1998, où Frey [54] a montré que l'on pouvait ainsi masquer le fait que l'on utilise un cryptosystème elliptique. Il a aussi pour la première fois évoqué les applications possibles au problème du logarithme discret elliptique. En effet, si cette variété abélienne A s'avère être la jacobienne d'une courbe de genre n, on peut y appliquer les algorithmes de calcul d'index de la section précédente, et probablement attaquer plus efficacement le cryptosystème elliptique initial.

La question d'exhiber de telles instances faibles a été largement étudiée depuis, et nous allons expliquer brièvement les résultats obtenus.

2.3.1 Algorithme GHS

Demander que la restriction de Weil A soit exactement une jacobienne de courbe est trop exigeant. Soit $\mathcal{C}$ une courbe définie sur $\mathbb{F}_q$ qui soit incluse dans A. Alors, sauf cas dégénéré, A est incluse dans la jacobienne de $\mathcal{C}$. Le genre de $\mathcal{C}$ est forcément supérieur ou égal à n. Le cas idéal où A est la jacobienne de $\mathcal{C}$ correspond au fait que $\mathcal{C}$ est de genre n. Si le genre est plus grand, on est amené à travailler dans un groupe qui est plus grand que le groupe de départ, toutefois, la complexité sous-exponentielle des algorithmes de calcul d'index peut compenser cette perte, si le genre n'est pas beaucoup plus grand que n. Le but va donc être de trouver des courbes elliptiques telles que l'on parvienne à trouver des courbes de petit genre dans leur restriction de Weil.

En collaboration avec Heß et Smart, nous avons fourni la première construction complète suivant ce programme [70] (en nous appuyant sur quelques travaux préliminaires par Galbraith et Smart [59]). Cela concerne les courbes elliptiques en caractéristique 2. L'idée pour obtenir une courbe de genre modéré est de chercher parmi les courbes de degré petit. Et pour obtenir une courbe de faible degré, on peut tenter de couper la variété A par des hypersurfaces de petit degré. De fait, on va donc couper A par des hyperplans de coordonnées. Une équation explicite de la courbe peut être obtenue, et il s'avère que cette courbe est hyperelliptique. Son genre est de l'ordre de 2^m où m est un paramètre facilement calculable à partir de l'équation initiale de la courbe elliptique. En général, on s'attend à ce que m soit de l'ordre de n, si bien que l'attaque échoue : le fait de disposer d'un algorithme sous-exponentiel ne compense pas le fait que la taille du groupe a augmenté de façon exponentielle. Toutefois, il existe des courbes elliptiques pour lesquelles le paramètre m est petit (et l'on peut construire et détecter facilement ces courbes).

Un aspect que nous n'avons pas encore évoqué est la manière d'envoyer le problème discret initial dans la jacobienne de $\mathcal{C}$. Pour cela, on utilise le langage des corps de fonctions : dans la construction GHS, on a de fait un morphisme algébrique explicite φ de $\mathcal{C}$ (que l'on voit

comme une courbe sur $\mathbb{F}_{q^n}$) vers la courbe elliptique E. Si P est un point de E, on peut former le diviseur D_P sur $\mathcal{C}$ obtenu comme somme des pré-images de P par φ. Si l'on étend cette construction aux diviseurs de degré 0 par linéarité, on obtient un morphisme entre les classes de diviseurs de degré 0 (il s'agit en fait de la conorme entre les corps de fonctions de $\mathcal{C}$ et E). Ainsi, on peut envoyer un problème du logarithme discret dans E dans la jacobienne de $\mathcal{C}$ sur $\mathbb{F}_{q^n}$. Il reste à projeter ce dernier dans la jacobienne de $\mathcal{C}$ sur $\mathbb{F}_q$. Pour cela, on applique une fonction norme : chaque diviseur est remplacé par la somme de ses conjugués galoisiens. La composition de la conorme et de la norme n'est pas injective. Il se pourrait donc que l'on ait envoyé tous les points sur 0. Toutefois, on peut montrer que si l'on est parti de points d'ordre premier suffisamment grand, ils ne peuvent pas être dans le noyau, si bien que le défaut d'injectivité n'est pas un problème en pratique.

Une fois établi le fait qu'il existe des courbes elliptiques faibles sur $\mathbb{F}_{2^n}$, la question naturelle est la proportion de telles courbes faibles parmi l'ensemble des courbes. Des études systématiques ont été produites dans ce sens [121, 118]. D'une manière générale, l'attaque a plus de chances de réussir si l'exposant n est composé. En effet, cela donne plus de marge pour choisir quel est le corps de base et quel est l'exposant dans l'attaque GHS.

Une attaque supplémentaire a été ajoutée par Galbraith, Heß et Smart [58] : soit à résoudre un problème de logarithme discret dans une courbe elliptique sur $\mathbb{F}_{2^n}$ pour laquelle l'attaque GHS échoue. Si la proportion de courbes elliptiques faibles sur $\mathbb{F}_{2^n}$ est suffisante, on peut appliquer des isogénies afin de sauter d'une courbe elliptique à une autre, tout en transportant le problème du logarithme discret. Le fait que l'attaque GHS fonctionne ou non (i.e. le paramètre m) n'est pas invariant par isogénie, on peut donc espérer parvenir rapidement sur une courbe faible. Notons que le calcul d'isogénie explicite est parfaitement maîtrisé dans le cas des courbes elliptiques, suite aux nombreux travaux liés à l'algorithme de Schoof. Ce n'est par contre pas le cas pour les courbes plus générales. Heß [86, 88] a ensuite de nouveau étendu l'attaque en considérant différente manière de tracer une courbe la courbe $\mathcal{C}$.

Prenant en compte cette nouvelle attaque, on peut de nouveau évaluer les proportions de courbes faibles pour chaque corps fini donné [122]. Il s'est alors révélé que pour certains corps, toutes les courbes elliptiques sont faibles (au sens où il existe un algorithme meilleur que l'attaque générique par la méthode de Pollard) [123]. Là encore, en évitant les extensions composées, on se met essentiellement à l'abri. C'est une assez mauvaise nouvelle pour les applications embarquées qui étaient friandes d'extensions composées pour accélérer l'arithmétique du corps de base.

Suite au succès de cette approche, diverses extensions et analyses ont été effectuées. De nouvelles classes de courbes faibles ont ainsi été mises en évidence, en grande caractéristique, ou pour les courbes hyperelliptiques, toujours en suivant l'approche GHS décrite ci-dessus pour construire la courbe $\mathcal{C}$ et l'exploiter [31, 33, 167, 165, 57, 164, 93, 3, 131, 130].

2.3.2 Calcul d'index dans les variétés abéliennes

Comme évoqué précédemment, il est possible que l'attaque GHS échoue à cause du fait que les seules courbes que l'on parvient à tracer sur la restriction de Weil sont de genre trop grand par rapport à la dimension de la variété, si bien que le calcul d'index aura lieu dans une jacobienne bien trop grande. Nous avons proposé un moyen de contourner ce problème dans certains cas. Le cœur de cette approche est la conception d'un algorithme de calcul d'index travaillant directement au niveau des variétés abéliennes, sans s'appuyer sur une jacobienne.

Précisons le cadre de travail : on suppose que l'on a une variété abélienne de dimension n sur $\mathbb{F}_q$ donnée par des équations explicites et dont la loi de groupe est aussi connue par des formules explicites. On a un problème de logarithme discret à résoudre entre deux points de cette variété abélienne. Le fait que cette variété puisse être la restriction de Weil d'une courbe elliptique ne sera pas utilisé par la suite (contrairement à l'attaque GHS, où on s'en est servi pour envoyer le problème du logarithme discret dans la jacobienne de $\mathcal{C}$).

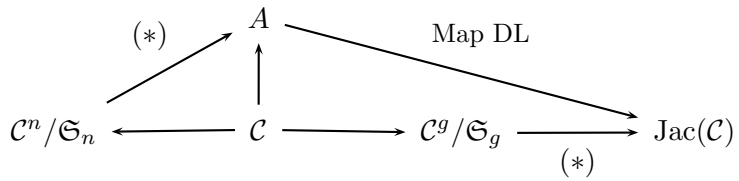
L'algorithme commence comme précédemment : on trace une courbe $\mathcal{C}$ sur A aussi simple que possible. Typiquement, on coupe A par des hyperplans. La courbe $\mathcal{C}$ est supposée absolument irréductible (ce qui est heuristiquement vraisemblable, puisque l'on part de la variété A qui l'est). On considère alors la base de facteurs $\mathcal{F}$ formée des points de $\mathcal{C}$ sur $\mathbb{F}_q$; il y en a de l'ordre de q.

Un élément de A sera dit friable s'il est égal à la somme de n points de $\mathcal{F}$. Les éléments friables de A sont des les éléments de la somme symétrique $\mathcal{F}^{(n)}$ qui comporte de l'ordre de $q^n/n!$ éléments. Ainsi la probabilité qu'un élément aléatoire soit friable est en $1/n!$. Jusqu'ici, tout semble parfait : cette notion de friabilité émule celle que l'on aurait avec une jacobienne de courbe de genre n, même si la courbe $\mathcal{C}$ a en fait un genre qui risque d'être exponentiel en n. Les difficultés apparaissent lorsque l'on cherche à rendre effective cette notion de friabilité. En effet, on ne peut plus ramener simplement celle-ci à une friabilité de polynômes en lisant directement la représentation de Mumford (ou l'analogue pour les courbes non-hyperelliptiques). Cela dit, le problème apparaît naturellement comme un système d'équations polynomiales : les coordonnées des n éléments de $\mathbb{F}_c$ étant laissées comme indéterminées, on peut écrire que leur somme vaut le point que l'on cherche à friabiliser grâce aux équations décrivant la loi de groupe. Ainsi, on obtient un système qui est génériquement de dimension 0, et dont le degré est borné indépendamment de q. Il est alors possible de faire appel aux techniques de calcul formel, telles que les bases de Gröbner pour trouver les solutions de ce système. La complexité de cette résolution est polynomiale en la taille du corps de base, et exponentielle en les autres paramètres du système, si bien que cette approche ne sera acceptable que pour de petites valeurs de n.

Si l'on fait une analyse plus précise de la complexité obtenue, toujours en gardant n fixé, on obtient un algorithme en $\tilde{O}(q^{2-2/n})$. Cette complexité est heuristique et implique notamment d'utiliser une variante avec deux «large primes». Comme dit précédemment, la dépendance en n est catastrophique à cause des résolutions de systèmes polynomiaux qu'il faut effectuer à chaque étape de friabilisation. Dans le cas où la variété A est la restriction de Weil d'une courbe elliptique sur $\mathbb{F}_{q^n}$, les systèmes polynomiaux à résoudre prennent une forme plus agréable en utilisant les polynômes de Semaev [147]. On peut alors estimer le degré des idéaux représentés par les systèmes comme étant de l'ordre de 2^{n^2}. Il est donc clair que l'on ne pourra pas traiter en pratique des valeurs de n autres que 3 ou 4 ($n = 2$ n'apporte rien par rapport à l'algorithme de Pollard).

Comparaison des deux attaques

Le diagramme suivant illustre les applications impliquées dans l'attaque GHS ainsi que dans le calcul d'index sur la variété abélienne.



Les flèches marquées d'une astérisque sont celles où le calcul d'index a lieu. Les avantages du calcul d'index dans la variété abélienne sont les suivants :

- Aucune connaissance de la géométrie de $\mathcal{C}$ n'est requise ; on ne travaille jamais dans sa jacobienne.
- La composante factorielle dans la complexité est toujours $n!$, à comparer à $g!$, où g est le genre de $\mathcal{C}$ qui peut être exponentiel en n.

Les inconvénients du calcul d'index dans la variété abélienne sont :

- Les calculs de bases de Gröbner sont un ingrédient complexe que l'on préférerait éviter.
- Si n est grand, la composante en 2^{n^2} dans la base de facteur est rédhibitoire.

Le cas le plus favorable au calcul d'index dans la variété abélienne est donc le cas $n = 3$ que nous allons regarder plus précisément. Les systèmes polynomiaux impliqués sont alors suffisamment simples pour être traités par des calculs de résultants. En comparaison, Diem [31] a montré que pour une courbe elliptique générique sur $\mathbb{F}_{q^3}$, l'attaque GHS produira une courbe de genre au moins 13. Ainsi, trouver une relation lors d'un calcul d'index dans cette jacobienne va requérir au moins $13! \approx 8\ 10^9$ essais, ce qui sera bien plus lent que dans le cas de l'attaque directe avec les résultants.

Au-delà de ces considérations, la complexité de l'attaque pour une courbe elliptique générique sur $\mathbb{F}_{q^3}$ sera de $\tilde{O}(q^{4/3})$ avec un calcul d'index dans la variété abélienne, alors que l'attaque GHS donnera $\tilde{O}(q^{1.85})$, ce qui est moins bon que la méthode de Pollard.

En revanche, il existe des courbes elliptiques sur $\mathbb{F}_{q^3}$ telles que l'attaque GHS produise des courbes de genre 3 [31]. Pour ces courbes là, il est bien meilleur d'utiliser GHS, car on économise les calculs de résultants.

2.3.3 Extensions

Ce travail a été étendu depuis dans diverses directions.

Tout d'abord, tel que présenté ci-dessus (tout comme dans l'article original), l'algorithme ne s'applique qu'aux variétés abéliennes. Il est toutefois assez simple de l'étendre aux groupes algébriques en général. En effet, tous les calculs explicites se font de toute façon sur une partie affine de la variété. Ainsi Granger et Vercauteren [81] ont montré que l'on pouvait mener un calcul d'index dans le même esprit que ci-dessus dans le contexte des tores algébriques, c'est-à-dire dans certains sous-groupes des groupes multiplicatifs de corps finis.

Une autre extension a été proposée par Diem (non-publié, annoncé à ECC 2004). L'idée est d'analyser précisément la complexité dans le cas de la restriction de Weil d'une courbe elliptique, en particulier la dépendance en n, le degré de l'extension du corps. Ainsi, on peut considérer une famille de courbes elliptiques pour lesquelles ce degré n croît à une vitesse contrôlée de sorte que la complexité du calcul de bases de Gröbner reste raisonnable

par rapport au reste. Diem montre qu'il existe une telle famille pour laquelle la complexité (heuristique) d'un calcul de logarithme discret est sous-exponentielle.

Dans un autre ordre d'idée, Nagao [134] a étudié plus précisément les systèmes algébriques obtenus lorsque l'on applique l'attaque à une courbe hyperelliptique sur une extension de corps. Son approche, s'appuyant sur les espaces de Riemann-Roch simplifie l'écriture des systèmes, et permet une analyse plus précise de la complexité de l'algorithme.

Chapitre 3

Algorithmes pour la conception de systèmes à base de courbes algébriques

Une part importante de ce chapitre est consacré au problème du comptage de points d'une courbe sur un corps fini. S'inspirant d'un article de survol [61] écrit à l'occasion d'un cours donné à l'IHP, nous avons tenté d'effectuer un panorama de la situation actuelle. À chaque fois, nous insistons un peu plus sur les parties auxquelles nous avons contribué. Nous convrons donc les algorithmes de complexité exponentielle (couvrant nos contributions [11, 75]), les algorithmes inspirés de l'algorithme de Schoof [69, 74, 76], l'algorithme de Satoh et ses variantes [49, 50, 65], l'algorithme de Kedlaya et ses extensions [68, 67], ainsi que les algorithmes p-adiques par déformation sur lesquels nous n'avons pas contribués.

Les autres travaux constructifs que nous mentionnons dans ce chapitre sont ceux liés à la théorie de la multiplication complexe en genre 2 [71], ainsi que des travaux sur les formules explicites pour une loi de groupe efficace en genre 2 [66, 73].

3.1 Comptage de points : survol des algorithmes disponibles

Avant de développer différents aspects du problème de comptage de points, nous allons passer en revue l'ensemble des techniques à notre diposition, en les regroupant par grandes catégories d'algorithmes.

3.1.1 Les algorithmes exponentiels

Nous commençons ce survol par les algorithmes de complexité exponentielle. Ceux-ci ne permettent pas de traiter de très grandes tailles de paramètres, mais ils peuvent en général se combiner avec d'autres algorithmes ayant une meilleure complexité.

L'approche classique «pas de bébé, pas de géant» peut être améliorée pour tenir compte des spécificités des courbes, comme par exemple dans [117]. Ensuite, des variantes heuristiques ou probabilistes permettent de diminuer la complexité en mémoire de ces méthodes.

Toujours utilisant la spécificité des courbes, la méthode dite d'approximation est parfois utilisée avant de passer aux «pas de bébé, pas de géant» : il s'agit de compter le nombre de points de la courbe sur les extensions de petit degré du corps de base, afin de gagner de l'information sur le polynôme caractéristique du Frobenius. Voir par exemple [160].

Récemment, Sutherland [163] a eu l'idée de rechercher des Jacobiennes dont l'ordre est friable, de sorte que le calcul de leur ordre est facilité. Ensuite, si l'on est intéressé par des ordres premiers (pour la cryptographie), on peut regarder la tordue quadratique, dont le cardinal se déduit facilement, et a des chances d'être premier.

Une dernière méthode que nous mentionnerons est le calcul de l'opérateur de Cartier-Manin. Cela sera développé en section 3.2.2

3.1.2 Les algorithmes ℓ-adiques

L'algorithme publié par Schoof en 1985 [145] pour compter le nombre de points d'une courbe elliptique sur un corps fini fut le premier algorithme à atteindre une complexité polynomiale. Il fut ensuite étendu aux variétés abéliennes par Pila [136], puis par Adleman et Huang [1] ainsi que Huang et Ierardi [89].

La complexité est polynomiale en $\log q = n \log p$ pour toute famille de courbes. Cette notion de famille, donnée par Pila, ne suffit pas pour conclure en toute généralité sur une complexité polynomiale en $n \log p$ uniformément pour toutes les courbes de genre fixé. Mais ce résultat couvre, par exemple, toutes les courbes hyperelliptiques de genre fixé.

On peut noter que l'algorithme original de Schoof est déterministe. Les extensions ont parfois été conçues en insistant pour conserver cette propriété, au prix de complications importantes. Dans notre exposé nous omettrons souvent de mentionner le caractère probabiliste ou déterministe des algorithmes, même si les avantages d'un algorithme déterministe ne sont pas à négliger.

3.1.3 Les algorithmes sous-exponentiels

Cette classe d'algorithmes a été initiée en 1994 par Adleman, DeMarrais et Huang [2] dans le cadre des courbes hyperelliptiques. Il s'agissait à l'origine d'une première étape dans un calcul de logarithmes discrets dans la jacobienne d'une courbe hyperelliptique. La complexité, bien qu'heuristique, était sous-exponentielle, sous la condition que le genre croisse suffisamment vite par rapport à $n \log p$.

L'algorithme initial a été étendu, amélioré, prouvé dans certains cas par diverses personnes [132, 39, 40, 27] et finalement une version a été prouvée dans un cadre très général par Heß [87]. Ce type d'algorithme ne fournit pas toute la fonction Zêta, mais seulement $\chi_\pi(1)$ et les diviseurs élémentaires du groupe des points de la jacobienne.

Toutefois, tous ces algorithmes requièrent que le corps fini et donc sa caractéristique ne soient pas trop grands par rapport au genre de la courbe. Ainsi on entre dans le champ d'application des algorithmes p-adiques, et désormais ces algorithmes sous-exponentiels ne sont plus les plus rapides (du moins asymptotiquement). Toutefois, si l'on est de plus intéressé par la structure du groupe ou par des calculs de logarithme discret, cette approche est la bonne.

3.1.4 Les algorithmes p-adiques utilisant le relèvement canonique

En 1999, Satoh [141] a inventé une nouvelle méthode pour compter les points d'une courbe elliptique. S'appuyant sur le relèvement p-adique canonique de la courbe, cette méthode fonctionne lorsque la caractéristique du corps de base est petite. À p fixé, la complexité est meilleure que celle de l'algorithme de Schoof.

De nombreuses améliorations et généralisations ont eu lieu. Finalement, on dispose d'un algorithme en temps polynomial en n pour calculer la fonction Zêta d'une courbe hyperelliptique de genre fixé sur $\mathbb{F}_{2^n}$. Cela ne change rien du point de vue théorique : ces cas étaient déjà couverts par l'algorithme de Pila. Toutefois, l'algorithme est cette fois-ci très pratique.

3.1.5 Les algorithmes p-adiques utilisant la cohomologie de Monsky et Washnitzer

Peu de temps après la publication de l'algorithme de Satoh, Kedlaya [97] a proposé une autre méthode de comptage de points utilisant un relèvement p-adique, mais cette fois-ci, le relèvement est quelconque et la cohomologie de Monsky-Washnitzer est l'outil qui permet de calculer la fonction Zêta.

Cet algorithme a une bonne complexité à caractéristique p fixée. Décrit tout d'abord dans le cas des courbes hyperelliptiques sur un corps de caractéristique impaire, il a été étendu en plusieurs étapes à des classes de courbes de plus en plus larges. Finalement, pour les courbes C_{ab}, sur un corps de caractéristique p fixée, la complexité est polynomiale en ng.

3.1.6 Les algorithmes p-adiques utilisant une approche à la Dwork

Toujours au début des années 2000, Lauder et Wan [107] ont proposé une autre approche p-adique au problème de comptage de points : leur algorithme suit la preuve de Dwork de la rationalité de la fonction Zêta. L'avantage de leur méthode est qu'elle est extrêmement générale : elle permet de compter le nombre de solutions d'une équation polynomiale sur un corps fini, avec une complexité qui dépend de manière polynomiale en la taille du corps fini et en le degré du polynôme, et exponentielle en le nombre de variables et en la caractéristique du corps.

Telle quelle leur méthode a une complexité moins bonne que l'algorithme de Kedlaya dans les cas où ce dernier s'applique. Des améliorations ont été effectuées pour des classes particulières de courbes [104, 108, 109], notamment en utilisant la cohomologie de Dwork, ce qui a permis de retrouver la même complexité que l'algorithme de Kedlaya pour les courbes hyperelliptiques.

À notre connaissance, très peu d'expériences pratiques ont été menées sur ce type d'algorithmes. Vercauteren a effectué une implantation pour les courbes d'Artin-Schreier, mais sans que cela batte l'algorithme de Kedlaya.

3.1.7 Les algorithmes p-adiques utilisant la théorie de la déformation

L'idée d'utiliser la théorie de la déformation est initialement due à Lauder [105] qui l'a appliquée à son approche à la Dwork. Le principe est de faire des précalculs pour traiter toute une famille de courbe ; ensuite compter les points sur une courbe individuelle de la famille se fait plus rapidement.

Des améliorations et extensions successives ont eu lieu pour aboutir finalement à des algorithmes compétitifs. Notons que désormais, c'est plutôt à un calcul de cohomologie de Monsky-Washnitzer, à base d'algorithme de Kedlaya, que les algorithmes de déformation se sont avérés les plus efficaces. Une brève présentation de ces développements récents se trouve dans les survols [21, 103].

3.1.8 Les implantations

Dans le diagramme de la figure 3.1, nous avons tenté de résumer la situation actuelle pour ce qui est des implantations des algorithmes ci-dessus dans la littérature. La surface dessinée est censée représenter la limite de ce que l'on est capable de traiter aujourd'hui, en tenant compte des algorithmes et de la puissance des ordinateurs. Ainsi, pour une courbe dont le triplet $(n, g, \log p)$ est sous la surface, on peut en principe calculer la fonction Zêta en un temps raisonnable. Nous avons de plus indiqué quelques points particuliers numérotés, la plupart sur cette surface ; ces points correspondent à des «records» pour un certain type de courbes, et nous donnons quelques détails et les références pour chacun d'eux ci-dessous.

Il convient de noter que le diagramme n'a pas vocation à être précis, par exemple le point 5 correspond en réalité à $n = 130\,020$ et le point 4 à $n = 50\,021$, ce qui n'est manifestement pas à l'échelle. Nous avons préféré faire un schéma le plus lisible possible et qui permette de situer les différents résultats dans la littérature les uns par rapport aux autres.

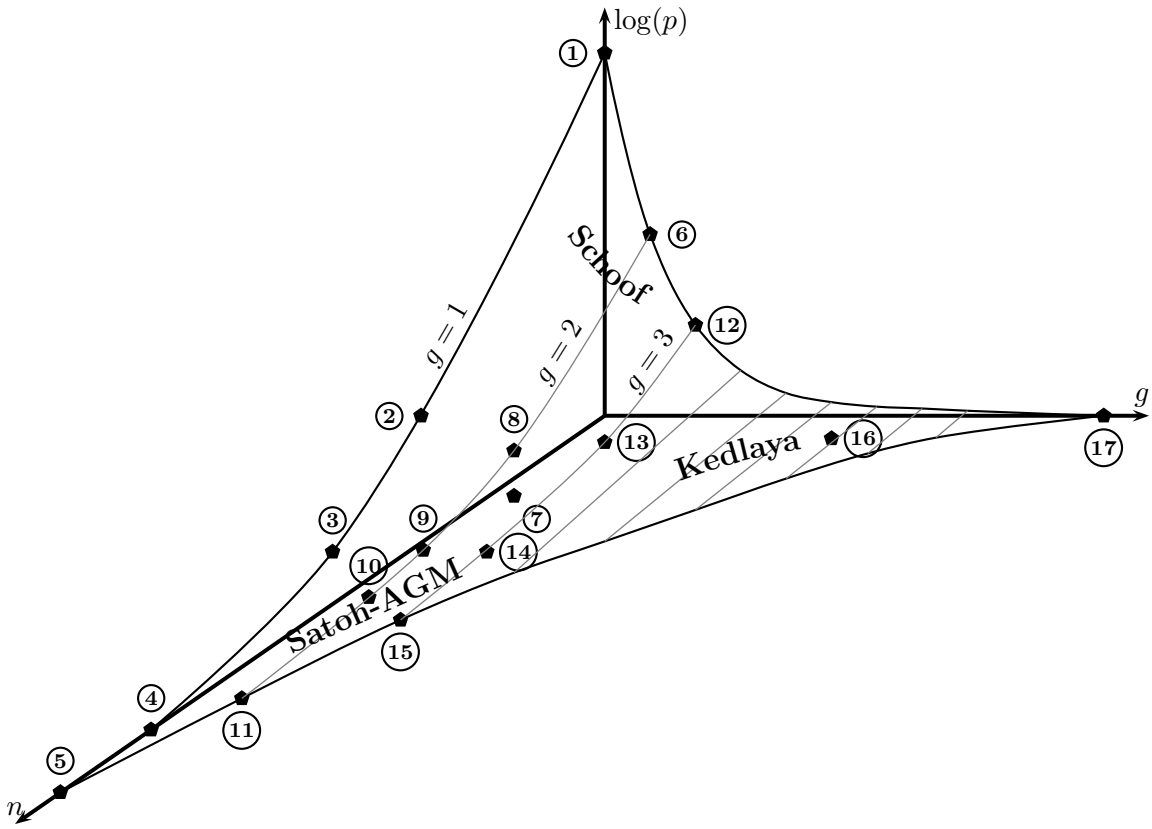
Nous insistons sur le fait que ces calculs n'ont pas été faits à la même date, ni par la même personne, et donc que certaines de ces lignes pourraient certainement être poussées plus loin aujourd'hui avec une implantation extrêmement optimisée.

3.2 Comptage de points par les algorithmes exponentiels

3.2.1 Algorithmes probabilistes recherchant des collisions

Nous avons déjà évoqué des algorithmes génériques dans le contexte du logarithme discret. Pour le comptage des points de la jacobienne d'une courbe, il est possible d'utiliser de tels algorithmes qui fonctionnent dans n'importe quel groupe, pourvu que l'on connaisse des bornes sur son cardinal. La complexité de ces algorithmes est en racine carrée de l'espace de recherche. Ils se distinguent en deux classes : les variantes de l'approche «pas de bébé, pas de géant» qui sont déterministes, et les méthodes s'appuyant sur le paradoxe des anniversaires qui sont probabilistes, voire heuristiques. Dans le premier cas, la complexité en mémoire est du même ordre de grandeur que la complexité en temps, si bien que c'est en général un problème de stockage qui bloque pour les gros calculs. Dans le second cas, la mémoire requise est en général

FIG. 3.1 – Records de calcul de fonctions Zêta de courbes. Afin de rendre le diagramme lisible, celui-ci est uniquement figuratif : les coordonnées des points de records ne sont pas les vraies coordonnées. Les pointes sur les axes devraient être beaucoup plus pointues, en particulier pour celle le long de l'axe des n.



N°	p	n	g (type)	Réf.	Algorithme	Commentaire
①	$\approx 10^{2500}$	1	1	[42]	Schoof	
②	7	4000	1	[92]	Déformation	
③	5	569	1	[141]	Satoh	Première implantation d'un algorithme p-adique
④	2	50 021	1	[82]	AGM	
⑤	2	130 020	1	[82]	AGM	La valeur de n permet l'utilisation de base normale gaussienne
⑥	$2^{127}-1$	1	2		Schoof	
⑦	$2^{32}-5$	3	2	[10]	Schoof	Utilisation de l'opérateur de Cartier-Manin
⑧	$2^{42}+15$	3	2	[84]	Kedlaya	
⑨	3	400	2	[90]	Déformation	
⑩	2	4 001	2	[83]	AGM/Richelot	
⑪	2	32 770	2	[111]	AGM/Borchardt	La valeur de n permet l'utilisation de base normale gaussienne
⑫	$\approx 2,8 \cdot 10^{17}$	1	3 (Picard)	[6, 174]	Ad-Hoc	Utilisation d'un algorithme exponentiel adapté aux courbes de Picard
⑫	$2^{50}-27$	1	3 (hyperell.)	[84]	Kedlaya	
⑬	251	7	3 (hyperell.)	[68]	Kedlaya	
⑭	3	200	3 (hyperell.)	[90]	Déformation	
⑮	2	4 098	3 (hyperell.)	[111]	AGM/Borchardt	
⑮	2	5 002	3 (non-hyperell.)	[139]	AGM/Borchardt	
⑯	2	20	8 (C_{ab})	[20]	Déformation	
⑰	2	1	350 (hyperell.)	[169]	Kedlaya	

Légende pour la figure 3.1

très faible ; c'est pourquoi les méthodes probabilistes sont souvent préférées en pratique, même si elles sont heuristiques.

Ces algorithmes génériques peuvent assez facilement tirer parti d'une information modulaire supplémentaire : soit G le groupe dont on cherche le cardinal, que l'on sait appartenir à un intervalle $[L, H]$. La complexité du calcul est en $O(\sqrt{H-L})$. Si de plus on sait que le cardinal est congru à n_0 modulo m, alors la complexité tombe à $O(\sqrt{(H-L)/m})$.

Nous nous concentrons maintenant sur le cas des courbes de genre 2. En genre 2, l'information modulaire que l'on a provient en général de l'algorithme de Schoof (voir Section 3.3), et par là-même, on dispose de plus d'informations que juste la congruence du cardinal du groupe. Le polynôme caractéristique de l'endomorphisme de Frobenius est de la forme

$$\chi(t) = t^4 - s_1 t^3 + s_2 t^2 - q s_1 t + q^2,$$

et le cardinal de la jacobienne est $\chi(1)$. L'algorithme de Schoof va calculer s_1 et s_2 modulo différents nombres premiers ℓ. Ceci fournit en particulier la valeur de $\chi(1)$ modulo ℓ, fournissant une accélération des algorithmes génériques par un facteur $\sqrt{\ell}$. Matsuo, Chao et Tsujii ont réalisé qu'en fait on peut obtenir une accélération d'un facteur ℓ. Nous allons rappeler l'idée de cette amélioration, ainsi que les grandes lignes de la variante que nous avons proposée en collaboration avec É. Schost, et qui permet de travailler avec une mémoire très réduite.

L'algorithme de Matsuo, Chao et Tsujii (MCT) consiste à effectuer un compromis-temps mémoire du type «pas de bébé, pas de géant» non pas sur le cardinal du groupe, mais sur les quantités s_1 et s_2. Les bornes sur s_1 et s_2 ne sont pas du même ordre de grandeur : on a $|s_1| \leqslant 4\sqrt{q}$ et $|s_2| \leqslant 6q$. Il va donc être nécessaire de découper s_2 en deux morceaux, l'un rejoignant s_1 de sorte que l'on équilibre les deux phases du compromis temps-mémoire. Mettons ceci en œuvre : soit $s_2 = t_1 + Bt_2$, où B et t_1 sont de l'ordre de $q^{1/4}$ et t_2 de l'ordre de $q^{3/4}$. Soit P un élément aléatoire de la jacobienne. On a donc $\chi(1)P = 0$, ce qui se récrit

$$(q^2 + 1 - s_1(q+1) + t_1)P = -t_2 BP.$$

Le nombre de membres de gauches possibles est en $q^{3/4}$, et de même pour le nombre de membres de droites. Ainsi, on peut précalculer tous les membres de gauches, puis rechercher le membre de droite qui correspond à l'un de ces membres de gauches. Ainsi on obtient un algorithme en $O(q^{3/4})$ opérations. Notons que si l'on avait cherché uniquement $\chi(1)$ par une méthode «pas de bébé, pas de géant», comme l'intervalle est en $q^{3/2}$, on aurait aussi un algorithme en $O(q^{3/4})$. Ainsi on ne perd rien à chercher s_1 et s_2. De plus, si l'on est dans un contexte où l'on connaît s_1 et s_2 modulo ℓ, on peut appliquer la même méthode pour rechercher les quotients de s_1 et s_2 dans la division euclidienne par ℓ, si bien que l'on va gagner un facteur ℓ^2 dans l'espace de recherche et donc un facteur ℓ sur le temps de calcul.

Modifier l'algorithme MCT pour éviter d'avoir à stocker un grand nombre de points n'est pas aussi simple qu'il n'y paraît. Le problème est bien maîtrisé dans le cas où l'on cherche le cardinal du groupe, et la solution s'appuie sur une marche pseudo-aléatoire dans le groupe et le paradoxe des anniversaires. Ici, il s'agit de rechercher deux scalaires ; il nous faudra donc utiliser une marche aléatoire en deux dimensions. Revenons à l'équation que doit vérifier un point aléatoire P :

$$(q^2+1)P + (s_2 - s_1(q+1))P = 0.$$

Soit R le rectangle des valeurs possibles pour s_1 et s_2 :

$$R = \{(\sigma_1, \sigma_2), \quad |\sigma_1| \leqslant 4\sqrt{q}, |\sigma_2| \leqslant 6q\}.$$

On peut alors définir deux ensembles de points W et T :

$$W = \{(q^2+1)P + (\sigma_2 - \sigma_1(q+1))P, \quad (\sigma_1, \sigma_2) \in R\},$$

$$T = \{(\sigma_2 - \sigma_1(q+1))P, \quad (\sigma_1, \sigma_2) \in R\}.$$

Par définition, l'intersection de ces deux ensembles de points de la jacobienne est non vide. En effet, supposons que la différence entre les coordonnées (σ_1, σ_2) servant à définir un point de W et celles servant à définir un point de T vaut exactement le couple (s_1, s_2) cherché, alors ces points sont égaux dans la jacobienne. On peut ainsi montrer que le cardinal de l'intersection de W et de T est au moins le quart du cardinal de W. L'algorithme va donc fabriquer une grande quantité de points répartis plus ou moins aléatoirement dans W ainsi que dans T. Au moins un quart devrait tomber dans l'intersection, et ensuite, grâce au paradoxe des anniversaires, on espère obtenir une collision entre un point de T et un point de W après avoir fabriqué une quantité de points en racine carrée du cardinal de W, ce qui donnera bien les valeurs de s_1 et s_2 en une complexité de l'ordre de $q^{3/4}$.

Cette première description n'a pas résolu le problème de mémoire, puisqu'il semble que l'on doive stocker tous les points tirés aléatoirement dans W et T jusqu'à ce qu'une collision se produise. L'approche classique pour éviter ce stockage est l'utilisation d'une marche pseudo-aléatoire et du concept de point distingué. On définit ainsi une fonction f de la jacobienne dans elle-même qui à un point associe un autre point de sorte que si le point de départ est dans W (respectivement de T), on a de bonnes chances d'y rester après l'application de f, et même après plusieurs applications de f. Pour cela, la fonction f va ajouter au point en entrée un petit «saut» de la forme $(\alpha + \beta(q+1))P$, où α et β sont des petits entiers qui dépendent du point que l'on donne en entrée à f. Les points de W et T ne seront donc plus tirés aléatoirement, mais fabriqués en itérant la fonction f à partir de points aléatoires de W et T. Au lieu de stocker tous les points, on va se contenter de stocker uniquement les points ayant une certaine propriété arbitraire (par exemple la représentation en machine de l'élément commence par une douzaine de 0). Ces points sont appelés points distingués, et on suppose que la propriété d'être distingué est complètement indépendante de l'arithmétique de la jacobienne et de la fonction f. Alors, en choisissant bien la propriété de distinction, on peut réduire la taille mémoire en conséquence. La contrepartie est que si l'on avait une collision entre deux éléments, on ne pourra plus la détecter directement : il faudra continuer d'itérer la fonction f pour tomber sur une collision entre deux éléments distingués.

Les détails de réglage de cet algorithme sont assez complexes, nous ne mentionnons ici que quelques-uns des problèmes :

- Définir une fonction f qui se calcule efficacement, qui est suffisamment proche d'une fonction aléatoire pour que le paradoxe des anniversaires se produise, qui peut être itérée suffisamment longtemps avant de sortir de W (resp. de T).
- Définir une propriété de distinction qui se calcule efficacement, qui soit bien indépendante de toutes les autres fonctions impliquées.
- Régler la probabilité d'être distingué pour avoir une taille mémoire raisonnable sans changer notablement le temps de calcul.

En pratique, dès que l'on atteint un point distingué, on va tirer un nouveau point aléatoire dans W (respectivement dans T) comme point de départ pour itérer la fonction f. On obtient alors une série de chaînes de points se terminant par des points distingués que l'on stocke. Ainsi, l'algorithme devient très naturellement parallélisable. Les communications sont réduites à l'envoi de points distingués.

L'algorithme obtenu est heuristique, mais fonctionne très bien en pratique : il ne consomme presque pas de mémoire, même pour des calculs de plusieurs jours, et se parallélise de manière presque parfaite. Ajoutons que cet algorithme a été depuis utilisé dans un autre contexte : Weng l'a utilisé pour le comptage de points sur des courbes de genre 3 sous forme de Picard [174].

3.2.2 Utilisation de l'opérateur de Cartier-Manin

Nous décrivons ici une méthode permettant de calculer le polynôme caractéristique du Frobenius à partir de la matrice de l'opérateur de Cartier-Manin. La première utilisation de cet opérateur dans le contexte du comptage de points se trouve dans [69]. En collaboration avec Bostan et Schost [10], nous avons montré comment réduire la complexité de ce calcul de $\tilde{O}(p)$ à $\tilde{O}(\sqrt{p})$.

Soit $\mathcal{C}$ une courbe hyperelliptique de genre g définie sur un corps fini $\mathbb{F}_{p^d}$ avec p^d éléments, où $p > 2$ est premier. Supposons que $\mathcal{C}$ est donnée par une équation de la forme $y^2 = f(x)$, où f est un polynôme unitaire, sans facteur carré, de degré $2g+1$. La matrice de Hasse-Witt [85] de $\mathcal{C}$ est une généralisation de l'invariant de Hasse d'une courbe elliptique et est définie comme suit. Soit h_k le coefficient de degré k du polynôme $f^{(p-1)/2}$. La matrice de Hasse-Witt est la matrice $g \times g$ à coefficients dans $\mathbb{F}_{p^d}$ donnée par $H = (h_{ip-j})_{1 \leqslant i,j \leqslant g}$. Il s'agit de la matrice, dans une base bien choisie d'un opérateur sur les formes différentielles défini par Cartier [18]. Manin [116] a montré que cette matrice est fortement reliée à l'action de l'endomorphisme de Frobenius sur la p-torsion de la jacobienne de $\mathcal{C}$. L'article de Yui [176] fournit un exposé complet de ces résultats, en particulier le théorème suivant :

Théorème 7 (Manin) *Soit $H_\pi = HH^{(p)} \cdots H^{(p^{d-1})}$, où la notation $H^{(q)}$ signifie l'élévation à la puissance q de chaque coefficient de H. Soit $\kappa(t)$ le polynôme caractéristique de la matrice H_π et $\chi(t)$ le polynôme caractéristique de l'endomorphisme de Frobenius de la jacobienne de $\mathcal{C}$. Alors*

$$\chi(t) \equiv (-1)^g t^g \kappa(t) \mod p.$$

Pour calculer les coefficients de la matrice de Hasse-Witt, la méthode naturelle consiste à développer le produit $f^{(p-1)/2}$. Grâce aux algorithmes de multiplication rapide de polynômes, cela peut se faire en temps essentiellement linéaire en p. En fait il est possible de calculer ces coefficients en une complexité essentiellement en $\sqrt{p}$. Pour simplifier, nous supposerons que le coefficient constant de f est non nul ; sinon le problème est en fait plus simple.

Introduction d'une suite récurrente linéaire. Dans [47], Flajolet et Salvy avaient déjà traité la question de calculer un coefficient particulier d'une grande puissance d'un polynôme, lors d'une réponse à un défi SIGSAM. Le point clef de leur approche est que $h = f^{(p-1)/2}$ vérifie une équation différentielle linéaire du premier ordre

$$fh' - \frac{p-1}{2} f'h = 0.$$

Ceci démontre que les coefficients de h satisfont une relation de récurrence linéaire d'ordre $2g+1$ à coefficients polynomiaux de degré 1. Notons h_k le coefficient de degré k de h ; et posons $h_k = 0$ pour $k < 0$. De manière similaire, on note f_k le coefficient de degré k de f.

Alors l'équation différentielle ci-dessus implique que pour tout k dans $\mathbb{Z}$,

$$(k+1)f_0h_{k+1} + \left(k - \frac{p-1}{2}\right) f_1h_k + \cdots + \left(k - 2g - \frac{(2g+1)(p-1)}{2}\right) f_{2g+1}h_{k-2g} = 0.$$

Posons $U_k = [h_{k-2g}, h_{k-2g+1}, \ldots, h_k]^t$, et notons $A(k)$ la matrice compagnon :

$$A(k) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 \\ \frac{f_{2g+1}((2g+1)(p-1)/2-(k-2g-1))}{f_0 k} & \cdots & \cdots & \cdots & \frac{f_1((p-1)/2-k+1)}{f_0 k} \end{pmatrix}.$$

Le vecteur initial $U_0 = [0, \ldots, 0, f_0^{(p-1)/2}]^t$ peut être calculé par exponentiation binaire en $O(\log p)$ opérations dans le corps de base ; ensuite, pour $k \geqslant 0$ nous avons $U_{k+1} = A(k+1)U_k$. Ainsi pour répondre à la question initiale il suffit de remarquer que le vecteur U_{ip-1} fournit les coefficients h_{ip-j} pour $j = 1, \ldots, g$ qui forment la i-ème ligne de la matrice de Hasse-Witt de $\mathcal{C}$.

Un algorithme dû aux frères Chudnovsky [24] permet de calculer le k-ième vecteur d'une telle suite en temps essentiellement $\sqrt{k}$. Dans notre cas, cela ramène la complexité du calcul de la matrice de Hasse-Witt à $\tilde{O}(\sqrt{p})$ (en supposant fixé tous les paramètres du calcul sauf p). L'algorithme est en fait un compromis temps-mémoire du type «pas de bébé, pas de géant». En collaboration avec Bostan et Schost nous avons montré dans [11] comment améliorer l'algorithme des Chudnovsky dans deux directions. D'une part nous avons réduit la complexité d'un facteur logarithmique, ce qui ne change pas grand chose du point de vue théorique, mais s'avère très appréciable en pratique. Nous avons aussi montré comment calculer simultanément plusieurs vecteurs de la suite sans changer la complexité.

Les détails de l'algorithme des Chudnovsky et des améliorations que nous y avons apportées sont assez techniques. Nous allons nous contenter d'une esquisse des méthodes employées, et pour cela nous commencerons par le cas de la factorielle modulaire, qui est à la base de l'algorithme de factorisation d'entiers de Strassen.

Le cas de la factorielle modulaire. Pour factoriser un entier N, on peut tenter de le diviser par tous les entiers jusqu'à $\sqrt{N}$ pour un coût essentiellement linéaire en $\sqrt{N}$. Pour faire mieux, Strassen [162] a proposé de grouper tous les entiers plus petits que $\sqrt{N}$ en c blocs de c entiers consécutifs, où $c \in \mathbb{N}$ est de l'ordre de $\sqrt[4]{N}$. Écrivons $f_0 = 1 \cdots c \mod N$, $f_1 = (c+1) \cdots (2c) \mod N$, ..., $f_{c-1} = (c^2-c+1) \cdots (c^2) \mod N$. Si les valeurs $f_0, \ldots, f_{c-1}$ peuvent être calculées efficacement, alors trouver un facteur de N devient facile en utilisant les PGCD de $f_0, \ldots, f_{c-1}$ avec N. Ainsi, la difficulté principale réside dans le calcul des valeurs f_i, dont le coût va, de fait, dominer la complexité globale.

Considérons l'anneau $R = \mathbb{Z}/N\mathbb{Z}$, et soit F le polynôme $(X+1) \cdots (X+c) \in R[X]$. La partie «pas de bébé» de l'algorithme consiste à calculer F : en utilisant un algorithme d'arbres de sous-produits [171, Chapter 10] et des algorithmes de multiplications à base de transformée de Fourier rapide, le nombre d'opérations dans R nécessaires peut être réduit à une quantité quasi-linéaire en c. Ensuite les «pas de géant» consistent à évaluer le polynôme F en les points $0, c, \ldots, (c-1)c$, puisque $F(ic) = f_i$. En utilisant des méthodes d'évaluation rapide (là encore,

s'appuyant sur des arbres de sous-produits) ces valeurs peuvent être calculées en un nombre quasi-linéaire en c d'opérations dans R. Ainsi la complexité est de l'ordre de c qui est en $\sqrt[4]{N}$. Plus précisément, si $\mathsf{M}(x)$ désigne le coût de la multiplication de deux polynômes de degré x à coefficients dans R, le coût de l'algorithme de Strassen est de $O(\mathsf{M}(\sqrt[4]{N})\log N)$ opérations dans R.

Indépendamment des questions de factorisation, on constate que la multiplication des valeurs $f_0, f_1, \ldots, f_{c-1}$ donne le produit $1 \cdots c^2$ modulo N. Ainsi, cet algorithme peut être utilisé pour calculer des factorielles : l'analyse ci-dessus montre que dans tout anneau R, pour tout $N \geqslant 0$, le produit $1 \cdots N$ peut être calculé en $O(\mathsf{M}(\sqrt{N})\log N)$ opérations dans R, ce qui améliore l'approche itérative naïve qui a une complexité linéaire en N.

Maintenant, la suite $U_N = N!$ est un exemple basique d'une solution d'une récurrence linéaire à coefficients polynomiaux, à savoir $U_N = NU_{N-1}$. Chudnovsky et Chudnovsky ont ainsi généralisé l'approche «pas de bébé, pas de géant» afin de calculer le N-ième terme d'une récurrence matricielle $n \times n$ à coefficients polynomiaux de degré au plus 1. Les tâches principales sont les mêmes que précédemment. Le calcul de la matrice équivalente au polynôme F peut s'effectuer en $O(\mathsf{MM}(n)\mathsf{M}(\sqrt{N}))$ opérations si $2, \ldots, \lceil\sqrt{N}\rceil$ sont inversibles dans R, et $O(\mathsf{MM}(n)\mathsf{M}(\sqrt{N})\log N)$ sinon, où $\mathsf{MM}(n)$ donne le nombre d'opérations dans R nécessaires pour multiplier deux matrices carrées de taille n à coefficients dans R. Ensuite, l'étape d'évaluation requiert $O(n^2\mathsf{M}(\sqrt{N})\log N)$ opérations. Cela va nous fournir la complexité quasi-linéaire en $\sqrt{p}$ mentionnée ci-dessus.

Nous allons maintenant décrire notre approche dans le cas de la factorielle (le cas matriciel est similaire). La clef est que l'on n'est pas intéressé par les coefficients du polynôme F, mais par ses valeurs en des points particuliers. À la fois F et les points d'évaluation ont une structure particulière : F est le produit de $(X+1)$, $(X+2)$, …, $(X+c)$, tandis que les points d'évaluation forment la progression arithmétique $0, c, \ldots, (c-1)c$. Ceci nous permet de réduire le coût de l'évaluation de F à $O(\mathsf{M}(c))$ au lieu de $O(\mathsf{M}(c)\log c)$. Pour cela, une approche «diviser pour régner» est utilisée, avec comme brique de base l'opération centrale suivante : étant données les valeurs d'un polynôme P sur un progression arithmétique, calculer les valeurs de P sur une translation de cette progression arithmétique.

Dans le cas général d'une récurrence matricielle, le fait de travailler directement avec les valeurs des polynômes permet d'éviter complètement les coûteuses multiplications de matrices de polynômes et de réduire d'un facteur logarithmique les coûts relatifs à l'évaluation multipoint. En contrepartie, cela impose des conditions d'inversibilité dans l'anneau R. Malheureusement, ces conditions ne sont pas anecdotiques, car aussi bien dans le cas de la factorisation que dans le cas de l'opérateur de Cartier-Manin, on travaille dans un anneau qui contient des diviseurs de zéro.

Dans le cas de la factorisation, un problème d'inversibilité se traduit par la découverte d'un facteur, si bien que l'on peut assez facilement s'en sortir et aboutir au résultat suivant, où l'on a noté $\mathsf{M}_{\mathrm{int}}(x)$ le coût d'une multiplication de 2 entiers de x bits.

Théorème 8 *Il existe un algorithme déterministe qui pour tout entier N retourne la factorisation complète de N en temps*

$$O\big(\mathsf{M}(\sqrt[4]{N})\mathsf{M}_{\mathrm{int}}(\log N)\big).$$

Ceci améliore d'un facteur $\log(N)$ la meilleure complexité déterministe de factorisation. Bien entendu, il existe des variantes heuristiques ou probabilistes ayant une complexité bien meilleure.

En ce qui concerne l'opérateur de Cartier-Manin, les problèmes d'inversibilité apparaissent à cause du fait que l'on va devoir diviser par (des puissances de) p, à plusieurs reprises, alors que l'on est en caractéristique p. Pour remédier à cela, on commence par relever la courbe dans un corps p-adique à une précision suffisamment grande pour que les divisions par p ne fassent pas perdre toute l'information. Tout l'algorithme est alors effectué dans les p-adiques, et le résultat final est réduit modulo p. Nous renvoyons à l'article [10] pour les détails sur l'évaluation de la précision nécessaire aux calculs.

Une autre considération est que, lorsque l'on souhaite calculer plusieurs termes d'une suite récurrente linéaire – ce qui est le cas pour l'opérateur de Cartier-Manin – on peut partager énormément de calculs. Ceci se traduit par un coût en mémoire légèrement supérieur. C'est pourquoi dans le théorème suivant, on donne aussi une version moins rapide.

Théorème 9 *Soit $p > 2$ un nombre premier, $d \geqslant 0$ et $\mathcal{C}$ une courbe hyperelliptique définie sur $\mathbb{F}_{p^d}$ par l'équation $y^2 = f(x)$, avec f de degré $2g+1$. En supposant $g < p^{1-\varepsilon}$ pour $0 < \varepsilon < 1$, on peut calculer la matrice de Hasse-Witt de $\mathcal{C}$ en utilisant l'une des deux stratégies suivantes :*

1. *Une stratégie efficace en mémoire donnant une complexité de*

$$O\left(g^{1+\log 7} p^{\frac{1}{2}} \mathsf{M}_{\mathrm{int}}(dg\log(dp)) \;+\; g^3 \mathsf{M}_{\mathrm{int}}(dgp^{\frac{1}{2}}\log(dp)) \;+\; dg^4 p^{\frac{1}{2}} \log g \log p\right)$$

   opérations binaires et $O\left(dg^3 p^{\frac{1}{2}} \log p + dgp^{\frac{1}{2}} \log d\right)$ en espace.

2. *Une stratégie efficace en temps donnant une complexité de*

$$O\left(g^{\frac{1}{2}+\log 7} p^{\frac{1}{2}} \mathsf{M}_{\mathrm{int}}(dg\log(dp)) \;+\; g^2 \mathsf{M}_{\mathrm{int}}(dg^{\frac{3}{2}} p^{\frac{1}{2}}\log(dgp)) \;+\; dg^{\frac{7}{2}} p^{\frac{1}{2}} \log g \log p\right)$$

   opérations binaires et $O\left(dg^{\frac{7}{2}} p^{\frac{1}{2}} \log p + dg^{\frac{3}{2}} p^{\frac{1}{2}} \log d\right)$ en espace.

La matrice H donne directement des informations sur la courbe $\mathcal{C}$: par exemple, H est inversible si et seulement si la jacobienne de $\mathcal{C}$ est ordinaire [176, Corollary 2.3]. Cependant, comme mentionné dans le théorème 7, la matrice H_π et en particulier son polynôme caractéristique κ donnent beaucoup plus dans un contexte de comptage de points. Ces calculs finaux sont en pratique bien plus simple que le calcul de H.

Combinaison avec d'autres algorithmes de comptage de points. Calculer le polynôme caractéristique de H_π n'est pas suffisant pour déduire le cardinal de la jacobienne de $\mathcal{C}$, car seul $\chi \bmod p$ est calculé. Pour terminer le calcul, on fait appel à d'autres algorithmes.

Notons tout d'abord que dans le cas où p est petit comparé à g ou d, alors les algorithmes p-adiques dont nous parlerons plus tard sont utilisés. Ces méthodes calculent χ modulo des puissances de p, si bien que l'information que l'on obtiendrait via l'opérateur de Cartier-Manin est de toute façon recalculée.

Considérons ensuite les extensions de l'algorithme de Schoof, dont nous parlerons plus précisément à la section suivante. Celles-ci étant d'une complexité polynomiale, elles rendent en théorie inutile le calcul de l'opérateur de Cartier-Manin, ou du moins d'un effet asymptotiquement négligeable. Cela dit, à notre connaissance, il n'existe pas d'implantation de l'algorithme de Schoof pour les courbes de genre au moins 3, et même pour le genre 2, les records sont obtenus en combinant l'algorithme de Schoof avec des méthodes de complexité exponentielle,

comme la recherche probabiliste de collisions de la section 3.2.1 ou la méthode d'approximation, qui consiste à calculer le nombre de points de $\mathcal{C}$ sur les extensions de petit degré de $\mathbb{F}_{p^d}$ afin de réduire l'espace de recherche.

Pour les courbes de genre 2 et 3 sur des corps de la forme $\mathbb{F}_{p^d}$ avec d petit, l'utilisation de l'opérateur de Cartier-Manin permet de réduire la complexité (exponentielle) de la combinaison de toutes les méthodes, exceptée l'algorithme de Schoof.

D'un point de vue pratique, on peut aussi combiner avec l'algorithme de Schoof. Grâce à notre algorithme, nous sommes ainsi parvenus à calculer le polynôme caractéristique de la jacobienne d'une courbe de genre 2 définie sur $\mathbb{F}_{p^3}$, avec $p = 2^{32} - 5$.

L'algorithme décrit dans cette section à été ensuite repris par Harvey [84] qui l'a incorporé à l'algorithme de Kedlaya, de manière à obtenir une complexité en p de l'ordre de $\sqrt{p}$ dans ce cas.

3.3 L'algorithme de Schoof et ses généralisations

L'algorithme de Schoof [145], inventé en 1985, fut le premier algorithme de comptage de points de complexité polynomial. Initialement décrit pour les courbes elliptiques, nous en donnerons tout d'abord une version plus générale, qui sera ensuite détaillée dans le cas elliptique et dans le cas du genre 2.

3.3.1 Un schéma d'algorithme de Schoof générique

Soit A une variété abélienne de dimension g sur le corps fini $\mathbb{F}_q$ pour laquelle on cherche à calculer le polynôme caractéristique $\chi_\pi(t)$ de l'endomorphisme de Frobenius.

Soit ℓ un nombre premier différent de la caractéristique du corps de base. Alors, le groupe des points de ℓ-torsion de A, noté $A[\ell]$ a une structure de $\mathbb{Z}/\ell\mathbb{Z}$-espace vectoriel de dimension $2g$, sur lequel l'endomorphisme de Frobenius agit de manière $\mathbb{Z}/\ell\mathbb{Z}$-linéaire. Le polynôme caractéristique de cette restriction de l'endomorphisme de Frobenius est alors $\chi_\pi(t)$ modulo ℓ.

L'algorithme de Schoof et ses variantes consistent à expliciter cette action de l'endomorphisme de Frobenius sur la ℓ-torsion de manière à déduire une partie de $\chi_\pi(t)$. On répète ensuite cette opération pour différentes valeurs de ℓ jusqu'à ce que les bornes théoriques sur les coefficients de $\chi_\pi(t)$ permettent de conclure par le théorème des restes chinois. Le nombre de chiffres du plus grand des coefficients de $\chi_\pi(t)$ est borné par une constante fois $g \log q$; d'après le théorème des nombres premiers, les nombres premiers ℓ à considérer sont donc de taille $O(g \log q)$ et en nombre $O(g \log q)$.

Le caractère exponentiel en la dimension g vient de la difficulté à manipuler $A[\ell]$. En effet, $A[\ell]$ contient ℓ^{2g} points. Pour décrire de manière concise ce nombre exponentiel de points, on pourrait rêver d'exploiter la structure d'espace vectoriel sur $\mathbb{Z}/\ell\mathbb{Z}$ et ne calculer qu'avec $2g$ points formant une base de la ℓ-torsion. Toutefois ce projet ne peut aboutir pour des raisons de corps de définition. En effet, $A[\ell]$ est défini sur $\mathbb{F}_q$ dans son ensemble (c'est le noyau de la multiplication par ℓ qui est un morphisme $\mathbb{F}_q$-rationnel), mais les éléments eux-mêmes sont en général définis sur une grande extension de $\mathbb{F}_q$. De fait, connaître la plus petite extension sur laquelle on peut trouver une base de $A[\ell]$ est fortement relié à la connaissance de $\chi_\pi(t)$ modulo ℓ qui est précisément ce que l'on cherche à calculer.

Ainsi, les approches directes pour décrire $A[\ell]$ vont requérir la donnée de ℓ^{2g} points définis *dans leur ensemble* sur $\mathbb{F}_q$, ce qui se traduira au minimum par un polynôme de degré environ

ℓ^{2g} à coefficients dans $\mathbb{F}_q$. La taille de l'objet considéré est donc de l'ordre de $\ell^{2g} \log q$.

Une fois que l'on dispose d'une description algorithmique de $A[\ell]$, il s'agit de trouver le polynôme caractéristique de l'action de π sur cet espace vectoriel. Une opération qu'il paraît difficile d'éviter est le calcul effectif de l'image par π des points de $A[\ell]$. Ceci se traduit par un calcul du type $X^q \bmod f(X)$, où $f(X)$ est un polynôme dont les racines décrivent les points de $A[\ell]$. Par une méthode d'exponentiation binaire, ceci coûte $O(\log q)$ opérations polynomiales modulo le polynôme $f(X)$, se qui fait donc au total $O(\log q\, \mathsf{M}(\ell^{2g} \log q))$. Notons que ce calcul ne donne pas $\chi_\pi(t)$ modulo ℓ et qu'il reste encore du travail d'algèbre linéaire. Toutefois, cette première étape peut-être vue comme la plus coûteuse.

Si l'on résume, l'algorithme de Schoof procède schématiquement ainsi

1. Tant que l'on n'a pas assez d'information modulaire sur $\chi_\pi(t)$, faire :
   (a) Choisir un nouveau ℓ premier différent de la caractéristique ;
   (b) Calculer une description effective de $A[\ell]$;
   (c) Calculer l'action de π sur $A[\ell]$ et en déduire $\chi_\pi(t)$ modulo ℓ ;
2. Reconstruire $\chi_\pi(t)$ par le théorème des restes chinois.

D'après la discussion précédente, la boucle va être effectuée $O(g \log q)$ fois avec des premiers ℓ de taille $O(g \log q)$. Le point 1.(b) nécessite de calculer un objet de taille de l'ordre de $\ell^{2g} \log q$, on s'attend donc *au mieux* à une complexité de $O(\ell^{2g} \log q)$ pour cette étape. Pour l'étape 1.(c), nous avons vu que l'on s'attend à un coût au mieux en $O(\log q\, \mathsf{M}(\ell^{2g} \log q)) = \tilde{O}(\ell^{2g} \log^2 q)$. Ainsi, en étant très optimiste, l'algorithme schématique de Schoof pourrait atteindre une complexité de $\tilde{O}(g^{1+2g} (\log q)^{2g+3})$.

Dans le cas des courbes elliptiques, cette complexité est effectivement atteinte, puisque l'algorithme original de Schoof a un temps de calcul borné par $\tilde{O}((\log q)^5)$. Dans le cas du genre 2, la complexité idéale serait $\tilde{O}((\log q)^7)$, mais le meilleur algorithme connu a une complexité de $\tilde{O}((\log q)^8)$, à cause principalement de la difficulté à calculer une description effective de $A[\ell]$.

Quoiqu'il en soit, la nature exponentielle en le genre de cet algorithme résidant principalement dans la taille de $A[\ell]$, on peut chercher à ne travailler que dans un sous-espace de dimension inférieure, lorsqu'il en existe un. Cela n'est pas sans rappeler les améliorations apportées par Atkin et Elkies à l'algorithme de Schoof dans le cas elliptique que nous allons maintenant rappeler rapidement.

3.3.2 Les courbes elliptiques : l'algorithme de Schoof et les améliorations d'Atkin et Elkies

Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ d'équation $y^2 = x^3 + ax + b$ (on suppose la caractéristique au moins 5, pour simplifier). Pour tout $\ell \geqslant 3$ premier, il existe un polynôme ψ_ℓ dans $\mathbb{F}_q[x]$, appelé *polynôme de division* tel que

1. Pour tout point non nul $P = (x, y)$ de E, $\psi_\ell(x) = 0$ si et seulement si $\ell \cdot P = 0$.
2. Si ℓ est premier à la caractéristique, le degré de ψ_ℓ est $(\ell^2 - 1)/2$.

Le calcul des polynômes de division ne pose pas de problème : des formules récurrentes permettent de calculer le polynôme ψ_ℓ par une méthode similaire à l'exponentiation binaire en temps $O(\mathsf{M}(\ell^2))$ opérations dans le corps de base.

Ainsi dans le cas des courbes elliptiques, la situation est idéale, puisque l'on peut calculer une représentation de $E[\ell]$ en temps quasi-optimal.

Soit $\mathcal{A}_\ell$ l'algèbre $\mathbb{F}_q[x,y]/(\psi_\ell(x), y^2-(x^3+ax+b))$. Le point de coordonnées (x,y) dans $\mathcal{A}_\ell$ est le point de ℓ-torsion non nul générique de E. Le polynôme $\chi_\pi(t)$ est de la forme t^2-ct+q, où l'entier c est appelé la trace de E. Ainsi, l'égalité

$$\pi^2(x,y) - (c \bmod \ell)\cdot\pi(x,y) + (q \bmod \ell)\cdot(x,y) = 0,$$

est vérifiée dans l'algèbre $\mathcal{A}_\ell$. Plus précisément, on peut montrer que cette équation n'est pas vraie si l'on met une autre valeur à la place de $(c \bmod \ell)$. Par la méthode d'exponentiation binaire, le calcul de $\pi(x,y) = (x^q, y^q)$ et de $\pi^2(x,y) = (x^{q^2}, y^{q^2})$ dans $\mathcal{A}_\ell$ coûte $O(\log q)$ opérations dans $\mathcal{A}_\ell$. Une fois ces quantités obtenues, on peut tester la validité de l'équation caractéristique pour les différentes valeurs possibles de $(c \bmod \ell)$ au prix de $O(\ell)$ opérations supplémentaires dans $\mathcal{A}_\ell$. Comme une opération dans $\mathcal{A}_\ell$ a une complexité de $\tilde{O}(\ell^2 \log q)$, on obtient la valeur de $\chi_\pi(t)$ modulo ℓ en temps $O((\ell+\log q)\ell^2 \log q)$. Comme $|c| \leqslant 2\sqrt{q}$, on doit traiter $O(\log q)$ nombres premiers ℓ différents, le plus grand étant de l'ordre de $\log q$. Finalement le temps de calcul total de l'algorithme de Schoof est de $\tilde{O}((\log q)^5)$.

On a affirmé que l'équation caractéristique sur $\mathcal{A}_\ell$ n'était vérifiée que pour la vraie valeur de $(c \bmod \ell)$. Lorsque ℓ est premier, ce qui est vrai pour le point générique reste vrai même si l'on ne considère qu'un seul point P non nul de $E[\ell]$. En effet, s'il existe c et c' deux entiers tels que $\pi^2(P) + c\cdot\pi(P) + q\cdot P = \pi^2(P) + c'\cdot\pi(P) + q\cdot = 0$, alors on a immédiatement $(c-c')\cdot\pi(P) = 0$, ce qui implique $(c-c')\cdot P = 0$, et donc $c \equiv c'$ modulo ℓ. Ainsi, on peut récupérer toute l'information souhaitée même si l'on ne travaille que dans un sous-espace de $E[\ell]$.

Les sous-groupes de $E[\ell]$ sont en correspondance avec les isogénies de domaine E, et celles-ci sont paramétrées par les équations modulaires. D'où l'entrée en scène de $\Phi_\ell(X,Y)$, le polynôme modulaire de degré ℓ. Soit j l'invariant modulaire de la courbe E. Alors le polynôme $\Phi_\ell(X,j)$ est un polynôme de degré $\ell+1$ sur $\mathbb{F}_q$; à chacune de ses racines j^* correspond une courbe E^* qui est ℓ-isogène à E. Le noyau d'une telle isogénie est un sous-groupe d'ordre ℓ de $E[\ell]$, et réciproquement à tout sous-groupe d'ordre ℓ de $E[\ell]$ on peut associer une racine de $\Phi_\ell(X,j)$. On a ainsi des correspondances bijectives

$$\text{racines de } \Phi_\ell(X,j) \quad\leftrightarrow\quad \ell\text{-isogénies } E \to E^* \quad\leftrightarrow\quad \text{sous-groupes d'ordre } \ell \text{ de } E[\ell].$$

De plus ces correspondances respectent la rationalité : l'extension de $\mathbb{F}_q$ définie par la racine considérée est exactement le corps de définition de l'isogénie et du sous-groupe correspondant (sauf dans certains cas facilement détectables où $\Phi_\ell(X,j)$ a des racines multiples). L'algorithme SEA, pour Schoof-Elkies-Atkin, procède donc ainsi pour chaque ℓ :

1. Calculer $\Phi_\ell(X,j)$;
2. Décider si ce polynôme a une racine dans $\mathbb{F}_q$ et si oui, en exhiber une que l'on note j^* ;
3. Calculer le facteur $g_\ell(X)$ de $\psi_\ell(X)$ dont les racines sont les abscisses des points du sous-groupe de $E[\ell]$ correspondant à j^* ;
4. Calculer l'action de π sur $E[\ell]$ en vérifiant l'équation caractéristique du Frobenius dans l'algèbre $\mathcal{B}_\ell = \mathbb{F}_q[x,y]/(g_\ell(x), y^2-(x^3+ax+b))$;

L'avantage de cet algorithme par rapport au précédent est que $\Phi_\ell(X,j)$ et g_ℓ sont des polynômes de degré $O(\ell)$, à comparer au degré $O(\ell^2)$ pour ψ_ℓ. Discutons brièvement chaque étape. En premier lieu, il s'agit de calculer le polynôme modulaire instancié en j. La meilleure méthode connue actuellement (du moins si le corps de base est un corps premier) est de

commencer par calculer le polynôme $\Phi_\ell(X,Y)$ sur $\mathbb{Z}$, ce qui coûte asymptotiquement $\tilde{O}(\ell^3)$, de le réduire modulo p et d'instancier Y, ce qui coûte $\tilde{O}(\ell^2(\ell+\log q))$. Évidemment, on peut considérer que l'obtention de $\Phi_\ell(X,Y)$ est un précalcul, mais du point de vue de la complexité cela n'est pas strictement nécessaire. Dans l'étape 2, le calcul dominant est celui de $X^q \bmod \Phi_\ell(X,j)$ dont la complexité est $\tilde{O}(\ell(\log q)^2)$. Notons que les motifs de factorisation de $\Phi_\ell(X,j)$ sont extrêmement contraints et l'on renvoie à [146] pour plus de détails. La partie 3 est une partie délicate en effet, l'algorithme utilisé en grande caractéristique utilise un relèvement vers $\mathbb{C}$ de manière à pouvoir considérer les formes modulaires et les fonctions elliptiques associées aux objets algébriques. Nous renvoyons de nouveau à [146] pour ces questions. Des identifications des coefficients de développements en série de Fourier fournissent alors des identités qui sont ensuite réduites sur le corps fini $\mathbb{F}_q$ de départ, ce qui permet de calculer g_ℓ en $O(\ell^2)$ opérations dans $\mathbb{F}_q$, sans avoir à calculer ψ_ℓ. Dans le cas où la caractéristique est petite, cette approche échoue car les coefficients de Fourier ont des dénominateurs jusqu'à $O(\ell)$, et donc pour pouvoir les réduire il est nécessaire d'avoir $p \gg \ell$. Des algorithmes dus à Couveignes et Lercier [26, 110, 112] permettent alors de s'en sortir. La dernière étape se déroule comme dans l'algorithme de Schoof original, mais avec une algèbre plus petite. La complexité est de $\tilde{O}((\ell+\log q)\ell)$ opérations dans $\mathbb{F}_q$.

Ainsi, pour tout ℓ tel que $\Phi_\ell(X,j)$ ait une racine dans $\mathbb{F}_q$, l'algorithme SEA permet de calculer $\chi_\pi(t)$ modulo ℓ en temps $\tilde{O}((\ell+\log q)\ell\log q)$. Heuristiquement, pour une courbe aléatoire, on s'attend à ce que la moitié des ℓ aient cette propriété, si bien que la complexité totale de l'algorithme SEA est de $\tilde{O}((\log q)^4)$. Cette complexité est non prouvée mais reflète bien ce que l'on observe en pratique.

3.3.3 Les courbes de genre supérieur

La tâche première lorsque l'on veut étendre l'algorithme de Schoof aux courbes de genre supérieur est de trouver une représentation des éléments de $A[\ell]$ qui permette de calculer efficacement l'action de Frobenius. Pour cela, on va utiliser une représentation des variétés en question sous forme d'idéaux. Géométriquement, $A[\ell]$ est une variété algébrique de dimension 0 et de degré ℓ^{2g}. Lorsque A est la jacobienne d'une courbe de genre g, on peut représenter les éléments génériques de A par $2g$ coordonnées, grâce à l'application du produit symétrique $\mathcal{C}^{(g)}$ vers A ; en effet, il suffit alors de prendre les fonctions symétriques de deux coordonnées de chaque point d'un modèle affine plan de $\mathcal{C}$. En utilisant cette représentation par $2g$ coordonnées, on peut décrire $A[\ell]$ par l'idéal radical des polynômes en $2g$ variables s'annulant exactement en les points de $A[\ell]$.

Si l'on applique ce programme aux courbes elliptiques, on retrouve précisément l'idéal engendré par $\psi_\ell(x)$ et $y^2-(x^3+ax+b)$ qui a servi à définir l'algèbre $\mathcal{A}_\ell$ utilisée dans l'algorithme de Schoof original.

Considérons maintenant le cas où A est la jacobienne d'une courbe de genre $g>1$, dont un ouvert U est représenté par des coordonnées $(x_1,\ldots,x_{2g})$. Alors l'intersection de U et de $A[\ell]$ est une variété de dimension 0, de degré $\ell^{2g}-\varepsilon$, où ϵ est le nombre de points de $A[\ell]$ que l'on ne peut pas représenter avec nos coordonnées. Par exemple dans le cas elliptique, le point 0 est le seul qui n'est pas pris en compte dans l'algèbre $\mathcal{A}_\ell$, donc $\varepsilon=1$. Notons $(x_1^{(i)},\ldots,x_{2g}^{(i)})$ les coordonnées du i-ème point de $A[\ell]\cap U$, pour i allant de 1 à $\ell^{2g}-\varepsilon$. Alors sous l'hypothèse que x_1 est une coordonnée séparante pour ces points, l'idéal est engendré par des polynômes

que l'on peut mettre sous la forme

$$I_\ell = \begin{cases} P(X_1) = \prod_{i=1}^{\ell^{2g}-\varepsilon}(X_1 - x_1^{(i)}) \\ X_2 - P_2(X_1) \\ \vdots \\ X_{2g} - P_{2g}(X_1), \end{cases}$$

où pour tout k, P_k est le polynôme de degré inférieur à $\ell^{2g} - \varepsilon$ tel que pour tout i, $x_k^{(i)} = P_k(x_1^{(i)})$. Avec notre hypothèse, il s'agit de simples polynômes interpolateurs de Lagrange. Une autre manière de dire les choses est de parler de base de Gröbner réduite pour l'ordre lexicographique.

La question de la rationalité se pose alors. La variété A_ℓ est définie sur $\mathbb{F}_q$. Les coordonnées $(x_1, \ldots, x_{2g})$ sont supposées choisies de telle sorte qu'elles respectent la rationalité : si un point de U est défini sur une extension $\mathbb{F}_{q^k}$, alors ses coordonnées dans ce système sont des éléments de $\mathbb{F}_{q^k}$. De plus on suppose que le complémentaire de U est rationnel. C'est en général automatique : si un point de A n'est pas représentable par les coordonnées x_i, alors ses conjugués ne le sont pas non plus. Dans ces conditions, l'idéal I_ℓ décrit une variété qui est elle-même définie sur $\mathbb{F}_q$, et donc les polynômes P et P_k sont définis sur $\mathbb{F}_q$.

Une fois l'idéal I_ℓ calculé, on peut créer un point de torsion générique en considérant l'algèbre quotient associée. Il ne reste plus qu'à calculer l'action de l'endomorphisme de Frobenius sur ce point de torsion générique pour déduire $\chi_\pi(t)$ modulo ℓ, comme dans l'algorithme de Schoof elliptique.

Quelques complications apparaissent. Tout d'abord, on s'attend génériquement à ce que ε reste sous contrôle. Typiquement, le point 0 est souvent exclus par le système de coordonnées affine car situé à l'infini, comme dans le cas elliptique, mais c'est le seul qui n'est pas pris en compte dans l'idéal I_ℓ. Toutefois rien ne garantit que l'on soit toujours dans ce cas typique : si l'on n'a pas de chance, ou si l'on a fait un choix de coordonnées particulièrement malheureux, il est possible que $A[\ell] \cap U$ soit vide. Par la suite, même si l'idéal I_ℓ capture suffisamment d'information, il est envisageable que lors des calculs dans l'algèbre quotient on en vienne à vouloir représenter dans cette algèbre un point qui n'est pas dans U (issu par exemple de l'application de la loi de groupe dans A). Cette fois-ci cela se traduirait par des divisions par zéro. Là encore, on s'attend à ce que l'ouvert U couvre suffisamment A pour que cela n'arrive que très rarement, mais on ne peut pas l'exclure a priori.

D'un point de vue pratique, on peut éluder la question : si dans une exécution d'un programme on tombe sur un des canulars précités, on peut aisément le détecter et retenter le calcul après un changement aléatoire de coordonnées. D'un point de vue théorique, c'est un tout autre problème. Prouver qu'une randomisation de ce type ou tout autre stratégie dynamique va effectivement fonctionner et réussir à estimer le temps de calcul moyen n'est pas simple. Si de plus on souhaite rester dans le monde déterministe, la situation est encore plus délicate. Nous ne rentrerons pas plus avant dans les détails et renvoyons le lecteur intéressé aux articles sur le sujet [136, 89, 1].

Le cas du genre 2. Afin d'illustrer les propos ci-dessus, nous allons développer le cas des courbes de genre 2, que nous avons plus particulièrement étudié. Soit donc $\mathcal{C}$ une courbe d'équation $y^2 = f(x)$ sur le corps fini $\mathbb{F}_q$ de caractéristique impaire, où f est un polynôme de degré 5, sans facteur carré, que l'on peut supposer unitaire sans perte de généralité. Les $2g = 4$

coordonnées que nous utiliserons sont les coefficients des polynômes $u(x) = x^2 + u_1 x + u_0$ et $v(x) = v_1 x + v_0$ de la représentation de Mumford, dans le cas générique où u est de degré 2. Plutôt que $(x_1, \ldots, x_4)$, on utilise les notations (u_1, u_0, v_1, v_0) pour les coordonnées d'un élément générique de $U \subset \mathrm{Jac}(\mathcal{C})$. Ces coordonnées ne remplissent pas les conditions énoncées dans le cadre général, car la coordonnée u_1 n'est manifestement pas séparante : dans la représentation de Mumford, un élément et son opposé dans $\mathrm{Jac}(\mathcal{C})[\ell]$ ont même polynôme u et des polynômes v opposés. Toutefois, c'est génériquement la seule obstruction, et on se retrouve donc dans une situation très similaire au cas des courbes elliptiques, où l'on a préféré travailler avec le polynôme de division en x, au prix de manipuler un polynôme de degré 2 en y. Ici on cherche donc à calculer un idéal I_ℓ de la forme

$$I_\ell = \begin{cases} P_1(u_1) \\ u_0 - P_2(u_1) \\ v_1^2 - P_3(u_1) \\ v_0 - v_1 P_4(u_1), \end{cases}$$

où P_1 est génériquement de degré $(\ell^4 - 1)/2$, et les polynômes P_2, P_3, P_4 sont de degré inférieur au degré de P_1.

Le calcul de I_ℓ s'effectue grâce aux polynômes de division de Cantor qui sont des polynômes univariés de degré $O(\ell^2)$. Soit $P = (x, y)$ un point de $\mathcal{C}$ que l'on plonge dans $\mathrm{Jac}(\mathcal{C})$ à l'aide du point à l'infini. L'élément de $\mathrm{Jac}(\mathcal{C})$ obtenu après multiplication par ℓ de la classe du diviseur $P - \infty$ admet une représentation de Mumford dont les coordonnées (u_1, u_0, v_1, v_0) sont des fractions rationnelles en x et y. Ces fractions rationnelles s'expriment directement à l'aide des polynômes de division de Cantor [15], dont le calcul se fait par des formules de récurrence assez simples à utiliser. Une fois ces fractions rationnelles calculées, l'idéal I_ℓ se déduit par le cheminement suivant : soit $D = P_1 + P_2 - 2\infty$ un diviseur de degré 0 sur $\mathcal{C}$ correspondant à une représentation de Mumford avec $\deg u = 2$. Alors D représente un élément de ℓ-torsion si et seulement si $\ell \cdot (P_1 - \infty) = -\ell \cdot (P_2 - \infty)$, cette égalité ayant lieu dans $\mathrm{Jac}(\mathcal{C})$, on la transcrit en terme de représentations de Mumford, ce qui donne des équations algébriques que doivent vérifier les coordonnées de P_1 et P_2. Ces dernières sont elles-mêmes reliées aux coordonnées de Mumford de D, si bien qu'à l'aide de résultants on parvient à éliminer les variables de manière à trouver les polynômes P_1, P_2, P_3, P_4.

La meilleure façon connue d'organiser ces calculs [74] a une complexité de $\tilde{O}(\ell^6)$ opérations dans $\mathbb{F}_q$, ce qui est quelque peu décevant, car l'objet en sortie est de taille $O(\ell^4 \log q)$. Ceci explique en partie le fait que l'algorithme de Schoof en genre 2 n'est pas aussi performant que l'on pourrait l'espérer. Le calcul de l'action de l'endomorphisme de Frobenius dans l'algèbre quotient associée à I_ℓ est ensuite un ordre de grandeur de complexité plus facile, si bien que le coût total de l'algorithme de Schoof en genre 2 est en $\tilde{O}(\log^8 q)$.

Une version de cet algorithme a été implanté par l'auteur au sein du logiciel Magma [8]. Une autre version [63] a été implantée en C++ sur la base de la bibliothèque NTL [151]. On y trouve en particulier une fonction qui calcule exactement l'idéal I_ℓ tel que décrit dans cette section. Une version plus récente (pas encore distribuée) nous a permis de calculer le ℓ-torsion d'une courbe de genre 2 jusqu'à $\ell = 31$, obtenant ainsi un calcul de cardinalité record[1] d'une courbe sur le corps premier $\mathbb{F}_{2^{127}-1}$.

[1] Les détails sur ce record sont disponibles à l'adresse `http://www.loria.fr/~gaudry/record127/`.

Pour aller plus loin. Comme dit précédemment, le problème principal de l'algorithme de Schoof en genre 2 est le temps de calcul de l'idéal I_ℓ. Une piste naturelle à explorer est de mimer l'algorithme SEA afin de calculer un idéal plus petit correspondant à un sous-groupe de $\mathrm{Jac}(\mathcal{C})[\ell]$. Pour cela, un point de passage obligé semble être les équations modulaires et leurs analogues en genre supérieur.

Dans [76], avec Schost, nous avons défini des équations modulaires de la manière suivante. Soit $D = \langle x^2+u_1x+u_0, v_1x+v_0\rangle$ le diviseur de torsion générique défini sur l'algèbre de torsion $\mathcal{A}_\ell = \mathbb{F}_q[u_1,u_0,v_1,v_0]/I_\ell$. On définit l'élément t_ℓ de $\mathcal{A}_\ell$ comme la somme des coordonnées u_1 des diviseurs $[i]D$ pour $i = 1,\ldots,(\ell-1)/2$. Comme D est un élément de ℓ-torsion et comme u_1 est le même pour un diviseur et son opposé, l'élément t_ℓ ne dépend que du sous-groupe engendré par D. Par la correspondance déjà évoquée entre sous-groupes et isogénies, on peut considérer que t_ℓ est un paramètre pour les ℓ-isogénies. Le polynôme minimal de t_ℓ dans $\mathcal{A}_\ell$ est un polynôme génériquement de degré $(\ell^4-1)/(\ell-1)$ à coefficients dans $\mathbb{F}_q$. C'est un polynôme modulaire, qui généralise le polynôme Φ_ℓ. Notons d'ailleurs que si l'on effectue cette construction en genre 1, on retrouve des polynômes introduits par Charlap, Coley et Robbins [22].

Malheureusement, la manière la plus rapide connue pour calculer ces équations modulaires passe par le calcul préalable de l'idéal I_ℓ, donc aucun gain en complexité n'est possible pour le moment dans cette direction.

D'autres équations modulaires sont calculées par Dupont dans [36] mais pour l'instant cela reste pour des ℓ très petits. L'approche de Dupont a été reprise récemment par Bröker et Lauter [13], mais cela reste pour l'instant assez théorique.

Un autre passage obligé pour étendre l'algorithme SEA au genre supérieur est le calcul explicite d'isogénies. En genre 1, on dispose d'algorithmes efficaces pour cela, mais qui ne s'adaptent pas de manière triviale ; toutefois des progrès récents ont été effectués [12] si bien que les connaissances s'étendent. En genre supérieur, les premiers développements sont en train d'apparaître : Smith [157] a montré récemment comment construire explicitement une isogénie entre la jacobienne d'une courbe de genre 3 hyperelliptique et une jacobienne de courbe de genre 3 non-hyperelliptique. De plus des formules utilisant les fonctions Théta sont étudiées par Lubicz et Robert pour rendre explicites des isogénies entre jacobiennes de courbes de genre 2.

Genre supérieur à 2. Concernant le genre supérieur, il serait intéressant d'étudier quel est le moyen le plus rapide pour calculer l'idéal I_ℓ dans le cas des courbes de genre 3 ou 4, et de tester quelles sont les ℓ-torsion calculables en pratique. À notre connaissance, personne ne s'est encore attaqué à formuler des versions implantables des algorithmes théoriques de [136, 89, 1]. Dans le cas hyperelliptique, les polynômes de division de Cantor peuvent aider, et dans certains autres cas particuliers (comme les courbes de Picard), on peut vraisemblablement trouver des analogues aux polynômes de division de Cantor.

3.4 L'algorithme de Satoh et ses généralisations

Après l'algorithme de Schoof, nous débutons maintenant la description de l'autre grande classe d'algorithmes de complexité polynomiale, à savoir les algorithmes p-adiques. Dorénavant, on considère des corps finis de la forme $\mathbb{F}_{p^n}$, avec p qui sera toujours relativement petit.

3.4.1 Extensions non ramifiées de $\mathbb{Q}_p$

Soit $\mathbb{Q}_p$ le corps des nombres p-adiques, et $\mathbb{Z}_p$ l'anneau des entiers p-adiques. Soit n un entier supérieur à 1. À isomorphisme près, il existe une unique extension non ramifiée $\mathbb{Q}_p$ de degré n, que l'on note $\mathbb{Q}_q$. L'anneau des entiers de $\mathbb{Q}_q$ est un anneau local dont le corps résiduel est isomorphe au corps fini $\mathbb{F}_q$ à $q = p^n$ éléments. Il s'agit de l'anneau $W(\mathbb{F}_q)$ des vecteurs de Witt sur $\mathbb{F}_q$. Dans la suite nous noterons cet anneau $\mathbb{Z}_q$, pour bien marquer le fait qu'algorithmiquement nous utiliserons une représentation similaire à celle des éléments de $\mathbb{F}_q$ et non pas la construction théorique décrite par exemple dans [149].

Soit $\overline{P}(x)$ un polynôme unitaire irréductible de degré n sur $\mathbb{F}_p$ dont on se sert pour définir l'extension $\mathbb{F}_q$. Soit $P(x)$ un polynôme unitaire à coefficients dans $\mathbb{Z}_p$ dont la réduction modulo p est le polynôme $\overline{P}(x)$. Le polynôme $P(x)$ est irréductible sur $\mathbb{Q}_p$ et on peut définir $\mathbb{Q}_q$ comme étant le quotient $\mathbb{Q}_p[x]/(P(x))$, et son anneau des entiers $\mathbb{Z}_q$ est alors le quotient $\mathbb{Z}_p[x]/(P(x))$.

Une fois qu'un relèvement $P(x)$ de $\overline{P}(x)$ a été choisi, les calculs dans $\mathbb{Z}_q$ se font de manière simple : si l'on veut travailler à précision k, on effectue les opérations dans $(\mathbb{Z}/p^k\mathbb{Z})[x]/(P(x))$. Cela rentre dans le cadre des objets algébriques que l'on peut multiplier par la méthode de Kronecker-Schönhage et donc on supposera que le temps requis pour un calcul élémentaire dans $\mathbb{Z}_q$ à précision k est de l'ordre de $\tilde{O}(nk \log p)$.

Le groupe de Galois de $\mathbb{Q}_q$ sur $\mathbb{Q}_p$ est cyclique d'ordre n et un générateur σ peut-être choisi de sorte que l'action sur le corps résiduel $\mathbb{F}_q$ héritée de l'action de σ est l'automorphisme de Frobenius $x \mapsto x^p$. L'automorphisme σ est alors aussi appelé automorphisme de Frobenius.

Il y a deux manières naturelles de choisir le relevé $P(x)$ de $\overline{P}(x)$ pour définir $\mathbb{Z}_q$.

1. Chaque coefficient de $\overline{P}(x)$ est relevé en l'entier de $[0, p-1]$ correspondant. L'intérêt de ce choix est que $P(x)$ est alors un polynôme ayant de petits coefficients.
2. On tente de garder la structure Galoisienne aussi simple (algorithmiquement) que possible. Il est clair que sur $\mathbb{Z}_q$ l'automorphisme de Frobenius n'agit pas simplement par élévation à la puissance p. Toutefois, si l'on choisit $P(x)$ tel qu'il divise $x^q - x$, l'élément x dans $\mathbb{Z}_p[x]/(P(x))$ est une racine $(q-1)$-ème de l'unité, et donc $\sigma(x) = x^p$. Ce choix de $P(x)$ est toujours possible, car la divisibilité recherchée est vraie dans $\mathbb{F}_p[x]$ et peut se relever par un procédé de type lemme de Hensel.

Quel que soit le choix de $P(x)$, les opérations d'anneau s'effectuent sans problème ; le premier choix permettant de rendre négligeable le coût de la réduction modulo $P(x)$. L'inversion se fait de manière simple par le procédé de Newton classique, et coûte donc une petite constante fois plus qu'une multiplication. L'application de l'automorphisme de Frobenius à un élément de $\mathbb{Z}_q$ est plus délicate. Nous reviendrons plus tard sur ce problème.

3.4.2 Principe général de l'algorithme de Satoh

Soit E une courbe elliptique définie sur $\mathbb{F}_q$. L'idée de Satoh [141] pour calculer la trace de E est de relever E dans les p-adiques ainsi que l'endomorphisme de Frobenius. L'existence d'un relèvement adéquat est garantie par un théorème de Lubin, Serre et Tate [114].

Théorème 10 (Lubin–Serre–Tate) *Soit E une courbe elliptique définie sur $\mathbb{F}_q$, ordinaire d'invariant modulaire j_E. Alors il existe une courbe elliptique $\mathcal{E}$ unique à isomorphisme près, définie sur $\mathbb{Z}_q$ telle que $\mathcal{E}$ se réduise en E modulo p et l'anneau des endomorphismes de E est isomorphe à celui de $\mathcal{E}$. De plus l'invariant modulaire $j_{\mathcal{E}}$ de $\mathcal{E}$ vérifie*

$$\Phi_p(j_{\mathcal{E}}, \sigma(j_{\mathcal{E}})) = 0.$$

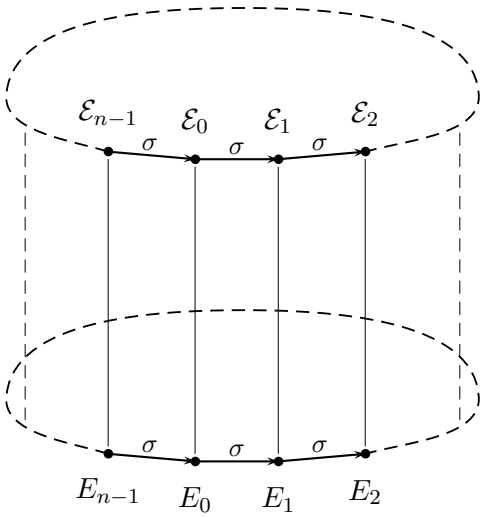


FIG. 3.2 – Cycle et cycle relevé de courbes conjuguées

Une courbe relevée $\mathcal{E}$ comme dans le théorème est appelée *relèvement canonique* de E. Plusieurs applications appelées *Frobenius* sont en jeu et interagissent. Nous noterons toujours σ de telles applications qui sont ou qui proviennent d'élévations à la puissance p, et π celles qui sont liées à l'élévation à la puissance q (donc la composée de n applications du premier type).

Notons E^σ la courbe conjuguée de E. Alors il existe une isogénie inséparable de degré p (toujours notée σ) qui va de E vers E^σ, il s'agit du morphisme qui élève chaque coordonnée à la puissance p. De plus par le théorème ci-dessus, il existe une isogénie (séparable, car en caractéristique 0) de degré p entre $\mathcal{E}$ et sa conjuguée $\mathcal{E}^\sigma$. On note aussi cette isogénie σ car c'est un relevé de l'isogénie de Frobenius entre E et E^σ (mais ce n'est pas l'application qui fait opérer l'automorphisme de corps $\mathbb{Q}_q$ sur chaque coordonnée, car celle-ci n'a aucune raison d'être un morphisme géométrique). On a donc le diagramme commutatif suivant :

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{\sigma} & \mathcal{E}^\sigma \\ \downarrow & & \downarrow \\ E & \xrightarrow{\sigma} & E^\sigma \end{array}$$

où les flèches verticales sont les réductions modulo p. On peut ensuite considérer les itérés de σ. Soit $E_0 = E$, et pour tout i dans $[1..n]$, soit $E_i = E^{\sigma^i}$. Comme σ est d'ordre n, on a $E_n = E_0 = E$. On définit de même $\mathcal{E}_i$ par $\mathcal{E}_i = \mathcal{E}^{\sigma^i}$. Le diagramme commutatif entre ce qui se passe sur $\mathbb{F}_q$ et ce qui se passe avec les relevés canoniques s'étend à tout le cycle de courbes, comme symbolisé dans la figure 3.2. Quand on fait un tour complet (en haut ou en bas), on n'obtient pas l'endomorphisme identité, mais l'endomorphisme de Frobenius π dont on cherche la trace. D'un point de vue algorithmique, la décomposition de π en produit d'isogénies plus simples va permettre de manipuler des objets suffisamment petits pour obtenir une complexité polynomiale lorsque p est petit.

La phase de relèvement de E. La première étape de l'algorithme de Satoh et de toutes ses variantes est de calculer une équation ou du moins l'invariant modulaire de $\mathcal{E}$, ceci à

une précision suffisante dont nous reparlerons plus tard. Pour cela on dispose d'équations polynomiales vérifiées par les invariants modulaires de toutes les courbes du cycle d'isogénies :

$$\forall i \in [0, n-1],\ \Phi_p(j_{\mathcal{E}_i}, j_{\mathcal{E}_{i+1}}) = 0, \tag{3.1}$$

ainsi que les relations galoisiennes :

$$\forall i \in [0, n-1],\ j_{\mathcal{E}_{i+1}} = \sigma(j_{\mathcal{E}_i}). \tag{3.2}$$

On a donc n équations algébriques et n équations tordues par σ en les n inconnues $j_{\mathcal{E}_i}$ dont on connaît la valeur modulo p. Dans l'algorithme original de Satoh, on se concentre uniquement sur les équations algébriques, de manière à éviter tout calcul de σ dans $\mathbb{Z}_q$. On dispose de suffisamment de relations pour espérer que l'algorithme de Newton multivarié fonctionne ; il faut cependant vérifier que la matrice jacobienne associée au système d'équations (3.1) est inversible. Nous renvoyons à l'article original [141] pour les détails de ces calculs ; mentionnons toutefois que le point clef de l'inversibilité est l'identité de Kronecker $\Phi_p(x, y) \equiv (x^p - y)(x - y^p)$ mod p. Au final, le relèvement tel que le décrit Satoh fonctionne dès que j_E n'appartient pas à $\mathbb{F}_{p^2}$.

D'autres méthodes permettent de calculer $j_{\mathcal{E}}$. Nous donnerons quelques détails ci-dessous de la méthode de Harley qui est la plus efficace asymptotiquement, et qui s'appuie sur deux équations en $j_{\mathcal{E}_0}$ et $j_{\mathcal{E}_1}$, l'une provenant du système (3.1), l'autre étant la relation galoisienne (3.2).

La phase de calcul de norme. Une fois la courbe relevée jusqu'à une précision suffisante, on peut en déduire la trace de l'endomorphisme de Frobenius π de $\mathcal{E}$ en le décomposant le long du cycle d'isogénies. Pour cela, à partir des invariants $j_{\mathcal{E}_0}$ et $j_{\mathcal{E}_1}$, on va choisir des équations pour $\mathcal{E}_0$ et $\mathcal{E}_1$, puis expliciter complètement l'isogénie $\sigma : \mathcal{E}_0 \to \mathcal{E}_1$. On se retrouve dans une situation similaire à celle que l'on a dans le calcul de l'isogénie dans l'algorithme SEA, toutefois, ici on dispose de l'information modulo p, et l'on désire juste la relever. Cela dit, comme modulo p l'isogénie σ n'est pas séparable, il est préférable de travailler avec son isogénie duale $\hat{\sigma}$. L'algorithme de Newton n'est alors plus gêné par les multiplicités, et on peut relever le facteur du polynôme de division Ψ_p de $\mathcal{E}_1$ qui correspond à $\hat{\sigma}$. Les formules de Vélu donnent ensuite toute l'information sur $\hat{\sigma}$, et en particulier son action sur la forme différentielle $\frac{\mathrm{d}x}{y}$. Ainsi on peut calculer l'élément γ dans $\mathbb{Q}_q$ tel que

$$\hat{\sigma}^* \left(\frac{\mathrm{d}x}{y} \right) = \gamma \, \frac{\mathrm{d}x}{y}.$$

On peut effectuer le même type de calcul pour l'isogénie $\sigma : \mathcal{E}_1 \to \mathcal{E}_2$; par conjugaison on obtiendrait alors γ^σ comme action sur la forme différentielle. Plus généralement, l'action sur les formes différentielles le long du cycle va être donnée par les conjugués successifs de γ. Pour finir, par composition, l'action de l'endomorphisme dual de π sur la forme différentielle principale de $\mathcal{E}$ est donnée par le produit de tous ces conjugués, c'est-à-dire par la norme de γ. La norme $\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}$ de γ est donc une valeur propre de $\hat{\pi}$, et comme le produit de ses valeurs propres vaut q, on en déduit que la trace cherchée est $\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(\gamma) + \frac{q}{\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(\gamma)}$.

Il y a plusieurs manières de calculer cette norme. Dans l'algorithme original de Satoh, comme on tient à éviter de calculer l'automorphisme de Frobenius de $\mathbb{Q}_q$, tous les conjugués de γ vont être calculés indépendamment en partant à chaque fois de deux invariants modulaires relevés successifs dans le cycle. Si par contre on dispose d'un algorithme efficace pour calculer la conjugaison, il vaut mieux l'utiliser pour accélérer ce calcul de norme.

Précision des calculs. La trace de la courbe est bornée en valeur absolue par $2\sqrt{q}$. Il faut donc calculer γ à une précision k telle que $p^k > 4\sqrt{q}$, ce qui donne $k = n/2 + O(1)$. Comme γ est obtenu essentiellement sans perte de précision à partir de $j_{\mathcal{E}_0}$ et $j_{\mathcal{E}_1}$, le relèvement canonique de ces j-invariants doit être effectué aussi à précision $n/2 + O(1)$.

Les améliorations successives de l'algorithme original. L'algorithme de Satoh tel que l'on vient brièvement de le décrire admet une complexité (à p fixé) en $\tilde{O}(n^3)$ et requiert un espace mémoire en $O(n^3)$. Un récapitulatif des améliorations successives ainsi qu'une implantation comparée de celles-ci se trouve dans [169]. Nous nous contentons ici de les survoler rapidement. Pour des raisons techniques, l'algorithme de Satoh ne fonctionnait que pour $p > 3$. L'extension au cas de $p = 2$ et $p = 3$ fut faite indépendamment par Fouquet-Gaudry-Harley [49] et Skjernaa [154]. Ensuite, Vercauteren [170] et Mestre [126] ont trouvé deux manières distinctes de réduire la complexité en espace à $O(n^2)$. Les approches sont en apparence assez différentes, mais l'algorithme de Mestre qui s'appuie sur la moyenne arithmético-géométrique (AGM) peut a posteriori être vu comme un changement de variables astucieux dans l'algorithme de Vercauteren (nous donnons quelques informations supplémentaires sur l'AGM au paragraphe suivant). Par rapport à l'algorithme de Vercauteren, la méthode de Mestre est plus rapide d'un facteur 3 environ, mais n'est valable qu'en caractéristique 2. Un peu plus tard, Satoh-Skjernaa-Taguchi [143] ont proposé l'utilisation de la représentation de $\mathbb{Z}_q$ à l'aide de racines de l'unité qui permet un calcul efficace de l'automorphisme de Frobenius. Ils obtinrent ainsi une complexité de $\tilde{O}(n^{2.5})$ après un précalcul de $\tilde{O}(n^3)$ ne dépendant que du corps fini considéré. Kim-Park-Cheon-Park-Kim-Hahn [98] ont ensuite montré que si le corps $\mathbb{F}_q$ admet une base normale gaussienne, alors le calcul de l'automorphisme de Frobenius peut être simplifié, ce qui gagne une constante appréciable dans le temps de calcul. En poussant plus loin dans cette direction, Lercier-Lubicz ont trouvé, toujours dans le cas où le corps $\mathbb{F}_q$ admet une base normale gaussienne, que l'on peut rajouter une étape récursive à l'intérieur du relèvement de Newton (qui est déjà récursif), de manière à obtenir un algorithme de complexité $\tilde{O}(n^2)$. Parallèlement, Harley a obtenu un autre algorithme ayant cette même complexité mais tout à fait général : aucune hypothèse sur le corps de base n'est nécessaire (si ce n'est bien sûr que la caractéristique est petite). Ce résultat a en quelque sorte clos le sujet de la recherche de la meilleure complexité possible, puisque le temps de calcul est quasi-linéaire en la taille de l'objet intermédiaire que l'on calcule.

En ce qui concerne les améliorations pratiques (qui concernent la constante dans le $O()$), nous avons montré [65] que la méthode AGM pouvait être intégrée aux algorithmes asymptotiquement rapides afin de bénéficier des avantages des formules extrêmement compactes. Dans cette optique, Kohel [99] et Madsen [115] ont proposé des adaptations de la méthode AGM aux petites caractéristiques différentes de 2.

Quelques mots sur l'AGM. Dans l'algorithme AGM de Mestre, on utilise le fait suivant. Soit E une courbe elliptique sur un corps de caractéristique 0 donnée par une équation $y^2 = x(x - a^2)(x - b^2)$; alors la courbe E' d'équation $y^2 = x(x - a'^2)(x - b'^2)$ avec $a' = \frac{a+b}{2}$ et $b' = \sqrt{ab}$ est 2-isogène à E. L'isogénie en question est complètement explicite et son noyau est donné par le point $(0, 0)$ de E. Partant d'un courbe elliptique en caractéristique 2, le relèvement canonique p-adique sera calculé sous la forme $y^2 = x(x - a^2)(x - b^2)$. Des conditions de congruence sur a et b permettent de s'assurer que le point $(0, 0)$ est le noyau du morphisme de Frobenius, si bien que l'isogénie donnée par les formules d'AGM est le

morphisme de Frobenius, composé avec un isomorphisme correspondant au changement de modèle pour la courbe conjuguée. Ainsi, si le relèvement canonique recherché a une équation mise sous la forme $y^2 = x(x-a^2)(x-b^2)$ avec $a \equiv b \equiv 1 \mod 4$ et $a/b \equiv 1 \mod 8$, on peut montrer qu'en itérant les formules d'AGM on obtient des équations pour tout le cycle des conjugués de relèvements canoniques. Lors de cette itération des formules d'AGM, un «miracle» se produit : si on était parti d'une approximation du relèvement canonique, chaque étape fournit une approximation d'un relèvement canonique à une précision qui augmente d'une unité (c'est ce point crucial qui a été remarqué par Vercauteren, dans le contexte d'un calcul utilisant le polynôme Φ_2). Partant d'une première approximation du relèvement canonique à très faible précision, au bout de $n/2 + O(1)$ étapes on a obtenu une équation de courbe relevée avec suffisamment de précision pour en déduire la trace.

Une manière de raccrocher l'algorithme AGM au cadre que nous avons utilisé ici est de considérer une version univariée de l'AGM : on pose $\lambda = \frac{b}{a}$ qui n'est autre que la racine carrée d'un invariant de Legendre. Alors $\lambda' = \frac{b'}{a'}$ vérifie l'équation $\lambda'^2(1+\lambda)^2 - 4\lambda = 0$. Cette relation est l'équation modulaire associée à l'AGM et peut-être utilisée en lieu et place de Φ_2 dans l'algorithme de Harley que nous allons maintenant décrire.

3.4.3 L'algorithme de Harley

Nous donnons quelques détails sur l'algorithme de Harley qui est le plus général parmi les algorithmes asymptotiquement les meilleurs. L'équation à résoudre est $\Phi_p(j_\mathcal{E}, j_\mathcal{E}^\sigma) = 0$, sachant que l'on connaît $j_\mathcal{E}$ modulo p. Notons que dans tout ce qui suit, on ne manipule que des entiers de $\mathbb{Z}_q$: lorsque l'on divise un élément par un puissance de p, c'est qu'il a une valuation suffisante pour rester entier.

Faisons pour l'instant l'hypothèse que l'automorphisme σ de $\mathbb{Q}_q$ est calculable efficacement. Un procédé à la Newton va être utilisé ; partant d'une valeur X dans $\mathbb{Z}_q$ qui est une approximation de $j_\mathcal{E}$ à précision p^k, on cherche à améliorer cette approximation. Notons e l'erreur, c'est-à-dire l'élément de $\mathbb{Z}_q$ tel que $j_\mathcal{E} = X + p^k e$. Alors on a aussi $j_\mathcal{E}^\sigma = X^\sigma + p^k e^\sigma$, et si l'on substitue ces expressions dans l'équation modulaire qui est censée s'annuler, on obtient par développement de Taylor :

$$\begin{aligned} 0 &= \Phi_p(X + p^k e, X^\sigma + p^k e^\sigma) \\ &= \Phi_p(X, X^\sigma) + p^k \left(e \, \tfrac{\partial \Phi_p}{\partial x}(X, X^\sigma) + e^\sigma \, \tfrac{\partial \Phi_p}{\partial y}(X, X^\sigma) \right) + p^{2k}(\ldots), \end{aligned}$$

où l'expression en facteur de p^{2k} est un entier de $\mathbb{Z}_q$. Cette égalité implique que la valuation de $\Phi_p(X, X^\sigma)$ est au moins k, et donc on peut l'écrire $p^k v$, avec v dans $\mathbb{Z}_q$. En divisant par p^k on obtient alors l'équation suivante :

$$v + e \, \frac{\partial \Phi_p}{\partial x}(X, X^\sigma) + e^\sigma \, \frac{\partial \Phi_p}{\partial y}(X, X^\sigma) \equiv 0 \mod p^k. \tag{3.3}$$

On a supposé X connu et σ calculable efficacement, de sorte que la seule inconnue de l'équation (3.3) est e. L'égalité de Kronecker implique que si j_E n'est pas dans $\mathbb{F}_{p^2}$, alors $\frac{\partial \Phi_p}{\partial x}(X, X^\sigma) \equiv 0 \mod p$ et $\frac{\partial \Phi_p}{\partial y}(X, X^\sigma) \not\equiv 0 \mod p$, de sorte qu'on est ramené à la question de résoudre une équation de la forme

$$e^\sigma + Ae + B = 0$$

dans $\mathbb{Z}_q$, avec A congru à 0 modulo p. Une telle équation est appelée une équation d'Artin-Schreier, par analogie avec les équations du même type sur les corps finis. La condition de congruence sur A assure l'existence et l'unicité de la solution e. Si l'on sait calculer cette solution e à la précision k, en remontant les calculs, on vérifie aisément que l'élément $X + p^k e$ est une approximation de $j_{\mathcal{E}}$ à la précision $2k$. Ainsi partant d'une solution approchée X, on peut améliorer l'approximation en doublant le nombre de chiffres significatifs au prix de quelques opérations correspondant aux évaluations de Φ_p et de ses dérivées en X et X^σ et une résolution d'équation d'Artin-Schreier.

Résolution d'équations d'Artin-Schreier dans $\mathbb{Z}_q$. On procède de nouveau par un algorithme de type Newton. Soit à résoudre une équation de la forme $e^\sigma + Ae + B = 0$, avec $A \equiv 0 \mod p$. Modulo p cette équation se réduit à calculer la racine p-ème de la réduction de B dans $\mathbb{F}_q$, ce qui est un calcul que l'on sait effectuer rapidement. Supposons dorénavant que l'on dispose d'une approximation X de la solution e à la précision k : on peut écrire $e = X + p^k f$, où f est un élément de $\mathbb{Z}_q$ qui est l'erreur que l'on veut déterminer. En substituant cette expression dans l'équation que l'on doit vérifier on obtient :

$$0 = e^\sigma + Ae + B = (X^\sigma + AX + B) + p^k(f^\sigma + Af),$$

par linéarité de σ. Par hypothèse, $X^\sigma + AX + B$ est congru à 0 modulo p^k, on peut donc diviser l'équation précédente par p^k et l'on obtient de nouveau une équation de la forme Artin-Schreier où f est l'inconnue. Ceci mène naturellement à un algorithme récursif, que nous explicitons maintenant afin de montrer que l'on traite effectivement des instances de tailles décroissantes. Nous supposons que A est congru à 0 modulo p et que k est un puissance de 2 pour simplifier.

Algorithme `ArtinSchreier`(A, B, k).

{ k puissance de 2 ; $A, B \in Z_q$; $A \equiv 0 \mod p$ }

1. Si $k = 1$, retourner $\sqrt[p]{B} \bmod p$.
2. $X \leftarrow$ `ArtinSchreier`$(A, B, \frac{k}{2})$. (X est connu modulo $p^{k/2}$.)
3. Relever arbitrairement X à la précision p^k.
4. $B' \leftarrow (X^\sigma + AX + B)/p^{k/2}$. ($B'$ est connu modulo $p^{k/2}$.)
5. $E \leftarrow$ `ArtinSchreier`$(A, B', \frac{k}{2})$. (E est connu modulo $p^{k/2}$.)
6. Retourner $X + p^{k/2}E$.

Complexité. Si l'on note $C(k)$ le coût de l'algorithme `ArtinSchreier` appliqué à la précision k, ce coût vérifie $C(k) = 2C(\frac{k}{2}) + \lambda \mathsf{M}(n\frac{k}{2})$, où λ est une constante, sous réserve que le coût du calcul de σ est au pire proportionnel à celui d'une multiplication. Ceci se résout en $C(k) = O(\log(k)\mathsf{M}(nk))$. Ainsi, la résolution d'une équation de type Artin-Schreier se fait en temps quasi-linéaire. Si l'on reporte cette complexité dans l'algorithme de Harley, on obtient finalement une complexité quasi-linéaire pour le calcul du j-invariant du relèvement canonique de la courbe E.

Calcul efficace du Frobenius. Il nous reste à préciser comment le calcul de σ peut-être rendu efficace. Comme évoqué ci-dessus, cela repose sur une représentation de $\mathbb{Z}_q$ à l'aide de racines de l'unité : on utilise un polynôme $P(x)$ sur $\mathbb{Z}_p[x]$, irréductible de degré n, qui soit un

facteur de $x^{q-1}-1$. Soit z un élément de $\mathbb{Z}_q$ représenté par un polynôme en x modulo $P(x)$, on a $z = z_0+z_1x+\cdots+z_{n-1}x^{n-1}$ où les z_i sont des éléments de $\mathbb{Z}_p$. Comme σ est un automorphisme et que $\sigma(x) = x^p$ par le choix de P, on a donc $\sigma(z) = z_0 + z_1x^p + \cdots + z_{n-1}x^{p(n-1)}$. Il ne reste plus qu'à réduire ce polynôme modulo $P(x)$ pour obtenir une représentation canonique de $\sigma(z)$, ce qui coûte $O(\mathsf{M}(nk))$, lorsque p est fixé et que l'on travaille à précision k.

Le calcul du polynôme $P(x)$ adéquat se fait (de nouveau) par un relèvement de Newton, en s'appuyant sur l'équation $P(x^p) = \prod_{i=0}^{p-1} P(x\zeta_p^i)$, où ζ_p est une racine primitive p-ème de l'unité. Cette équation est algébrique, si bien que le relèvement ne pose pas de problème, et une analyse de complexité donne un temps de calcul quasi-linéaire en la taille de l'objet calculé.

Ceci clôt notre description de la phase de calcul du relèvement canonique d'une courbe elliptique par l'algorithme de Harley.

Calcul asymptotiquement rapide de la norme. Une fois calculé le relèvement canonique de E, il reste une norme à évaluer, qui fournit directement la trace de E. Pour cette phase, Harley propose d'utiliser un calcul de résultant : si z est un élément de $\mathbb{Z}_q$ représenté par un polynôme modulo P, la norme de z sur $\mathbb{Z}_p$ est précisément le produit des évaluations de ce polynôme en chaque racine de P, ce qui correspond à la définition du résultant à un signe près. Il existe des algorithmes asymptotiquement rapides pour calculer le résultant de deux polynômes dont la complexité est quasi-linéaire en la taille des entrées. En pratique, en particulier en caractéristique 2, on utilise d'autres méthodes (surtout celle de [143]) ayant une complexité moins bonne mais plus efficace pour les tailles atteignables aujourd'hui.

Complexité totale. Afin d'avoir assez d'information pour conclure, par les bornes de Hasse, il suffit de connaître la trace modulo p^k avec $k > \frac{n}{2} + \log_p(4)$. Il faut donc relever canoniquement le j invariant de E jusqu'à cette précision, ce qui coûte $\tilde{O}(n^2)$, puis effectuer le calcul de norme, ce qui se fait en la même complexité.

3.4.4 Les algorithmes de Mestre pour le genre supérieur

Limites théoriques. Le théorème d'existence d'un relèvement canonique s'étend aux variétés abéliennes ordinaires. On peut aussi relever une polarisation éventuellement principale en même temps que la variété abélienne qui nous intéresse. Toutefois il n'est en général pas possible de relever canoniquement une jacobienne de courbe en une jacobienne de courbe : le relèvement est une variété abélienne principalement polarisée, mais cette polarisation ne provient pas d'une courbe. Nous renvoyons à [135] pour ces questions.

Néanmoins, en genre 2 la situation est encore favorable, car toute variété abélienne principalement polarisée de dimension 2 est une jacobienne de courbe de genre 2. On peut donc espérer mimer le cas du genre 1 et relever canoniquement la courbe. Pour le genre supérieur, la situation est plus chaotique, et il convient d'oublier la courbe et de se concentrer sur sa jacobienne.

Calcul du relèvement canonique. Un outil crucial pour relever canoniquement la jacobienne d'une courbe est une représentation explicite d'une $(p, p, \ldots, p)$-isogénie qui se réduit en l'endomorphisme de Frobenius modulo p. Ceci peut-être un idéal modulaire qui relie les invariants de deux variétés abéliennes isogènes ou bien des formules à la Vélu qui donne

une variété abélienne isogène à partir d'un sous-groupe explicite (les formules d'AGM sur les courbes elliptiques sont de ce type).

Mestre a proposé deux solutions différentes pour le cas de la caractéristique 2. La première ne fonctionne qu'en genre 2 et s'appuie sur les formules de Richelot [138, 9]. Il s'agit de formules qui donnent explicitement une courbe de genre 2 dont la jacobienne est $(2,2)$-isogène à la jacobienne d'une courbe de genre 2 de départ. La deuxième méthode est plus générale puisqu'elle s'applique aux courbes hyperelliptiques de genre quelconque. Comme dit précédemment, dans une telle méthode il n'est pas possible de relever la courbe ; Mestre [125] propose donc d'utiliser les formules de duplication de fonctions Thêta, sous la forme appelée *moyenne de Borchardt*. Ainsi on ne manipule que des constantes liées à des variétés abéliennes qui ne sont pas des jacobiennes en général.

Une fois que l'on dispose d'une représentation effective (Richelot ou Borchardt) de l'isogénie qui nous intéresse, l'extension de la phase de relèvement par l'algorithme de Harley est relativement aisée (mais assez technique dans sa mise en pratique). Finalement, à genre g fixé, cette phase de relèvement a une complexité qui est de nouveau quasi-linéaire en la taille de la sortie. Toutefois la dépendance en g est très mauvaise, puisque la moyenne de Borchardt manipule 2^g fonctions Thêta.

Mentionnons aussi que des travaux récents [17] permettent de s'affranchir de la contrainte $p = 2$.

Déduction de la fonction Zêta. Dans le cas du genre 2, si l'on utilise les formules de Richelot, on est dans un situation très similaire à celle du genre 1. Une fois une courbe et sa conjuguée relevées canoniquement à une précision suffisante, à l'aide des formules de Richelot on peut expliciter complètement l'isogénie relevée de Frobenius qui relie ces deux courbes. On en déduit son action sur la base canonique des formes différentielles $(\frac{\mathrm{d}x}{y}, \frac{x\mathrm{d}x}{y})$, sous la forme d'une matrice M. La norme matricielle de cette matrice : $\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(M) = M^{\sigma^{n-1}} M^{\sigma^{n-2}} \cdots M^{\sigma} M$ fournit alors la matrice de l'action de l'automorphisme de Frobenius π de la jacobienne de la courbe sur les formes différentielles. Le polynôme caractéristique de π que l'on cherche à calculer se déduit dont simplement de celui de cette norme matricielle.

Dans le cas plus général de l'utilisation des formules de Borchardt, l'absence d'information sur la courbe sous-jacente fait que l'on n'a pas le moyen de trouver directement toute l'information. Toutefois, grâce aux formules de Thomae, la norme d'un scalaire (et non plus d'une matrice) de $\mathbb{Z}_q$ donne le produit des valeurs propres de l'automorphisme de Frobenius qui sont inversibles. Si la précision est assez grande, cela suffit pour reconstruire complètement la fonction Zêta cherchée par l'algorithme LLL. Cette approche a été analysée en détail et implantée par Lercier et Lubicz [111].

Courbes non hyperelliptiques. Dans les algorithmes de Mestre, une première étape dont nous n'avons pas parlé est de calculer une information modulo p qui sera relevable. Il s'agit en l'occurrence de Thêta-constantes. Le calcul de telles Thêta-constantes à partir d'une équation de courbe est bien maîtrisé dans le cas des courbes hyperelliptiques. Pour les courbes non-hyperelliptiques, ce n'est pas toujours le cas. Ritzenthaler [139] est parvenu à calculer les invariants adéquats dans les cas des courbes de genre 3 non-hyperelliptiques en caractéristique 2, étendant par là-même l'algorithme de Mestre à ces courbes.

3.5 L'algorithme de Kedlaya

Tout comme l'algorithme de Satoh, l'algorithme de Kedlaya [97] repose sur des calculs dans un relèvement p-adique de la courbe. Toutefois, Kedlaya fait peser la difficulté non plus sur la manière d'effectuer le relèvement, puisque l'on choisit un relèvement essentiellement quelconque, mais sur les calculs effectués avec ce relèvement. En effet, des efforts supplémentaires seront nécessaires avant de passer à la phase de norme (qui ressemble à celle de l'algorithme de Satoh).

3.5.1 L'espace «dague» et la formule des traces associée

Soit $\mathcal{C}$ une courbe affine lisse sur $\mathbb{F}_q$, donnée par une équation $\overline{f}(x,y) = 0$. Soit $f(x,y)$ un relèvement quelconque du polynôme $\overline{f}$ en un polynôme sur $\mathbb{Q}_q$, tel que la courbe définie par $f(x,y) = 0$ soit lisse. L'anneau de coordonnées $A = \mathbb{F}_q[x,y]/(\overline{f}(x,y))$ de $\mathcal{C}$ se relève p-adiquement en $\mathcal{A} = \mathbb{Q}_q[x,y]/(f(x,y))$. Toutefois, sauf cas particulier, le morphisme de Frobenius de A vers A^σ ne se relève pas en un morphisme de $\mathcal{A}$ vers $\mathcal{A}^\sigma$. Prendre le complété $\mathcal{A}^\infty$ de $\mathcal{A}$ défini par les séries convergentes de $\mathbb{Q}_q[[x,y]]/(f(x,y))$ suffit pour définir un relèvement du morphisme de Frobenius, mais cet espace est trop grand : il contient trop d'éléments sans primitives, ce qui fait que l'espace H^1 associé est de dimension infinie. La solution proposée par Monsky est de travailler dans l'espace des séries qui convergent rapidement, c'est-à-dire que la valuation des coefficients croît au moins linéairement en le degré : il s'agit de la complétion faible (ou «dague») qui prend, dans le cas qui nous intéresse, la forme

$$\mathcal{A}^\dagger = \mathbb{Q}_q\langle\langle x,y\rangle\rangle/(f(x,y)),$$

où

$$\mathbb{Q}_q\langle\langle x,y\rangle\rangle = \left\{\sum r_{i,j}x^iy^j \in \mathbb{Q}_q[[x,y]] \ ; \ \exists\delta\in\mathbb{R}, \exists\varepsilon>0, \forall i,j, \ \mathrm{ord}_p(r_{i,j}) \geqslant \varepsilon(i+j)+\delta\right\}.$$

Cet espace est bien adapté à notre problème. Il est possible de relever le Frobenius dans cet espace. Soit $H^i(\mathcal{A}^\dagger)$ les espaces de cohomologie définis par le complexe de de Rham associé à $\mathcal{A}^\dagger$. Monsky et Washnitzer ont démontré une formule des traces de Lefschetz pour cette cohomologie :

$$\#\mathcal{C}/\mathbb{F}_{q^k} = \mathrm{Tr}\left((q\pi_*^{-1})^k | H^0(\mathcal{A}^\dagger)\right) - \mathrm{Tr}\left((q\pi_*^{-1})^k | H^1(\mathcal{A}^\dagger)\right),$$

où π_* est le morphisme induit par le Frobenius sur les espaces de cohomologie.

La stratégie employée dans l'algorithme de Kedlaya est de calculer explicitement l'action du Frobenius sur une base de $H^1(\mathcal{A}^\dagger)$, sachant que $H^0(\mathcal{A}^\dagger)$ est de dimension 1 et que la contribution correspondante dans la formule ne pose pas de problèmes.

3.5.2 Le cas superelliptique

Nous présentons ici une version un peu plus générale que l'algorithme original de Kedlaya, qui inclut les courbes de la forme $y^r = \overline{f}(x)$, où r est premier à la caractéristique. On suppose de plus que le degré d de $\overline{f}$ est premier avec r et que $\overline{f}$ est sans facteur carré. Sous ces conditions, le genre d'une telle courbe est $g = (d-1)(r-1)/2$.

La tâche principale est de trouver une base des formes différentielles holomorphes dans $H^1(\mathcal{A}^\dagger)$. Sur le corps fini, et pour la courbe complète, une base algébrique est facilement

calculable avec des éléments du type $\frac{x^i \mathrm{d}x}{y^j}$. Le problème est de trouver une autre base qui corresponde à la courbe affine et qui ne fasse pas intervenir de division par y, de manière à pouvoir relever dans l'anneau $\mathcal{A}^\dagger$. Ceci est possible, mais l'approche de Kedlaya pour ce type de courbes est de retirer des points à la courbe de manière à autoriser les divisions par y. Cela perturbe le nombre de points, et donc aussi la fonction Zêta ; cela perturbe aussi la dimension du H^1 qui passe de $2g$ à $2g+d$. Toutefois on contrôle sans problèmes ces perturbations, et en contrepartie les calculs sont simplifiés car on peut travailler avec une base très simple.

Soit donc $\mathcal{C}^\bullet$ une courbe relevée d'une courbe $\mathcal{C}$ d'équation $y^r = \overline{f}(x)$, obtenue en relevant arbitrairement le polynôme $\overline{f}$ en un polynôme f à coefficients dans $\mathbb{Z}_q$, de même degré, et en retirant le diviseur de y, c'est-à-dire d points. L'anneau des coordonnées associé à $\mathcal{C}^\bullet$ est $\mathcal{A}^\dagger = \mathbb{Q}_q\langle\langle x, y, y^{-1}\rangle\rangle/(y^r - f(x))$. L'espace $H^1(\mathcal{A}^\dagger)$ se décompose en la somme directe de deux sous-espaces : H^1_+ qui est stable sous l'action de $y \mapsto \zeta_r y$, et H^1_- qui regroupe les espaces propres pour les autres valeurs propres de $y \mapsto \zeta_r y$. On a alors le résultat suivant :

Théorème 11 *Une base de $H^1_+(\mathcal{A}^\dagger)$ est donnée par $\left\{\frac{x^i \mathrm{d}x}{y^r};\ i \in [0, d-1]\right\}$.*
Une base de $H^1_-(\mathcal{A}^\dagger)$ est donnée par $\left\{\frac{x^i \mathrm{d}x}{y^j};\ i \in [0, d-2], j \in [1, r-1]\right\}$.

La preuve de ce théorème est le cœur de l'algorithme de Kedlaya : partant d'une forme différentielle écrite sous forme générale, on a des méthodes de réécriture qui permettent de se ramener à une combinaison d'éléments de la base annoncée. Ensuite un lemme technique de convergence assure que la réécriture se fait avec une perte contrôlée de précision, si bien qu'elle a du sens dans l'espace des séries surconvergentes. L'algorithme procède donc par les étapes suivantes :

1. Calculer un relèvement du morphisme de Frobenius de $\mathcal{A}^\dagger$ vers $\sigma(\mathcal{A}^\dagger)$. On fixe arbitrairement $\sigma(x) = x^p$, et l'on calcule $\sigma(y)$ correspondant, comme une série surconvergente.
2. Calculer $\omega_{i,j} = \left(\frac{x^i \mathrm{d}x}{y^j}\right)^\sigma$ comme une série (tronquée), pour $i \in [0, d-2]$ et $j \in [1, r-1]$.
3. Récrire $\omega_{i,j}$ sur la base de $H^1_-(\sigma(\mathcal{A}^\dagger))$; stocker le résultat sous la forme d'une matrice M à $2g$ lignes et colonnes.
4. Retourner le polynôme caractéristique de $M^{\sigma^{n-1}} \cdots M^\sigma \cdot M$.

La matrice M représente l'action du morphisme de Frobenius σ de l'espace $H^1_-(\mathcal{A}^\dagger)$ vers son conjugué. La norme matricielle représente la composée de tous les conjugués, c'est-à-dire l'action de l'automorphisme π sur $H^1_-(\mathcal{A}^\dagger)$. La formule des traces permet alors de relier le polynôme caractéristique de cette norme matricielle au polynôme caractéristique du Frobenius de la jacobienne de la courbe de départ. Il faut tenir compte des points que l'on a enlevés, des contributions correspondants à H^0 et à H^1_+, mais au final tout se compense et les polynômes caractéristiques sont identiques.

Calcul de $1/y^\sigma$. Le relèvement du morphisme de Frobenius doit rester compatible avec l'équation de la courbe. Ainsi on doit avoir $(y^\sigma)^r = (f(x))^\sigma$. Comme on choisit de fixer $x^\sigma = x^p$, on obtient

$$\begin{aligned}\frac{1}{y^\sigma} &= (f(x)^\sigma)^{-1/r} = (f(x)^\sigma - f(x)^p + f(x)^p)^{-1/r} \\ &= \frac{1}{y^p}\left(1 + \frac{f(x)^\sigma - f(x)^p}{f(x)^p}\right)^{-1/r} = \frac{1}{y^p}\left(1 + (f(x)^\sigma - f(x)^p)\frac{1}{y^{rp}}\right)^{-1/r}.\end{aligned}$$

Cette dernière expression de la forme $(1+X)^\alpha$ se développe aisément en une série en x et $\frac{1}{y}$ dont les termes tendent p-adiquement vers 0 à une vitesse au moins linéaire en le degré, car $(f(x)^\sigma - f(x)^p)$ est divisible par p. D'un point de vue algorithmique, on calcule ce développement par une itération de Newton : la suite initialisée avec $u_0 = 1$ et définie par $u_{n+1} = \frac{1}{r}((r+1)u_n - au_n^{r+1})$ converge rapidement vers $a^{-1/r}$.

Il est à noter que dans ces calculs avec des séries en x et $1/y$, on prend soin de maintenir le degré en x borné par d en utilisant l'équation de la courbe.

Calcul de $\omega_{i,j}$ sur la base de H^1_-. La formule pour $\omega_{i,j}$ donne directement $\omega_{i,j} = \frac{1}{(y^\sigma)^j} p x^{ip+p-1} \mathrm{d}x$. En substituant la série calculée à l'étape précédente pour $1/y^\sigma$, on obtient une série de la forme

$$\omega_{i,j} = \left(\sum_{k \geqslant 0} Q_k(x) \left(\frac{1}{y^r} \right)^k \right) \frac{\mathrm{d}x}{y^{jp \bmod r}},$$

où Q_k est un polynôme de degré strictement inférieur à d sauf Q_0 dont le degré peut être plus grand. Pour simplifier, on pose $\tau = 1/y^r$ et $\ell = jp \bmod r$. Soit à réduire le terme $Q_k(x)\tau^k \frac{\mathrm{d}x}{y^\ell}$. Comme f est sans facteur carré, on peut écrire une identité de Bézout $Q_k = Uf + Vf'$. En considérant alors la différentielle exacte $\mathrm{d}\left(\frac{V(x)}{y^{r(k-1)+\ell}}\right)$, on obtient $Q_k(x)\tau^k \frac{\mathrm{d}x}{y^\ell} \equiv \left(U(x) + \frac{r}{r(k-1)+\ell} V'(x)\right) \tau^{k-1} \frac{\mathrm{d}x}{y^\ell}$. En itérant ce procédé, on rassemble toutes les contributions de la série sur un seul terme de la forme $Q(x)\frac{\mathrm{d}x}{y^\ell}$, avec Q un polynôme de degré δ. Si $\delta \geqslant d-1$, alors on utilise la différentielle exacte $\mathrm{d}(x^{\delta-d+1}y^{r-\ell})$ pour faire baisser d'au moins un le degré de Q, et finalement on obtient une écriture de $\omega_{i,j}$ sur la base souhaitée.

Critère de convergence. Avant les étapes de réécriture, l'expression de $\omega_{i,j}$ est bien sous la forme d'une série surconvergente de $\mathcal{A}^\dagger$. Un lemme technique de Kedlaya (prouvé à l'aide d'une étude locale au voisinage des points enlevés) implique que le processus de réduction ne fait perdre au plus qu'une puissance de p *logarithmique* en le degré en τ, si bien que cela ne perturbe pas la convergence dans $\mathcal{A}^\dagger$. Par ailleurs d'un point de vue pratique, cela permet de décider où tronquer les séries de manière à obtenir *in fine* une précision suffisante pour pouvoir conclure grâce aux bornes que l'on a sur les coefficients du polynôme caractéristique du Frobenius.

Complexité. La taille des coefficients recherchés requiert une précision p-adique en $O(ng)$. Les séries que l'on manipule doivent donc être tronquées à une précision τ-adique en $O(png)$. Ces séries ont pour coefficients des polynômes en x de degré au plus d à coefficients dans $\mathbb{Z}_q$ tronqués à précision p-adique en $O(ng)$. Ainsi les objets manipulés ont une taille qui est en $\tilde{O}(pn^3g^2d)$. Une analyse de complexité un peu plus fine montre qu'à r fixé, le temps de calcul est en $\tilde{O}(p^2n^3g^4)$. En changeant légèrement la représentation des séries [68], on peut faire baisser la complexité à $\tilde{O}(pn^3g^4)$. Plus récemment, en s'inspirant de [11], Harvey [84] est parvenu à ramener la dépendance en p à $\sqrt{p}$.

3.5.3 Les extensions.

L'algorithme de Kedlaya a été étendu à d'autres classes de courbes par Vercauteren et Denef. La première extension concerne les courbes hyperelliptiques en caractéristique 2 [29].

Dans cet algorithme, de manière similaire à l'algorithme original de Kedlaya, on retire des points de la courbe pour définir une base de différentielle qui soit agréable à manipuler. Toutefois, les points retirés sont un peu plus délicats à décrire. Les étapes sont ensuite les mêmes que pour l'algorithme original, mais les complications techniques, notamment pour évaluer les pertes de précision, sont plus nombreuses. Au final la complexité obtenue est en $\tilde{O}(g^4 n^3)$ pour une courbe hyperelliptique générique de genre g sur $\mathbb{F}_{2^n}$.

Une deuxième extension de l'algorithme de Kedlaya [28] concerne les courbes C_{ab}, c'est-à-dire les courbes lisses qui admettent une équation de la forme $y^a + \sum_{i=1}^{a-1} f_i(x) y^i + f_0(x) = 0$, où f_0 est de degré b et tel que $a \deg f_i + bi < ab$ pour $i = 1, \ldots, a-1$. On peut montrer qu'une telle courbe a une unique place à l'infini et que son genre est $g = (a-1)(b-1)/2$. Cette classe de courbes contient les courbes superelliptiques déjà étudiées. Dans cet algorithme, les courbes sont trop générales pour que l'on puisse espérer trouver quelques points à enlever qui permettent de simplifier les formules ; on garde donc la courbe relevée telle quelle. Ceci complique la première phase où l'on relève le Frobenius. En effet, il n'est plus possible de fixer $\sigma(x) = x^p$ et de calculer le $\sigma(y)$ correspondant : a priori un tel relèvement du Frobenius n'existe pas. Il faut donc procéder à un relèvement simultané de $\sigma(x)$ et de $\sigma(y)$. Celui-ci, bien que s'appuyant encore une fois sur une itération de Newton est non-trivial, notamment en ce qui concerne l'estimation de la vitesse de convergence. Une fois le Frobenius relevé, il reste encore à déterminer une base explicite des formes différentielles de $H^1(\mathcal{A}^\dagger)$. Ceci ce fait en déterminant tout d'abord une base algébrique, puis en vérifiant que les formules de réduction qui permettent d'écrire toute forme différentielle sur cette base préservent bien la convergence au sens †. Nous ne rentrerons pas dans les détails qui deviennent rapidement techniques. La complexité de l'algorithme obtenu est en $\tilde{O}(g^5 n^3)$.

Ce dernier algorithme a encore été étendu dans [19] à une classe plus générale de courbes : les courbes non dégénérées, c'est-à-dire les courbes données par une équation dont le polygone de Newton vérifie des hypothèses de généricité.

3.6 L'utilisation de la théorie de la déformation

L'idée est de considérer une famille de courbes à un paramètre. Dans le cas elliptique en caractéristique impaire, on peut considérer par exemple la famille de Legendre $y^2 = x(x-1)(x-\lambda)$; en caractéristique 2, on peut prendre la famille $y^2 + xy = x^3 + \lambda$. Plus généralement, Hubrechts [92] a montré que pour toute caractéristique, on pouvait trouver une famille qui couvre l'ensemble des courbes ordinaires à tordue quadratique près. Dans le cas du genre supérieur, on choisira une famille arbitraire, c'est-à-dire une équation générale de courbe du type choisi, dans laquelle tous les coefficients sont fixés sauf un paramètre λ qui est laissé libre. Pour les applications cryptographiques, on pourra de plus choisir une famille de courbes pour lesquelles la loi de groupe est plus efficace que les courbes génériques.

Sous quelques hypothèses de généricité, il est possible de relever la famille de courbes vers les p-adiques, puis de définir un espace H^1 ainsi qu'un opérateur de Frobenius associé. Le calcul de cet opérateur, c'est-à-dire de sa matrice dans une base donnée (que l'on prend de manière à simplifier les calculs) constitue la première phase de l'algorithme qui peut être vue comme un précalcul qui ne dépend que de la famille considérée.

Comme dans les algorithmes de Satoh et de Kedlaya, on va calculer la matrice associée à l'opérateur de Frobenius «puissance p», et non pas à l'opérateur complet : ce dernier s'obtient in fine par un calcul de norme matricielle.

Précisons un peu l'allure de la matrice $M = (m_{ij})$ que l'on cherche : elle doit vérifier $\omega_i^\sigma = \sum m_{ij}\omega_j$, où les ω_i forment une base de l'espace H^1. Les coefficients m_{ij} seront des éléments d'un espace †, c'est-à-dire des séries surconvergentes en le paramètre λ. En pratique, c'est séries seront évidemment tronquées à un certain dégré et à une certaine précision p-adique.

Une fois la matrice M calculée, il «suffit» de l'évaluer en une valeur de λ particulière pour obtenir la matrice de l'opérateur de Frobenius de l'espace H^1 de la courbe de paramètre λ. On se retrouve alors dans la situation finale de l'algorithme de Kedlaya et il ne reste qu'à effectuer le calcul de norme matricielle.

Le calcul de la matrice M s'effectue grâce à la théorie de la déformation : cette dernière donne une équation différentielle p-adique vérifiée par l'opérateur de Frobenius. Cette équation est en général d'une forme suffisamment agréable pour être résolue par une méthode itérative qui va aboutir grâce à des arguments de surconvergence. Il est toutefois nécessaire d'avoir un point de départ pour initialiser cette résolution. Ce point de départ sera une courbe particulière dans la famille, qui sera choisie de sorte que sa matrice de Frobenius ne soit pas trop délicate à calculer. Si aucune courbe ne paraît plus attractive qu'un autre, il est de toute façon possible d'appliquer l'algorithme de Kedlaya sur une courbe aléatoire de la famille.

L'algorithme procède donc ainsi :

1. Choisir une famille de courbes à un paramètre λ ; trouver une base du H^1 associée ;
2. Pour une valeur particulière λ_0, calculer la matrice M_0 de l'opérateur de Frobenius sur le H^1 du relevé de la courbe de paramètre λ_0 ;
3. Utiliser l'équation différentielle de la théorie de la déformation pour calculer la matrice M, avec comme valeur initiale M_0 en λ_0 ;
4. Évaluer la matrice M en le paramètre λ correspondant à la courbe dont on cherche la fonction Zéta ;
5. Retourner le polynôme caractéristique de la norme matricielle de cette matrice.

L'étape 1 n'est pas vraiment algorithmique : il s'agit d'un travail théorique que l'on doit refaire pour chaque famille de courbes considérées. Les étapes 2 et 3 sont des précalculs qui ne dépendent que de la famille de courbes considérées. Les étapes 4 et 5 sont à effectuer pour chaque courbe de la famille dont on veut la fonction Zéta.

L'algorithme ainsi esquissé a été mis en œuvre avec succès par Castryck, Gerkmann, Hubrechts, Lauder, Vercauteren [79, 90, 91, 92, 20, 106], pour diverses classes de courbes ou de variétés. Un des points les plus cruciaux lors de la conception de ces algorithmes est de bien contrôler les pertes de précision de manière à pouvoir tronquer les séries au degré adéquat et garantir l'intégralité des objets intermédiaires.

3.7 Autres travaux «constructifs»

3.7.1 Multiplication complexe en genre 2

En grande caractéristique, le calcul du nombre de points d'une courbe de genre 2 étant encore problématique, il est nécessaire de disposer aussi de la méthode de multiplication complexe (CM). Par ailleurs, cette méthode permet de construire des courbes ayant des propriétés supplémentaires intéressantes ; c'est en particulier une brique de base pour la construction de courbes admettant des couplages efficaces [52, 53, 51, 96]. Bien maîtrisée depuis une trentaine

d'années [4], la méthode CM en genre 1 a été étendue au genre 2 relativement tôt par Spallek [158] et van Wamelen [168] ; toutefois, ça n'est qu'avec les travaux de Weng [172, 173] que l'on a pu traiter des nombres de classes non triviaux.

La méthode CM classique utilise des calculs flottants, ce qui induit des problème de contrôle de précision. Pour contourner ces difficultés, en collaboration avec Houtmann, Kohel, Ritzenthaler et Weng, nous avons développé une méthode 2-adique pour la construction de courbes à multiplication complexe. Nous allons exposer brièvement le fonctionnement de cette méthode avant de discuter ses avantages, ses inconvénients, et les développements ultérieurs qui ont eu lieu.

La méthode CM fonctionne en deux étapes :

1. Construction d'un idéal CM associé à un corps CM ;
2. Construction d'une courbe CM et de son nombre de points à partir de cet idéal.

La première phase est un précalcul : les polynômes définissant un idéal CM peuvent être stockés et utilisés pour construire de nombreuses courbes en répétant la deuxième phase. C'est aussi cette première phase de précalcul qui est la plus délicate à mettre en œuvre.

Définition 12 *Un corps à multiplication complexe de degré 4 est un corps de nombres K qui est une extension totalement imaginaire d'un corps quadratique réel K_0.*

Le lien avec les courbes de genre 2 est le suivant. Si l'on considère une courbe de genre 2 définie sur un corps fini dont la jacobienne est simple et ordinaire, alors l'anneau d'endomorphismes de cette jacobienne est isomorphe à un ordre dans un corps CM. Ce résultat profond permet de mieux saisir l'importance de l'aspect CM quant au calcul du nombres de points. En effet, l'endomorphisme de Frobenius doit correspondre à un élément du corps CM, et comme de plus sa norme relative est imposée, il ne reste que très peu de possibilités pour son polynôme caractéristique. Le nombre de points de la jacobienne se calcule alors facilement.

Ainsi, pour résumer, si l'on a une courbe dont on sait que l'anneau d'endomorphismes est isomorphe à un ordre connu d'un corps CM, alors le nombre de points de sa jacobienne se déduit de la résolution d'une équation aux normes dans le corps CM, ce qui se fait de manière relativement efficace.

Il reste à réussir à forcer l'anneau d'endomorphismes. Pour cela, il va falloir considérer des courbes définies sur $\mathbb{Q}$, voire sur un corps de nombres L. Dans ce cas, l'anneau d'endomorphismes n'est pas forcément un ordre d'un corps CM, c'est même exceptionnellement le cas, car en général les seuls endomorphismes sont ceux issus de la loi de groupe : les morphismes de multiplication par une constante. Cela dit, si l'on a réussi à construire une courbe sur $\mathbb{Q}$ ayant CM par un ordre de K, en réduisant son équation modulo un nombre premier p, on va obtenir une courbe sur $\mathbb{F}_p$, et son anneau d'endomorphismes va contenir la réduction des endomorphismes qui existaient sur $\mathbb{Q}$. Sauf pour un nombre fini de nombres premiers p, la courbe obtenue est bien telle qu'on le souhaite. Ensuite pour une proportion raisonnable de p, on n'aura pas plus d'endomorphismes que ceux obtenus par réduction, si bien que l'anneau d'endomorphismes est le même que celui dont on est parti.

La théorie de la multiplication complexe fournit un moyen de décrire ses courbes : à partir d'un ordre d'un corps CM K, on peut définir un corps L sur lequel seront définies toutes les courbes à multiplication complexe par l'ordre considéré. Il n'y a en fait qu'un nombre fini de telles courbes (à isomorphisme près). Le cœur de la méthode CM est une méthode algorithmique pour calculer ces courbes.

Comme l'on travaille à isomorphisme près, plutôt que de décrire une courbe par son équation, il est plus pratique d'utiliser ses invariants d'Igusa : il s'agit de 3 scalaires (j_1, j_2, j_3) qui caractérisent la classe d'isomorphisme d'une courbe, tout comme le j-invariant d'une courbe elliptique caractérise sa classe d'isomorphisme.

Définition 13 *L'idéal CM associé à un ordre d'un corps CM est l'idéal des polynômes en 3 variables qui s'annulent exactement en les invariants d'Igusa des courbes ayant multiplication complexe par cet ordre.*

Exemple : Le corps $K = \mathbb{Q}(i\sqrt{23 + 4\sqrt{5}})$ est un corps CM. On peut montrer que l'idéal CM associé à l'anneau des entiers de K (i.e. l'ordre maximal) est défini par les trois polynômes suivants :

$$H_1(j_1) = 0,\ H_1'(j_1)j_2 = G_2(j_1),\ H_1'(j_1)j_3 = G_3(j_1).$$

où les polynômes H_1, G_2, G_3 sont les suivants :

$$\begin{aligned}
H_1 &= 2^{18}5^{36}7^{24}\ T^6 \\
&\quad - 1118773039927368977400974047014016967290290543651580810546875000\ T^5 \\
&\quad + 501512527690591679504420832767471421512684501403834547644662988263671875000\ T^4 \\
&\quad - 101124092427873917866762846337305750476145431355720256674682214327042638578082629 23\ T^3 \\
&\quad + 118287000250588667564540744739406154398135978447792771928535541240797386992091828213521875\ T^2 \\
&\quad - 2^1 3^{50} 5^{10} 11^1 13^1 53^1 701^1 16319^1 69938793494948953569198870004032131926868578084899317\ T \\
&\quad + 3^{60} 5^{15} 23^5 409^5 179364113^5 \\
G_2 &= 2^{-3}(27342492849745895420865597820165639113330322809219360351562500 00\ T^5 \\
&\quad + 57554607277149797568849387967258354564256002479144001401149377453125000000\ T^4 \\
&\quad + 24021378160854085829663614804129234099772970403767605010145433823381894838618879 23\ T^3 \\
&\quad - 75691166837057576824962404339816428897154828109931810138346946500235981947587900092046875\ T^2 \\
&\quad + 2^1 3^{48} 5^{10} 35828519670812312117443096939126403484719666514876459782054400437\ T \\
&\quad - 3^{58} 5^{15} 11^1\ 13^2 23^3 409^3 23879^1 179364113^3 370974539856105277) \\
G_3 &= 2^{-4}(20062002297726501938753962499493388123426921176910400390625000 0\ T^5 \\
&\quad - 23006467431764975697282545882188900514908468992554759536043135578125000000\ T^4 \\
&\quad + 61501729461967806861131941471814416154508821826021421156385015129113664689498754 7\ T^3 \\
&\quad - 14310698742415340178789612716269929249317950024503557714370659520249839645781463819312875\ T^2 \\
&\quad - 2^1 3^{46} 5^8 13^1 61^1 18373951326869^1 257132885872612082121079857244680586515097341609 07\ T \\
&\quad + 3^{55} 5^{13} 23^2 409^2 23561^1 440131^1 179364113^2 45198640235201788172471264168 9)
\end{aligned}$$

Notons que cette représentation de l'idéal, bien que classique en calcul formel (il s'agit d'une représentation univariée rationnelle) n'était pas utilisée dans le cadre de la multiplication complexe avant nos travaux, ce qui posait des problèmes de description précise de l'idéal ou de tailles des polynômes.

La méthode CM en genre 2 a été développée par Spallek, Weng et van Wamelen [158, 172, 168] en suivant une approche flottante. Le principe est de calculer des approximations complexes des invariants de toutes les courbes à multiplication complexe par l'ordre donné. Pour cela il est nécessaire d'évaluer des fonctions transcendantes à très grande précision en des points que l'on déduit par des calculs de théorie algébrique des nombres dans le corps CM considéré. Une fois les invariants calculés, on peut calculer des approximations des coefficients des polynômes H_1, G_2, G_3. Il reste ensuite à reconnaître ces nombres complexes comme des rationnels, ce qui se fait grâce à la théorie des fractions continues.

Afin de contourner les problèmes liés à l'évaluation des fonctions complexes, et en particulier le contrôle des chutes de précision des calculs, on peut remplacer l'approche complexe classique par une approche p-adique.

Le point de départ n'est plus le corps CM et les calculs de théorie algébrique des nombres que l'on y effectue d'habitude : on part désormais d'une courbe définie sur un petit corps fini

et qui a CM par un ordre qui nous intéresse. Comme le corps fini est petit, on dispose d'algorithmes suffisamment efficaces pour travailler sur la courbe, en particulier compter ses points, et dans bien des cas vérifier que l'on a CM par l'ordre maximal dans le corps. Ensuite, en utilisant les algorithmes développés dans le contexte du comptage de points (voir section 3.4), on peut relever cette courbe de manière canonique sur un corps p-adique. Cela signifie que l'on conserve l'anneau d'endomorphismes, et donc celui-ci continue d'être isomorphe à l'ordre du corps CM considéré. Une fois que l'on a relevé la courbe avec suffisamment de précision, on peut calculer ses invariants d'Igusa. Il s'agit de nombres p-adiques qu'on connaît à grande précision et qui doivent être racines de l'idéal cherché. L'algorithme de réduction de réseau LLL peut alors être utilisé pour reconstruire les polynômes H_1, G_2, G_3 cherchés.

L'algorithme est donc très similaire à l'algorithme classique, mais on doit disposer d'une courbe sur un petit corps fini ayant les bonnes propriétés comme point de départ. En échange, les problèmes de perte de précision sont essentiellement éliminés.

Nous avons mis en œuvre cette approche dans le cas où l'on part d'une courbe en caractéristique 2 ; en effet, on peut alors relever vers un corps 2-adique par les formules d'isogénies de Richelot. Par exemple, considérons la courbe $\mathcal{C}$ d'équation $y^2 + h(x)y + f(x) = 0$ sur $\mathbb{F}_8 = \mathbb{F}_2[t]/(t^3 + t + 1)$, avec

$$\begin{aligned} f(x) &= x^5 + t^6x^3 + t^5x^2 + t^3x \ , \\ h(x) &= x^2 + x \ . \end{aligned}$$

Le polynôme caractéristique du Frobenius de $\mathcal{C}$ est $\chi_\pi(T) = T^4 + 4T^3 + 15T^2 + 32T + 64$ qui est un polynôme de définition possible pour le corps $K = \mathbb{Q}(i\sqrt{23 + 4\sqrt{5}})$ déjà mentionné ci-dessus. On peut facilement vérifier que cette courbe a multiplication complexe par l'ordre maximal du corps K. L'algorithme de Richelot permet de relever cette courbe sur un corps 2-adique, et finalement, ses invariants sont branchés dans l'algorithme LLL qui fournit les polynômes H_1, G_2, G_3 ci-dessus. Imaginons maintenant que l'on souhaite fabriquer un cryptosystème de genre 2. Le nombre premier $p = 954090659715830612807582649452910809$ de 120 bits va convenir : on peut vérifier en résolvant une équation aux normes dans K que la jacobienne d'une courbe CM par l'ordre maximal de K aura un ordre premier de 240 bits :

$$910288986956988885753118558284481029311411128276048027584310525408884449$$

Une courbe correspondante est alors construite en cherchant un point d'annulation de l'idéal CM associé modulo p : on trouve des invariants à partir desquels l'algorithme de Mestre [127] permet de reconstruire une équation de courbe :

$$\begin{aligned} \mathcal{C} : y^2 = x^6 &+ 827864728926129278937584622188769650\, x^4 \\ &+ 102877610579816483342116736180407060\, x^3 \\ &+ 335099510136640078379392471445640199\, x^2 \\ &+ 351831044709132324687022261714141411\, x \\ &+ 274535330436225557527308493450553085. \end{aligned}$$

Extensions. Ces travaux ont été étendus par Carls, Kohel et Lubicz [16] de manière à pouvoir utiliser un relèvement 3-adique, ce qui augmente le nombre de courbes pouvant être traitées par une méthode évitant les calculs flottants. Kohel a ensuite effectué une étude plus systématique [100] des relations entre le ℓ-adique et la multiplication complexe.

Entre temps, la méthode CM utilisant les flottants a elle-même été améliorée, suite aux travaux de Dupont et Houtmann, si bien que les méthodes p-adiques restent intéressantes surtout pour leur simplicité d'implantation.

Mentionnons de plus des travaux théoriques [80] sur les corps CM de genre 2, ainsi qu'une méthode alternative s'appuyant sur le théorème chinois [38].

3.7.2 Formules efficaces pour la loi de groupe en genre 2

Rendre les cryptosystèmes hyperelliptiques aussi populaires que les cryptosystèmes elliptiques ne pourra se faire que s'ils présentent un certain avantage en termes d'efficacité. Pour cela, optimiser les formules pour la loi de groupe est crucial. De nombreux travaux ont été effectués afin de réduire le nombre d'opérations nécessaires pour les genre 2, 3 et 4. De nos jours, suite aux améliorations successives des algorithmes de logarithme discret, il paraît raisonnable de se concentrer sur les courbes de genre 2. Pour celles-ci, on dispose de diverses formules correspondant à divers systèmes de coordonnées. Nous renvoyons à [25] pour un panorama complet de la situation en 2005, ainsi qu'une liste de références, que l'on peut compléter par les publications plus récentes suivantes : [45, 46, 44, 101, 175, 5]. On peut résumer la situation ainsi : en caractéristique impaire, une opération de groupe coûte environ 25 multiplications et 1 inversion ; en caractéristique 2, cela coûte environ 20 multiplications et 1 inversion. Si l'on ne souhaite pas faire d'inversions, on peut prendre des coordonnées projectives. Le coût d'une opération est alors de l'ordre d'une quarantaine de multiplications quelle que soit la caractéristique.

Afin d'améliorer ces coûts, on peut s'inspirer de ce qu'on appelle les formules de Montgomery pour les courbes elliptiques. Il s'agit de formules qui ne font intervenir que les abscisses des points d'une courbe elliptique. En effet, si $P = (x, y)$ est un point d'une courbe elliptique, pour tout n, l'abscisse de nP ne dépend que de x et pas de y. Les formules de Montgomery sont suffisantes pour bien des protocoles (par exemple l'échange de clef Diffie-Hellman) où l'on ne fait pas appel à la loi de groupe autrement que par multiplication d'un point par un scalaire. Ces formules s'avèrent extrêmement efficaces aussi bien pour une implantation logicielle que pour une implantation matériel. Étendre les formules de Montgomery aux courbes de genre 2 a été tenté à plusieurs reprises [156, 37, 102] sans parvenir à battre les formules classiques.

En nous inspirant de travaux de Chudnovsky et Chudnovsky [23], nous avons proposé de telles formules qui cette fois-ci permettent d'accélérer grandement les calculs. La comparaison avec la situation des courbes elliptiques est aussi avantageuse. L'idée principale est de partir de formules de duplication des fonctions Thêta. Ces formules sont a priori valides seulement sur $\mathbb{C}$, et de fait, les fonctions Thêta prennent des valeurs transcendantes en les points qui pourraient nous intéresser, si bien que les réduire modulo p ne semble guère avoir de sens. Cependant, certains quotients de fonctions Thêta sont de nature algébrique, si bien qu'en travaillant en projectif, on a des formules qui garderont un sens sur un corps fini.

Les courbes de genre 2 sur $\mathbb{C}$. La jacobienne d'une courbe de genre 2 sur $\mathbb{C}$ est une variété abélienne de dimension 2. On peut montrer que ces objets s'obtiennent comme des tores complexes particuliers. L'ensemble des matrices symétriques 2×2 à coefficients complexes, et dont la partie imaginaire est définie positive est appelé le demi-espace de Siegel. C'est l'analogue en dimension supérieure du demi-plan de Poincaré (les nombres complexes de partie imaginaire strictement positive). Soit Ω une matrice du demi-espace de Siegel. Alors

$\mathbb{Z}^2 + \Omega\mathbb{Z}^2$ est un réseau L_Ω de $\mathbb{C}^2$, et son quotient associé $\mathbb{C}^2/L_\Omega$ est un tore complexe de dimension 2. On peut de plus montrer que c'est une variété abélienne, et que toute variété abélienne complexe de dimension 2 est isomorphe à un tore complexe de ce type.

Les fonctions Thêta fournissent un moyen de définir des fonctions du tore vers $\mathbb{C}$.

Définition 14 *La fonction Thêta de Riemann associée à Ω est définie, pour $\mathbf{z} \in \mathbb{C}^2$, par :*

$$\vartheta(\mathbf{z}, \Omega) = \sum_{n \in \mathbb{Z}^2} \exp\left(\pi i\, {}^t n \Omega n + 2\pi i\, {}^t n \cdot \mathbf{z}\right).$$

Plus généralement, on définit les fonctions Thêta avec caractéristique $[a, b]$ où a et b sont des vecteurs dans $\mathbb{Q}^2$, par :

$$\vartheta[a; b](\mathbf{z}, \Omega) = \exp\left(\pi i\, {}^t a \Omega a + 2\pi i\, {}^t a \cdot (\mathbf{z} + b)\right) \cdot \vartheta(\mathbf{z} + \Omega a + b, \Omega).$$

Les caractéristiques qui vont nous intéresser sont les vecteurs a et b dont les coordonnées sont dans $\{0, \frac{1}{2}\}$. Ainsi les 16 fonctions formées correspondent essentiellement à des translatées de la fonction Thêta de Riemann par les points de 2-torsion du réseau. Ces fonctions ne sont pas directement périodiques de période L_Ω. Mais le facteur multiplicatif supplémentaire ajouté aux fonctions avec caractéristique permet de pallier ce défaut de périodicité, du moins projectivement : la fonction de $\mathbb{C}^2$ vers $\mathbb{P}^{15}$ qui à $\mathbf{z}$ associe les valeurs des fonctions Thêta en $2\mathbf{z}$ est invariante si l'on ajoute à $\mathbf{z}$ un élément de L_Ω, si bien que l'on obtient un plongement du tore $\mathbb{C}^2/L_\Omega$ dans un espace projectif, les fonctions θ servant de fonctions coordonnées.

Une autre propriété fondamentale des fonctions Thêta est obtenue en injectant le vecteur nul à la place de $\mathbf{z}$: on obtient des scalaires appelés Thêta-constantes. Ces scalaires sont liés à la classe d'isomorphisme de la variété abélienne. Si l'on fait agir une matrice de $\mathrm{Sp}_4(\mathbb{Z})$ sur Ω, on obtient une nouvelle matrice du demi-espace de Siegel, et l'on peut montrer que l'on obtient une variété abélienne isomorphe à la variété abélienne de départ. Réciproquement, pour deux matrices donnant des variétés isomorphes, on peut passer de l'une à l'autre par l'action d'une matrice de $\mathrm{Sp}_4(\mathbb{Z})$. Les Thêta-constantes ne sont pas invariantes si l'on change une matrice par une matrice équivalente sous $\mathrm{Sp}_4(\mathbb{Z})$. Toutefois, là encore, le défaut d'invariance est suffisamment simple pour pouvoir être corrigé : en prenant des carrés de quotients de Thêta-constantes appropriées, on obtient de vrais invariants.

Surface de Kummer. Les formules de Montgomery en genre 1 n'opèrent que sur les abscisses des points. De manière similaire, on souhaite avoir des formules pour le genre 2 qui n'opère que sur la surface de Kummer associée à la courbe, c'est-à-dire que l'on va définir des opérations entre les couples {élément, élément opposé}. En terme de fonctions Thêta, cela signifie qu'on ne va utiliser que certaines fonctions Thêta paires, puisque celles-ci envoient $\mathbf{z}$ et $-\mathbf{z}$ sur le même point. On vérifie facilement qu'en prenant les caractéristiques $[a; b]$, où a est le vecteur $(0, 0)$, on obtient des fonctions paires. On va ainsi considérer les fonctions suivantes :

$$\begin{aligned}
\vartheta_1(\mathbf{z}) &= \vartheta[(0,0);(0,0)](\mathbf{z}, \Omega) \\
\vartheta_2(\mathbf{z}) &= \vartheta[(0,0);(\tfrac{1}{2},\tfrac{1}{2})](\mathbf{z}, \Omega) \\
\vartheta_3(\mathbf{z}) &= \vartheta[(0,0);(\tfrac{1}{2},0)](\mathbf{z}, \Omega) \\
\vartheta_4(\mathbf{z}) &= \vartheta[(0,0);(0,\tfrac{1}{2})](\mathbf{z}, \Omega)\ .
\end{aligned}$$

La surface de Kummer $\mathcal{K}_\Omega$ est alors définie comme l'image dans $\mathbb{P}^3(\mathbb{C})$ de la fonction

$$\varphi\ :\ \mathbf{z} \mapsto (\vartheta_1(2\mathbf{z}), \vartheta_2(2\mathbf{z}), \vartheta_3(2\mathbf{z}), \vartheta_4(2\mathbf{z}))\,.$$

Il s'agit d'une variété projective de dimension 2, qui peut être décrite par une équation. Rajoutons pour cela quelques notations. Soient a, b, c, d, les Thêta-constantes associées aux fonctions que l'on a choisies :

$$a = \vartheta_1(0),\ b = \vartheta_2(0),\ c = \vartheta_3(0),\ d = \vartheta_4(0).$$

On définit ensuite de nouvelles constantes A^2, B^2, C^2, D^2, E, F, G, H à partir de a, b, c, d :

$$\begin{aligned}
4A^2 &= a^2 + b^2 + c^2 + d^2,\\
4B^2 &= a^2 + b^2 - c^2 - d^2,\\
4C^2 &= a^2 - b^2 + c^2 - d^2,\\
4D^2 &= a^2 - b^2 - c^2 + d^2.
\end{aligned}$$

$$\begin{aligned}
E &= 256\,abcdA^2B^2C^2D^2/(a^2d^2 - b^2c^2)(a^2c^2 - b^2d^2)(a^2b^2 - c^2d^2)\\
F &= (a^4 - b^4 - c^4 + d^4)/(a^2d^2 - b^2c^2)\\
G &= (a^4 - b^4 + c^4 - d^4)/(a^2c^2 - b^2d^2)\\
H &= (a^4 + b^4 - c^4 - d^4)/(a^2b^2 - c^2d^2)\ .
\end{aligned}$$

L'équation de la surface de Kummer $\mathcal{K}$ est alors donnée par la formule suivante, où x, y, z, t sont les coordonnées correspondantes aux fonctions Thêta choisies :

$$(x^4 + y^4 + z^4 + t^4) + 2Exyzt - F(x^2t^2 + y^2z^2) - G(x^2z^2 + y^2t^2) - H(x^2y^2 + z^2t^2) = 0.$$

On trouve cette formule déjà dans la littérature du XIX$^{\text{e}}$ siècle ; mais on peut se référer à [7] pour un traitement moderne. Sur $\mathcal{K}$, on n'a pas à proprement parler de loi de groupe. Toutefois, on a une pseudo-loi de groupe héritée de la loi de groupe dans la jacobienne : si $P = (x, y, z, t)$ est le point correspondant à $\varphi(\mathbf{z})$, on peut calculer $2P$ le point correspondant à $\varphi(2\mathbf{z})$, grâce aux formules de doublement des fonctions Thêta. Celles-ci se traduisent par l'algorithme de doublement suivant :

Algorithme de doublement : `DoubleKummer(P)`
Entrée : Un point $P = (x, y, z, t)$ de $\mathcal{K}$;
Sortie : Le double $2P = (X, Y, Z, T)$ dans $\mathcal{K}$.

1. $x' = (x^2 + y^2 + z^2 + t^2)^2/A^2$;
2. $y' = (x^2 + y^2 - z^2 - t^2)^2/B^2$;
3. $z' = (x^2 - y^2 + z^2 - t^2)^2/C^2$;
4. $t' = (x^2 - y^2 - z^2 + t^2)^2/D^2$;
5. $X = (x' + y' + z' + t')/a$;
6. $Y = (x' + y' - z' - t')/b$;
7. $Z = (x' - y' + z' - t')/c$;
8. $T = (x' - y' - z' + t')/d$;
9. Retourner (X, Y, Z, T).

Pour l'addition la situation est plus compliquée : connaître deux points sur la surface de Kummer ne suffit pas pour pouvoir les additionner ; en effet il est impossible de distinguer

leur somme de leur différence. C'est pourquoi on dispose d'un algorithme de pseudo-addition, tout comme dans le cas des formules de Montgomery.

Algorithme de pseudo-addition : `PseudoAddKummer(P, Q, R)`
Entrée : Deux points $P = (x, y, z, t)$ et $Q = (\underline{x}, \underline{y}, \underline{z}, \underline{t})$ de $\mathcal{K}$ et $R = (\bar{x}, \bar{y}, \bar{z}, \bar{t})$ l'un des points $P + Q$ ou $P - Q$, avec $\bar{x}\bar{y}\bar{z}\bar{t} \neq 0$.
Sortie : Le point (X, Y, Z, T) de $\mathcal{K}$ parmi $P + Q$ et $P - Q$ qui est différent de R.

1. $x' = (x^2 + y^2 + z^2 + t^2)(\underline{x}^2 + \underline{y}^2 + \underline{z}^2 + \underline{t}^2)/A^2$;
2. $y' = (x^2 + y^2 - z^2 - t^2)(\underline{x}^2 + \underline{y}^2 - \underline{z}^2 - \underline{t}^2)/B^2$;
3. $z' = (x^2 - y^2 + z^2 - t^2)(\underline{x}^2 - \underline{y}^2 + \underline{z}^2 - \underline{t}^2)/C^2$;
4. $t' = (x^2 - y^2 - z^2 + t^2)(\underline{x}^2 - \underline{y}^2 - \underline{z}^2 + \underline{t}^2)/D^2$;
5. $X = (x' + y' + z' + t')/\bar{x}$;
6. $Y = (x' + y' - z' - t')/\bar{y}$;
7. $Z = (x' - y' + z' - t')/\bar{z}$;
8. $T = (x' - y' - z' + t')/\bar{t}$;
9. Retourner (X, Y, Z, T).

Ces deux opérations permettent d'avoir une multiplication scalaire d'un point dans la surface de Kummer, par exemple en utilisant une méthode binaire où l'on effectue un doublement et une pseudo-addition pour chaque bit du multiplieur, ou bien par l'algorithme PRAC de Montgomery.

En organisant correctement les calculs à partir des formules ci-dessus, on obtient un coût par bit du multiplieur de 9 carrés, 10 multiplications génériques et 6 multiplications par des constantes qui ne dépendent que de la surface de Kummer choisie. Si ces constantes peuvent être choisies petites, on peut organiser les calculs différemment afin d'avoir 12 carrés, 7 multiplications génériques et 9 multiplications par des constantes de la surface.

Caractéristique 2. L'algorithme tel qu'il est décrit ci-dessus ne fonctionne pas en caractéristique 2 ; en effet, il s'appuie de manière forte sur la 2-torsion, qui change de comportement en caractéristique 2. Dans le cas où l'on considère une courbe de genre 2 dont la jacobienne est ordinaire, on peut toutefois effectuer le cheminement suivant. Soit $\mathcal{C}$ une courbe ordinaire sur $\mathbb{F}_{2^n}$, et soit $\mathfrak{C}$ son relevé canonique sur un corps de nombres K. Comme K est inclus dans $\mathbb{C}$, on peut effectuer la construction de la surface de Kummer ci-dessus, et en particulier trouver son équation et une pseudo loi de groupe donnée par les mêmes formules que ci-dessus. Ces formules ont mauvaises réduction en 2, toutefois, en effectuant un changement de variables, on peut les rendre réductible modulo 2. Les nouvelles formules de pseudo loi de groupe que l'on obtient par ce changement de variables s'avèrent être particulièrement simples. Nous ne rentrerons pas plus dans les détails, et renvoyons à l'article [73] pour une description complète des formules. Lors d'une exponentiation binaire, on obtient un coût par bit du multiplieur de 9 carrés (très peu coûteux en caractéristique 2), 15 multiplications génériques et 3 multiplications par des constantes qui ne dépendent que de la surface de Kummer choisie.

Le cheminement qui nous a permis de trouver les formules ne fournit pas une preuve rigoureuse ; en effet, rien ne garantit que dans le procédé de relèvement / réduction, on n'a pas perdu la cohérence. En collaboration avec Lubicz [73], nous avons toutefois obtenu une démonstration rigoureuse au prix de l'utilisation de la théorie des fonctions Théta algébriques.

Courbes elliptiques. Il est à noter que dériver des formules en s'appuyant sur les fonctions Théta peut aussi être fait pour les courbes elliptiques. En caractéristique 2, on retrouve alors les formules de Stam [159]. En caractéristique impaire, on trouve des formules qui ne sont pas dans la littérature, mais qui ressemblent beaucoup aux formules de Montgomery ; nous renvoyons de nouveau à [73] pour la description de ces formules.

Expériences et temps de calcul. Nous avons implanté ces formules ainsi que les formules de Montgomery afin de comparer les efficacités pour une implantation logicielle sur une station de travail. Pour cela nous nous sommes appuyés sur la bibliothèque $\texttt{mp}\mathbb{F}_q$ décrite en section 4.2.

Les temps de calcul obtenus sont résumés dans le tableau suivant. Il s'agit de nombres de cycles pour effectuer un échange de clef de type Diffie-Hellman. Notons que l'implantation n'est pas optimisée pour les machines 32 bits, ce qui explique (en partie seulement) la grande différence de temps de calcul entre un Pentium 4 et un Core2.

	`curve25519`	`surf127eps`	`curve2251`	`surf2113`
Opteron K8	310,000	296,000	1,400,000	1,200,000
Core2	386,000	405,000	888,000	687,000
Pentium 4	3,570,000	3,300,000	3,085,000	2,815,000
Pentium M	1,708,000	2,000,000	2,480,000	2,020,000

La colonne `curve25519` correspond à un cryptosystème elliptique sur $\mathbb{F}_{2^{255}-19}$, la colonne `curve2251` à un système elliptique sur $\mathbb{F}_{2^{251}}$, la colonne `surf127eps` à un système de genre 2 sur $\mathbb{F}_{2^{127}-735}$, et la colonne `surf2113` à un système de genre 2 sur $\mathbb{F}_{2^{113}}$.

3.7.3 Perspectives

Il est assez clair que l'on peut aussi obtenir des formules efficaces en genre 3 et 4 par la méthode esquissée ici. La partie délicate est de relier les formules que l'on pourrait obtenir aux coordonnées classiques (représentation de Mumford des courbes hyperelliptiques). En effet, les variétés abéliennes principalement polarisées de dimension 3 sont des jacobiennes, mais pas forcément des jacobiennes de courbes hyperelliptiques ; et en dimension 4, on n'a même pas la garantie d'être une jacobienne de courbe.

Un point que nous avons passé sous silence dans notre description rapide est une condition assez forte de rationalité : pour qu'une courbe de genre 2 puisse avoir une surface de Kummer sous la forme choisie, il est nécessaire d'avoir toute la 2-torsion rationnelle. Il est probable que lever cette restriction se fera au prix d'une pseudo-loi de groupe plus coûteuse, mais il serait intéressant de rendre ce surcoût aussi faible que possible.

Une autre extension envisageable est de traiter le cas des courbes non-ordinaires en caractéristique 2. En effet, pour ces courbes la loi de groupe classique en représentation de Mumford peut être accéléré ; on peut espérer le même genre de gain avec des coordonnées Théta. Toutefois, il reste des obstructions théoriques à lever pour y parvenir.

Chapitre 4

Quelques travaux liés à l'arithmétique efficace

Dans ce dernier chapitre, on trouve mes contributions à l'arithmétique efficace. Un première thématique est celle de la multiplication de grands entiers par l'algorithme de Schönhage-Strassen [72]. Le second volet est la conception d'une bibliothèque de corps finis, appelée $\texttt{mp}\mathbb{F}_q$ dont la vocation première est l'efficacité dans le cas des corps utiles pour la cryptographie [78].

4.1 L'algorithme de Schönhage-Strassen

La multiplication des entiers est un problème élémentaire, pour lequel on dispose de nombreux algorithmes et des implantations satisfaisantes. Il s'agit aussi d'un problème fondamental. De nombreux autres problèmes arithmétiques ont une complexité qui se réduit à celle de la multiplication. Par exemple, les meilleurs algorithmes connus (théoriques aussi bien que pratiques) pour effectuer une division euclidienne ou un calcul de PGCD sont des algorithmes qui s'appuient de manière fondamentale sur le fait que l'on sait multiplier deux entiers rapidement. La méthode de substitution de Kronecker (aussi appelée méthode de segmentation) est un autre exemple : pour multiplier deux polynômes à coefficients entiers, on peut se ramener directement au problème de multiplier de grands entiers. En pratique, c'est parfois la manière la plus efficace de faire ; par ailleurs, d'un point de vue logiciel, il est très attirant d'avoir une primitive simple sur laquelle on peut bâtir autant d'algorithmes complexes : toute amélioration de cette primitive permet d'accélérer toute une famille d'algorithmes.

Jusqu'à très récemment, le meilleur algorithme asymptotique connu pour multiplier deux entiers était l'algorithme de Schönhage-Strassen [144] (le récent résultat de Fürer [56] vient de changer la situation). Malgré cela, il n'y a pas beaucoup d'implantations existantes de cet algorithme. Ceci est probablement dû en partie au fait qu'il ne devient intéressant que pour des entiers de taille qui semblait déraisonnable jusqu'au début des années 90. Pour les entiers plus petits que (disons) 100,000 chiffres décimaux, les algorithmes de Karatsuba ou de Toom-Cook sont plus efficaces.

La meilleure implantation disponible en logiciel libre est celle de la bibliothèque GMP[1], qui permet de manipuler des nombres en précision arbitraire. Une autre implantation non-libre est intégrée au logiciel de calcul formel Magma. Cette dernière implantation est plus efficace que celle de la version 4.1.4 de GMP et comparable à celle de la version 4.2.1 de GMP. Partant de cette dernière version de GMP, en collaboration avec Alexander Kruppa et Paul Zimmermann, nous avons travaillé sur des améliorations algorithmiques et d'implantations que nous allons brièvement présenter, et qui ont abouti à des temps de calculs jusqu'à deux fois plus courts que ceux de Magma.

Principe général de l'algorithme. L'algorithme de Schönhage-Strassen est un algorithme de multiplication par transformée de Fourier rapide (FFT). Comme dans tous les algorithmes de ce type, on commence par convertir les entiers a et b à multiplier en polynômes à coefficients entiers $A(x)$ et $B(x)$. Les chiffres binaires de a sont regroupés en paquets qui forment les coefficients de $A(x)$, ou plus formellement, $A(x)$ est le polynôme qui redonne l'entier a lorsque l'on évalue $A(x)$ en une puissance de 2 bien choisie. Ensuite les coefficients entiers de $A(x)$ et $B(x)$ sont plongés dans un anneau R pour lequel on dispose d'un algorithme de FFT (ce qui signifie que l'on souhaite que R contiennent suffisamment de racines de l'unité). On obtient ainsi deux polynômes $\overline{A}(x)$ et $\overline{B}(x)$ dans $R[x]$. On calcule leur transformées de Fourier respectives, on multiplie les coefficients point à point dans R, puis on calcule la transformée de Fourier inverse du résultat. Il ne reste ensuite plus qu'à remonter à un résultat dans $\mathbb{Z}[x]$, puis dans $\mathbb{Z}$. Il faut bien entendu imposer des conditions pour que le résultat ait bien le sens escompté une fois que l'on est revenu à un entier.

Des choix naturels pour l'anneau R sont les nombres complexes, ou un corps fini premier. Dans ces deux cas, on va pouvoir trouver (ou imposer) suffisamment de racines de l'unité

[1] `http://gmplib.org`

pour pouvoir gérer des transformées de Fourier d'ordre très grand, ainsi on peut considérer des polynômes $A(x)$ et $B(x)$ ayant un grand degré et des coefficients petits – typiquement, le nombre de bits des coefficients sera de l'ordre du logarithme du degré. Le problème (théorique aussi bien que pratique) de cette approche est que les racines de l'unité sont a priori des éléments quelconque de R, et les multiplications par celles-ci qui ont lieu en grand nombre durant une transformée de Fourier rapide vont être coûteuses. L'idée de l'algorithme de Schönhage-Strassen est de prendre pour R un anneau de la forme $\mathbb{Z}/(2^{2^k}+1)\mathbb{Z}$. L'entier 2 est alors une racine 2^{k+1} ième de l'unité dans R, et l'on pourra donc effectuer des transformées de Fourier de polynômes de degré 2^{k+1} sans perte d'information. On voit donc que dans ce cas, le degré des polynômes à considérer va être du même ordre de grandeur que le nombre de bits de chaque coefficient. En revanche, durant une transformée de Fourier, les multiplications par des racines de l'unité seront extrêmement peu coûteuses, puisqu'il s'agira de multiplications par des puissances de 2, qui reviennent à faire des décalages sur les bits. Les seules opérations coûteuses sont les multiplications point à point, mais on en fait beaucoup moins que les opérations durant les transformations de Fourier. Pour conclure sur ce bref aperçu de l'algorithme de Schönhage-Strassen, il faut mentionner que ces multiplications point à point dans R peuvent (doivent !) ensuite être effectuées en appelant récursivement l'algorithme. On obtient alors la fameuse complexité de la multiplication en $\mathsf{M}(n) = O(n \log n \log\log n)$, pour multiplier deux entiers de n bits.

Améliorations apportées. Nous avons travaillé sur plusieurs aspects de l'implantation ; chacune des améliorations gagnant quelques pourcents sur le temps de calcul. Nous allons mentionner quelques-unes de ces améliorations. Des détails supplémentaires sont donnés dans l'article [72] ; le code source est aussi distribué sous licence libre LGPL.

Une part très importante du temps de calcul est dû à des opérations élémentaires (addition, soustraction, multiplication par une puissance de 2) dans l'anneau $R = \mathbb{Z}/(2^{2^k}+1)\mathbb{Z}$. Écrire soigneusement celles-ci en évitant au maximum les branchements difficiles à prédire pour le processeur a permis de gagner sur le temps de calcul.

Comme présenté ci-dessus, l'algorithme de Schönhage-Strassen est récursif. Aux niveaux internes de récursion, comme on souhaite calculer un résultat dans R, il y a un découpage très naturel des entiers modulaires en polynômes, de telle sorte que la transformée de Fourier subséquente se comporte bien vis-à-vis du modulo qui définit R. Moralement, une transformée de Fourier permet de multiplier des polynômes modulo $x^n \pm 1$, on peut donc s'arranger pour découper suivant une valeur de substitution qui fait correspondre ces deux modulos. Ceci est bien connu, et déjà employé en pratique depuis longtemps. Par contre, au premier niveau, on a plus de liberté sur la façon de découper les entiers en polynômes. Une amélioration que nous avons apportée est de commencer par couper les entiers en deux parties (pas forcément égales), en réduisant d'une part modulo $2^N - 1$ et d'autre part modulo $2^{aN} + 1$, pour un petit entier a. Ces deux nombres sont premiers entre eux, et de plus, le théorème Chinois associé peut s'écrire de manière simple avec uniquement des opérations du type addition, soustraction, décalage. Pour certaines tailles d'entiers à multiplier, cette approche est très payante.

Une astuce supplémentaire a été implantée : dans l'anneau $R = \mathbb{Z}/(2^{2^k}+1)\mathbb{Z}$, l'entier 2 est une racine de l'unité d'ordre 2^{k+1}, mais en fait, $\sqrt{2}$ est aussi dans R et a une expression simple : on a $\sqrt{2} = 2^{3n/4} - 2^{n/4}$, où $n = 2^k$, dans R. Ceci permet d'avoir une transformée deux fois plus longue, pour une même taille de coefficients.

Un gain notable a été obtenu en repensant la procédure de réglage des paramètres en fonction du processeur utilisé. L'idée est que pour une taille d'entiers donnée, il y a plusieurs choix pour découper un entier en polynômes, suivant le choix de l'anneau R. Comme mentionné ci-dessus, on peut aussi découper de manière asymétrique au premier niveau de récursivité, si bien que là encore il y a plusieurs choix. En faisant tourner des tests, on obtient une table des meilleurs choix pour chaque taille. Cette table est spécifique à chaque architecture, car elle dépend grandement des efficacités comparées des opérations de base, ainsi que des caractéristiques de la mémoire et du cache. Un outil est fourni qui permet de recalculer une table adaptée à une machine particulière. Cette approche permet non seulement d'améliorer les temps de calculs en général, mais aussi de lisser quelque peu les «sauts», inévitables dans un contexte de transformée de Fourier.

Pour finir, une attention particulière a été apportée aux questions de localité : l'algorithme de Schönhage-Strassen est adapté aux grands voire aux très grands entiers, qui ne tiennent pas forcément dans les mémoires cache, si bien que l'accès aux données risque d'être largement aussi coûteux que les calculs que l'on fait avec ces données. Une étape de l'algorithme où l'on doit prendre garde à ces questions de localité est lors du calcul d'une transformée de Fourier par FFT. L'algorithme FFT, en écriture récursive, est déjà relativement local : une fois que l'on a atteint un certain niveau de récursion, on travaille toujours sur le même ensemble de données. Le problème est que pour les très grands nombres, ce niveau de récursion auquel la localité apparaît est très profond, et finalement la plupart du calcul s'effectue de manière non-locale, avec une pénalité en conséquence pour les accès mémoire. La littérature est riche de nombreux articles sur les problèmes de localité lors d'une transformée de Fourier rapide. Cependant, le contexte est presque toujours très différent du notre, puisqu'il s'agit de transformée de Fourier complexe, notamment pour des applications en traitement du signal. Ainsi, les scalaires considérés sont petits (au sens de l'espace mémoire occupé), par rapport à la taille des transformations. Dans l'algorithme de Schönhage-Strassen, les scalaires considérés sont gros. On peut même atteindre des cas où le premier niveau de cache peut difficilement en contenir un en entier. Nous avons donc testé plusieurs variantes de la littérature, pour voir celle qui s'adaptait le mieux à notre cas. Au final, l'algorithme «4-step» de Bailey, combiné avec des transformations «radix-4» semble le plus efficace. Nous renvoyons à [72] pour une description de ces algorithmes, dans lesquels on effectue exactement les mêmes opérations que dans un algorithme de FFT classique, mais dans un ordre différent. À d'autres endroits de l'algorithme, il est possible d'améliorer la localité. L'idée sous-jacente est toujours la même : quand on a une donnée sous la main, il faut faire autant de calculs que possible avec celle-ci, avant de la laisser sortir de la mémoire cache.

Le résultat de ces améliorations est une réduction certaine des temps de calculs, notamment pour les grandes tailles de nombres. La figure 4.1 montre un graphique des performances des différentes implantations.

4.2 La bibliothèque $\texttt{mp}\mathbb{F}_q$

La bibliothèque $\texttt{mp}\mathbb{F}_q$ est en cours de développement, en collaboration avec E. Thomé. L'objet de $\texttt{mp}\mathbb{F}_q$ est de fournir une bibliothèque pour manipuler efficacement les corps finis. Il existe déjà un certain nombre de bibliothèques, certaines très complètes, pour manipuler les corps finis ; il est donc nécessaire d'expliquer en quoi celles-ci ne nous convenaient pas pour nos travaux, et les choix que nous avons faits pour répondre aux questions posées.

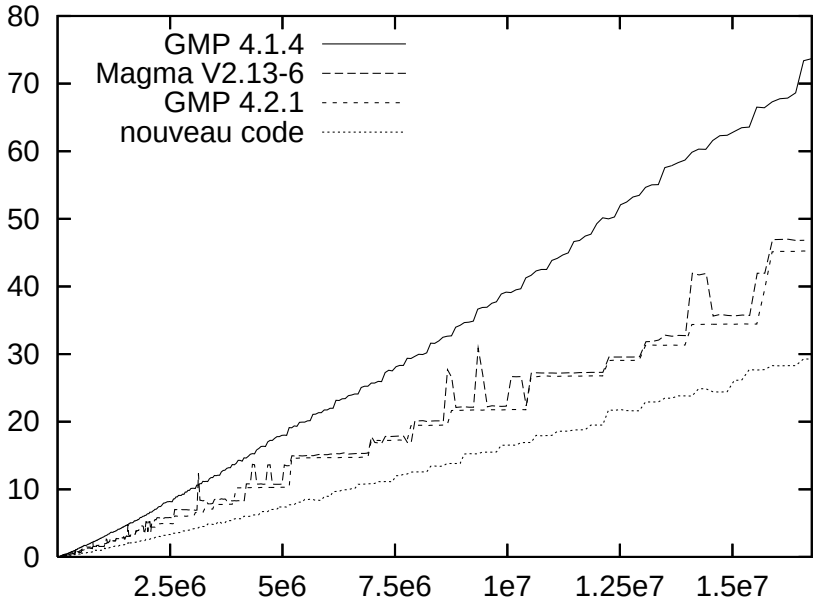


FIG. 4.1 – Comparaison de GMP 4.1.4, GMP 4.2.1, Magma V2.13-6 et le nouveau code pour le multiplication d'entiers sur un Opteron 2.4Ghz (en abscisse, le nombre de mots 64 bits, en ordonnée le temps en secondes).

Pourquoi une nouvelle bibliothèque ? Parmi les bibliothèques existantes qui permettent de calculer dans des corps finis, on peut mentionner NTL, ZEN, Miracl, Givaro.

- NTL est une bibliothèque générale de théorie des nombres, écrite en C++. Elle inclut un module pour calculer dans les corps finis. NTL est extrêmement simple d'utilisation, et fournit de bonnes voire de très bonnes performances.
- ZEN est une bibliothèque écrite en ANSI C, mais qui est orientée objet dans son esprit (de manière similaire à X11) : presque tous les identificateurs accessibles au client sont en fait des macros CPP qui masquent un appel indirect de fonction via un pointeur stocké dans une structure passée en argument. Ceci offre des possibilités très intéressantes, par exemple pour construire des tours d'extensions et y calculer. Seule cette interface haut-niveau est documentée, mais pour une pleine efficacité dans le cas de petits corps finis, il est nécessaire d'appeler les fonctions de bas niveau non documentées.
- Miracl est une bibliothèque orientée vers les applications cryptographiques. Elle contient des fonctionnalités de corps finis, mais aussi les opérations de base sur les courbes elliptiques utiles en cryptographie.
- Givaro est une base C++ qui permet de calculer efficacement dans les petits corps premiers (ceux qui tiennent dans un mot machine).

Lors de l'implantation de certains algorithmes exposés dans ce mémoire, nous nous sommes retrouvés parfois frustrés d'utiliser l'une ou l'autre des bibliothèques ci-dessus. La raison en est que dans bien des cas, on connaît le corps fini dans lequel on va travailler au moment de la compilation. Ceci concerne par exemple l'implantation d'un cryptosystème à base de courbes, où non seulement le corps fini, mais aussi la courbe est connue au moment de la compilation. Un autre exemple est lorsque l'on effectue des expériences de calcul de logarithme discret : le code compilé va tourner pendant des jours, voire des semaines ou des mois, et ceci pour un

seul corps fini (ou une seule courbe sur un corps fini).

Les bibliothèques ci-dessus ne sont pas capables d'utiliser la connaissance du corps fini au moment de la compilation pour améliorer les temps de calcul. Pourtant un certain nombre d'optimisations peuvent être faite dans ce contexte :

- «Code inlining» : il s'agit d'une optimisation très simple à mettre en œuvre (avec les compilateurs modernes), et qui permet d'économiser le surcoût d'appels de fonctions.
- Élimination de branchement : dans les processeurs actuels, les tests coûtent parfois plus cher que les calculs proprement dits. Plus on dispose d'informations au moment de la compilation, moins de tests seront nécessaires à l'exécution.
- Déroulement de boucles : les compilateurs actuels peuvent le faire de manière automatique, mais ceci est plus efficace si la longueur de la boucle est une constante connue à la compilation.
- Choix du meilleur algorithme pour une tâche donnée : ceci peut aussi être fait dynamiquement (ZEN offre cette possibilité), mais c'est plus confortable de le faire à la compilation.

Nous souhaitons donc que `mp`$\mathbb{F}_q$ offre ces possibilités d'optimisation lorsque le corps fini est connu à la compilation. Le critère de succès principal sera donc l'efficacité du code. Cela dit, lors de l'écriture d'une bibliothèque, il est capital de songer à sa vie sur le long terme. La possibilité de maintenir le code est donc un soucis qu'il faut avoir dès le début. Nous avons cependant décidé que cela ne devait pas avoir priorité sur la première préoccupation qui reste l'efficacité.

Le design de `mp`$\mathbb{F}_q$. Les langages choisis pour l'écriture de `mp`$\mathbb{F}_q$ sont `Perl`, C et l'assembleur. Des scripts `Perl` se chargent de générer des sources C ou assembleur adaptées aux différents corps finis que l'on souhaite utiliser, puis ces sources peuvent être incluses lors de la compilation avec l'application. Le langage C++ pourrait fournir un certain niveau de généricité sans compromis sur l'efficacité à l'aide du système de templates. Cela dit, il nous est apparu que le type de manipulations syntaxiques fournies par les templates est aussi bien, sinon mieux, géré par un langage de script indépendant comme `Perl`. L'inconvénient est la perte du typage au niveau de la génération de code, mais c'est le prix à payer pour avoir accès à un langage de programmation très complet lors de cette génération de code.

Pour ce type de bibliothèque, il est clair que l'on peut gagner beaucoup en efficacité en écrivant quelques fonctions fondamentales en assembleur. Nous mentionnons juste que `mp`$\mathbb{F}_q$ autorise l'écriture de telles routines assembleur, et nous ne rentrerons pas plus dans les détails.

L'API (Application Programming Interface – interface de programmation) de `mp`$\mathbb{F}_q$ est uniforme, de manière à avoir des noms de fonctions cohérents d'un corps fini à l'autre. Cette cohérence est assurée par la manière dont les scripts `Perl` génèrent le code. Soit `TAG` une mnémonique décrivant un corps fini ou une famille de corps finis (en fait plusieurs `TAG` peuvent correspondre à un même corps fini, selon la représentation des éléments). Par exemple `2_27` peut correspondre au corps fini $\mathbb{F}_{2^{27}}$ avec une représentation polynomiale des éléments. Alors les types et les fonctions C correspondant à cette mnémonique commenceront par `mpfq_TAG`. Par exemple les éléments de $\mathbb{F}_{2^{27}}$ seront de type `mfpq_TAG_elt` et la fonction de multiplication entre deux tels éléments sera `mpfq_TAG_mul`.

Tous ces types et toutes ces fonctions sont déclarés et définis dans deux fichiers `mpfq_TAG.c` et `mpfq_TAG.h`. Insistons sur le fait qu'il s'agit de véritables identificateurs C, non masqués par des macros. En particulier, les petites fonctions pour lesquelles le coût de l'appel de fonction

risque d'être non négligeable sont définies comme des fonctions `static inline` dans le fichier `.h`, et non comme des macros. Cela aide lors des phases de test, puisque l'on peut désactiver les fonctionnalités «code inlining» du compilateur. Cela permet aussi parfois d'offrir quelques opportunités d'optimisation au compilateur. Par exemple, l'addition de deux éléments de $\mathbb{F}_{2^{89}}$ en représentation polynomiale, sur une machine 64 bits, prendra cette forme :

```
static inline mpfq_2_89_add(mpfq_2_89_field_ptr K,
  mpfq_2_89_dst_elt r, mpfq_2_89_src_elt s1, mpfq_2_89_src_elt s2)
{
    r[0] = s1[0]^s2[0];
    r[1] = s1[1]^s2[1];
}
```

On peut noter que le premier argument est toujours un pointeur vers le corps fini considéré. Dans bien des cas, il s'agit d'un pointeur nul, non utilisé par la fonction, et qui est donc ignoré à la compilation. Mentionnons aussi le fait que le type `mfpq_TAG_elt` se décompose en deux variantes `src` et `dst`, correspondant à l'ajout du mot-clé `const`, en suivant ainsi les pratiques de la bibliothèque GMP qui nous a servi de source d'inspiration.

Quelques benchmarks. La bibliothèque `mp`$\mathbb{F}_q$ en est encore à ses premiers stades de développement. L'API et le système de scripts permettant de générer du code conforme sont essentiellement en place. Il reste encore de nombreux algorithmes à implanter mais on peut d'ores-et-déjà voir que les temps de calcul confirment l'avantage qu'il y a à tirer parti de toute l'information connue à la compilation.

Pour les corps premiers, nous avons des implantations spécifiques pour chaque taille (en nombre de mot-machines) du module. Nous avons aussi optimisé deux corps en particulier : les corps à $2^{127} - 735$ et à $2^{255} - 19$ éléments. La table 4.1, contient les temps de calculs pour ces corps et les comparaisons avec NTL et ZEN sur une machine Opteron.

En ce qui concerne les corps finis de caractéristique 2, nous avons du code généré pour tous les petits degrés. Là encore, nous donnons les temps de calcul sur une machine Opteron, ainsi que la comparaison avec NTL et ZEN (voir figure 4.2).

Développements futurs. Les choix de design et la structure générale étant désormais plus ou moins arrêtés, le gros travail d'ajout de nouveaux algorithmes est commencé. Une première release officielle devrait bientôt avoir lieu. Assurément, les développements seront en partie dictés par les demandes des utilisateurs. Par exemple, il est probable que des fonctionnalités pour les polynômes et l'algèbre seront ajoutés à terme.

TAB. 4.1 – Temps (en nanosecondes, avec 2 chiffres significatifs) pour les opérations de base dans $\mathbb{F}_p$ sur un processeur **AMD Opteron** 250 à 2.40 GHz.

$\texttt{mp}\mathbb{F}_q$:

	1 mot	2 mots	3 mots	4 mots	$2^{127}-735$	$2^{255}-19$
add	2	4	5	7	4	8
sub	2	3	5	5	4	9
sqr	67	108	170	230	14	30
mul	66	109	180	240	16	45
inv	420	2600	4600	7500	2600	7400

NTL :

	1 mot	2 mots	3 mots	4 mots
add	40	42	36	47
sub	38	40	28	44
sqr	120	150	230	290
mul	120	150	230	290
inv	1600	4400	6600	9200

ZEN/ZENmgy :

	1 mot	2 mots	3 mots	4 mots
add	8/11	44/44	44/44	48/49
sub	7/8	64/71	66/70	73/75
sqr	62/90	270/170	420/270	520/320
mul	68/95	300/180	450/270	600/340
inv	1700/2100	3300/4300	4800/5900	6500/7500

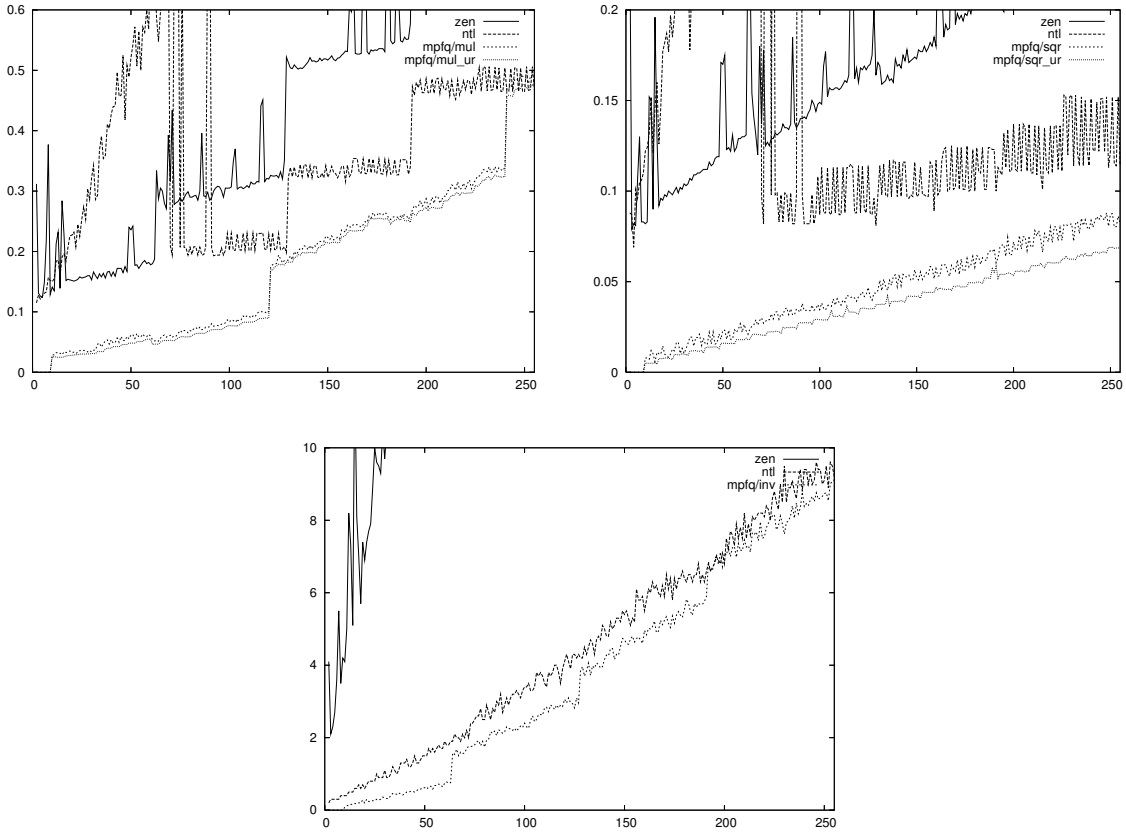


FIG. 4.2 – Temps (en microsecondes) pour la multiplication, la carré et l'inversion dans $\mathbb{F}_{2^n}$ sur un processeur AMD Opteron 2.40 GHz.

Conclusion, perspectives

Bilan des travaux effectués. Depuis une dizaine d'années, le fil conducteur de nos travaux a été l'étude de l'algorithmique des courbes algébriques en vue d'une utilisation cryptographique. Nous avons contribué significativement à l'étude du problème du logarithme discret avec pour conséquence une diminution importante des courbes utilisables, puisque désormais il ne paraît pas raisonnable d'envisager l'utilisation de courbes de genre autre que 1, 2 ou 3, et le genre 3 est aussi en train de tomber.

Nous avons aussi travaillé sur la génération de courbes de genre 1 ou 2 convenables, c'est-à-dire sur le problème du comptage de points. Encore maintenant, notre implantation de l'algorithme de Schoof en genre 2 est la seule à notre connaissance qui permette de générer des cryptosystèmes de genre 2 en grande caractéristique s'appuyant sur des courbes aléatoires. De plus nous avons suivi et contribué aux approches p-adiques de Kedlaya et de Satoh, qui permettent désormais de compter les points en petite caractéristique de manière très efficace.

Pour finir, nous nous sommes attaqué au problème de l'efficacité des systèmes de genre 2, en proposant des formules pour la loi de groupe qui sont bien meilleures que celles connues auparavant. Nous ne nous sommes pas arrêtés là et avons mené des travaux de développement de bibliothèques, afin d'obtenir des records de temps de calcul pour l'opération cryptographique d'échange de clef.

Tout ceci se retrouve dans les logiciels `surf2113` et `surf127eps` que nous avons publiés pour la compétition européenne eBATS[2]. En effet, le fait de se restreindre aux courbes de genre 2 provient des progrès sur le logarithme discret ; les algorithmes modernes de comptage de points ont permis de s'assurer que l'ordre du groupe est presque premier ; les lois de groupe utilisées sont celles que j'ai développées ; et la bibliothèque de corps finis utilisée est `mp`$\mathbb{F}_q$. Cela résume bien le champ d'application de nos travaux.

Perspectives. En ce qui concerne le logarithme discret, la prochaine avancée à attendre est la réduction complète d'un problème hyperelliptique de genre 3 à un problème non-hyperelliptique de genre 3, suivant les travaux de Smith. Le point de blocage se situe actuellement dans le calcul d'isogénies effectives entre jacobiennes. À part cet aspect, nous n'attendons pas d'avancée majeure sur le problème du logarithme discret, sauf si une idée vraiment nouvelle surgit.

Pour le problème du comptage de points, les vrais problèmes se situent en grande caractéristique, car les extensions de l'algorithme de Schoof, bien que de complexité polynomiale à genre fixé, se comportent assez mal en pratique. En genre 2, pour aller plus loin que les 128 bits de sécurité que nous avons récemment obtenus, les améliorations d'Atkin et Elkies doivent être étendues, mais là encore le calcul effectif d'isogénies entre jacobiennes est un

[2] `http://www.ecrypt.eu.org/ebats/`

point de blocage.

Du côté des formules pour les lois de groupe, le sujet est encore en pleine activité, puisque même le cas elliptique a évolué récemment, avec l'introduction des coordonnées d'Edwards. Le genre 2 est donc très loin d'avoir révélé tous ses secrets, et l'on peut s'attendre à des améliorations dans un futur relativement proche.

Ainsi, je compte continuer à travailler sur l'algorithmique des courbes, principalement dans la direction des formules efficaces, notamment en m'appuyant sur la théorie des fonctions Thêta. Un doctorant de l'équipe est actuellement en train de s'attaquer aux questions de calcul d'isogénies effectives, ce qui permet de garder une activité sur les autres aspects.

Suite à ma mutation au LORIA, dans l'équipe CACAO, ma recherche s'est peu à peu orientée vers des aspects arithmétiques. Cela s'est traduit principalement par les aspects du chapitre 4 de ce mémoire, ainsi que par ma participation, en tant que coordinateur, au projet ANR CADO sur le thème de la factorisation d'entiers par la méthode du crible algébrique.

Ma vision du domaine de l'arithmétique efficace n'est pas encore aussi précise que pour les courbes algébriques en cryptographie, mais j'ai quelques projets en cours et à venir. Le développement de la bibliothèque $\texttt{mp}\mathbb{F}_q$ en fait bien entendu partie. Par ailleurs, l'arithmétique asymptotiquement rapide, et notamment les algorithmes à base de transformée de Fourier rapide continuent à me fasciner. À travers le projet CADO, je me familiarise avec la problématique de la factorisation des grands entiers, qui recèle, elle aussi, encore bien des secrets excitants.

Bibliographie

[1] L. Adleman and M.-D. Huang. Counting points on curves and abelian varieties over finite fields. *J. Symbolic Comput.*, 32 :171–189, 2001.

[2] L. M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. In L. Adleman and M.-D. Huang, editors, *ANTS-I*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 28–40. Springer–Verlag, 1994.

[3] S. Arita, K. Matsuo, K. Nagao, and M. Shimura. A Weil descent attack against elliptic curve cryptosystems over quartic extension fields. Cryptology ePrint Archive : Report 2004/240.

[4] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61(203) :29–68, July 1993.

[5] R. Avanzi, N. Thériault, and Z. Wang. Rethinking low genus hyperelliptic jacobian arithmetic over binary fields : Interplay of field arithmetic and explicit formulae. Preprint.

[6] M. Bauer, E. Teske, and A. Weng. Point counting on Picard curves in large characteristic. *Math. Comp.*, 74 :1983–2005, 2005.

[7] Ch. Birkenhake and H. Lange. *Complex abelian varieties*, volume 302 of *Grundlehren der mathematischen Wissenschaften*. Springer–Verlag, 1992.

[8] W. Bosma and J. Cannon. *Handbook of Magma functions*, 1997. `http://www.maths.usyd.edu.au:8000/u/magma/`.

[9] J.-B. Bost and J.-F. Mestre. Moyenne arithmético-géométrique et périodes de courbes de genre 1 et 2. *Gaz. Math. Soc. France*, 38 :36–64, 1988.

[10] A. Bostan, P. Gaudry, and É. Schost. Linear recurrences with polynomial coefficients and computation of the Cartier-Manin operator on hyperelliptic curves. In G. Mullen, A. Poli, and H. Stichtenoth, editors, *Finite Fields and Applications, 7th International Conference, Fq7*, volume 2948 of *Lecture Notes in Comput. Sci.*, pages 40–58. Springer-Verlag, 2004.

[11] A. Bostan, P. Gaudry, and É. Schost. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM J. Comput.*, 36 :1777–1806, 2007.

[12] A. Bostan, F. Morain, B. Salvy, and É. Schost. Fast algorithms for computing isogenies between elliptic curves. To appear in Math. Comp.

[13] R. Bröker and K. Lauter. Modular polynomials for genus 2. Cryptology ePrint Archive : Report 2008/161.

[14] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48(177) :95–101, 1987.

[15] D. G. Cantor. On the analogue of the division polynomials for hyperelliptic curves. *J. Reine Angew. Math.*, 447 :91–145, 1994.

[16] R. Carls, D. Kohel, and D. Lubicz. Higher dimensional 3-adic CM construction. *J. Algebra*, 319 :971–1006, 2008.

[17] R. Carls and D. Lubicz. A p-adic quasi-quadratic point counting algorithm. Preprint 2007.

[18] P. Cartier. Une nouvelle opération sur les formes différentielles. *C. R. Acad. Sci. Paris Sér. I Math.*, 244 :426–428, 1957.

[19] W. Castryck, J. Denef, and F. Vercauteren. Computing zeta functions of nondegenerate curves. *International Mathematical Research Papers*, 2006 :Article ID 72017, 57 pages, 2006.

[20] W. Castryck, H. Hubrechts, and F. Vercauteren. Computing zeta functions in families of C_{ab} curves using deformation. In A. van der Poorten and A. Stein, editors, *ANTS-VIII*, volume 5011 of *Lecture Notes in Comput. Sci.*, pages 296–311. Springer-Verlag, 2008.

[21] A. Chambert-Loir. Compter (rapidement) le nombre de solutions d'équations dans les corps finis. *Séminaire Bourbaki*, 968, 2006.

[22] L. S. Charlap, R. Coley, and D. P. Robbins. Enumeration of rational points on elliptic curves over finite fields. Draft, 1991.

[23] D. V. Chudnovsky and G. V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Adv. in Appl. Math.*, 7 :385–434, 1986.

[24] D. V. Chudnovsky and G. V. Chudnovsky. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited (Urbana-Champaign, Ill., 1987)*, pages 375–472. Academic Press, Boston, MA, 1988.

[25] H. Cohen and G. Frey, editors. *Handbook of elliptic and hyperelliptic curve cryptography.* Chapman & Hall / CRC, 2005.

[26] J.-M. Couveignes. Computing l-isogenies using the p-torsion. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 59–65. Springer Verlag, 1996. Second International Symposium, ANTS-II, Talence, France, May 1996, Proceedings.

[27] J.-M. Couveignes. Algebraic groups and discrete logarithm. In *Public-key cryptography and computational number theory*, pages 17–27. de Gruyter, 2001.

[28] J. Denef and F. Vercauteren. Counting points on C_{ab} curves using Monsky-Washnitzer cohomology. *Finite Fields and Their Applications*, 12 :78–102, 2006.

[29] J. Denef and F. Vercauteren. An extension of Kedlaya's algorithm to hyperelliptic curves in characteristic 2. *J. of Cryptology*, 19 :1–25, 2006.

[30] C. Diem. An index calculus algorithm for non-singular plane curves of high genus. Talk given at the ECC conference, 2006.

[31] C. Diem. The GHS-attack in odd characteristic. *J. Ramanujan Math. Soc.*, 18 :1–32, 2003.

[32] C. Diem. An index calculus algorithm for plane curves of small degree. In S. Pauli F. Hess and M. Pohst, editors, *ANTS-VII*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 543–557. Springer–Verlag, 2006.

[33] C. Diem and J. Scholten. Cover attacks : a report for the AREHCC project, 2003.

[34] C. Diem and E. Thomé. Index calculus in class groups of non-hyperelliptic curves of genus three, 2007. To appear.

[35] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22-6 :644–654, November 1976.

[36] R. Dupont. *Moyenne arithmético-géométrique, suites de Borchardt et applications*. PhD thesis, École polytechnique, 2006.

[37] S. Duquesne. Montgomery scalar multiplication for genus 2 curves. In D. Buell, editor, *ANTS-VI*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 153–168. Springer–Verlag, 2004.

[38] K. Eisentraeger and K. Lauter. A CRT algorithm for constructing genus 2 curves over finite fields. To appear in Proceedings of AGCT 2005.

[39] A. Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Math. Comp.*, 71 :729–742, 2002.

[40] A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith.*, 102 :83–103, 2002.

[41] A. Enge and P. Gaudry. An $L(1/3+\varepsilon)$ algorithm for the discrete logarithm problem for low degree curves. In M. Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Comput. Sci.*, pages 379–393. Springer-Verlag, 2007.

[42] A. Enge and F. Morain. SEA in genus 1 : 2500 decimal digits. `http://listserv.nodak.edu/archives/nmbrthry.html`, December 2006.

[43] A. Enge and A. Stein. Smooth ideals in hyperelliptic function fields. *Math. Comp.*, 71 :1219–1230, 2002.

[44] X. Fan and G. Gong. Efficient explicit formulae for genus 2 hyperelliptic curves over prime fields and their implementations. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Comput. Sci.*, pages 155–172. Springer-Verlag, 2007.

[45] X. Fan, T. Wollinger, and Y. Wang. Inversion-free arithmetic on genus 3 hyperelliptic curves and its implementations. In *ITCC*, pages 642–647. IEEE Computer Society, 2005.

[46] X. Fan, T. Wollinger, and Y. Wang. Efficient doubling on genus 3 curves over binary fields. In D. Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Comput. Sci.*, pages 64–81. Springer-Verlag, 2006.

[47] P. Flajolet and B. Salvy. The SIGSAM challenges : Symbolic asymptotics in practice. *SIGSAM Bull.*, 31(4) :36–47, 1997.

[48] R. Flassenberg and S. Paulus. Sieving in function fields. *Experiment. Math.*, 8(4) :339–349, 1999.

[49] M. Fouquet, P. Gaudry, and R. Harley. An extension of Satoh's algorithm and its implementation. *J. Ramanujan Math. Soc.*, 15 :281–318, 2000.

[50] M. Fouquet, P. Gaudry, and R. Harley. Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 14–29. Springer-Verlag, 2001.

[51] D. Freeman. A generalized Brezing-Weng method for constructing pairing-friendly ordinary abelian varieties. To appear in Pairing 2008.

[52] D. Freeman. Constructing pairing-friendly genus 2 curves with ordinary Jacobians. In T. Takagi et al., editor, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Comput. Sci.*, pages 152–176. Springer-Verlag, 2007.

[53] D. Freeman, P. Stevenhagen, and M. Streng. Abelian varieties with prescribed embedding degree. In A. van der Poorten and A. Stein, editors, *ANTS-VIII*, volume 5011 of *Lecture Notes in Comput. Sci.*, pages 60–73. Springer-Verlag, 2008.

[54] G. Frey. How to disguise an elliptic curve (Weil descent). Talk at Waterloo workshop ECC'98, 1998. `http://cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html`.

[55] W. Fulton. *Algebraic curves*. Math. Lec. Note Series. W. A. Benjamin Inc, 1969.

[56] M. Fürer. Faster integer multiplication. In Uriel Feige, editor, *STOC '07*, pages 57–66. ACM, 2007.

[57] S. Galbraith. Weil descent of Jacobians. *Discrete Appl. Math.*, 128 :165–180, 2003.

[58] S. Galbraith, F. Heß, and N. Smart. Extending the GHS Weil descent attack. In L. R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Comput. Sci.*, pages 29–44. Springer-Verlag, 2002.

[59] S. Galbraith and N. Smart. A cryptographic application of Weil descent. In *Cryptography and Coding, 7th IMA Conference*, volume 1746 of *Lecture Notes in Comput. Sci.*, pages 191–200. springer–Verlag, 1999. Full paper is HP-LABS Technical Report (Number HPL-1999-70).

[60] Steven D. Galbraith, Sachar Paulus, and Nigel P. Smart. Arithmetic on superelliptic curves. *Mathematics of Computation*, 71(237) :393–405, 2002.

[61] P. Gaudry. Algorithmes de comptage de points d'une courbe définie sur un corps fini. Preprint, 28 pages, 2006. Soumis pour un numéro spécial d'Astérisque, en lien avec le trimestre «Méthodes explicites en théorie des nombres», à l'IHP en automne 2004.

[62] P. Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Cryptology ePrint Archive : Report 2004/073. Accepted for publication in *J. Symbolic Comput.*

[63] P. Gaudry. *NTLJac2, Tools for genus 2 Jacobians in NTL.* `http://www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/NTLJac2/`.

[64] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Comput. Sci.*, pages 19–34. Springer–Verlag, 2000.

[65] P. Gaudry. A comparison and a combination of SST and AGM algorithms for counting points of elliptic curves in characteristic 2. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Comput. Sci.*, pages 311–327. Springer–Verlag, 2002.

[66] P. Gaudry. Fast genus 2 arithmetic based on Theta functions. *J. of Mathematical Cryptology*, 1 :243–265, 2007.

[67] P. Gaudry and N. Gürel. An extension of Kedlaya's algorithm to superelliptic curves. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Comput. Sci.*, pages 480–494. Springer-Verlag, 2001.

[68] P. Gaudry and N. Gürel. Counting points in medium characteristic using Kedlaya's algorithm. *Experiment. Math.*, 12 :395–402, 2003.

[69] P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. In W. Bosma, editor, *ANTS-IV*, volume 1838 of *Lecture Notes in Comput. Sci.*, pages 313–332. Springer–Verlag, 2000.

[70] P. Gaudry, F. Hess, and N. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. of Cryptology*, 15 :19–46, 2002.

[71] P. Gaudry, T. Houtmann, D. Kohel, C. Ritzenthaler, and A. Weng. The 2-adic CM method for genus 2 curves with application to cryptography. In X. Lai and K. Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Comput. Sci.*, pages 114–129. Springer-Verlag, 2006.

[72] P. Gaudry, A. Kruppa, and P. Zimmermann. A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm. In C. Brown, editor, *ISSAC 2007*, pages 167–174. ACM, 2007.

[73] P. Gaudry and D. Lubicz. The arithmetic of characteristic 2 Kummer surfaces. Preprint, 18 pages, 2008.

[74] P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Comput. Sci.*, pages 239–256. Springer-Verlag, 2004.

[75] P. Gaudry and E. Schost. A low memory parallel version of Matsuo, Chao and Tsujii's algorithm. In D. Buell, editor, *ANTS-VI*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 208–222. Springer-Verlag, 2004.

[76] P. Gaudry and É. Schost. Modular equations for hyperelliptic curves. *Math. Comp.*, 74 :429–454, 2005.

[77] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comp.*, 76 :475–492, 2007.

[78] P. Gaudry and E. Thomé. The mpFq library and implementing curve-based key exchanges. In *SPEED : Software Performance Enhancement for Encryption and Decryption*, pages 49–64, 2007.

[79] R. Gerkmann. Relative rigid cohomology and point counting on families of elliptic curves. To appear in Journal of the Ramanujan Mathematical Society.

[80] E. Goren and K. Lauter. Class invariants of quartic CM fields. *Annales de l'Institut Fourier*, 57 :457–480, 2007.

[81] R. Granger and F. Vercauteren. On the discrete logarithm problem on algebraic tori. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Comput. Sci.*, pages 66–85. Springer–Verlag, 2005.

[82] R. Harley. Asymptotically optimal p-adic point-counting. e-mail to the NMBRTHRY list, December 2002.

[83] R. Harley. Cardinality of a genus 2 hyperelliptic curve over $GF(2^{4001})$. e-mail to the NMBRTHRY list, September 2002.

[84] D. Harvey. Kedlaya's algorithm in larger characteristic. Preprint 2007.

[85] H. Hasse and E. Witt. Zyklische unverzweigte Erweiterungskörper vom primzahlgrade p über einem algebraischen Funktionenkörper der Charakteristik p. *Monatsch. Math. Phys.*, 43 :477–492, 1936.

[86] F. Hess. The GHS attack revisited. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Comput. Sci.*, pages 374–387. Springer-Verlag, 2003.

[87] F. Heß. Computing relations in divisor class groups of algebraic curves over finite fields. Preprint, 2004.

[88] F. Hess. Generalising the GHS attack on the elliptic curve discrete logarithm. *LMS J. Comput. Math.*, 7 :167–192, 2004.

[89] M.-D. Huang and D. Ierardi. Counting points on curves over finite fields. *J. Symbolic Comput.*, 25 :1–21, 1998.

[90] H. Hubrechts. Point counting in families of hyperelliptic curves. To appear in Foundations of Computational Mathematics.

[91] H. Hubrechts. Point counting in families of hyperelliptic curves in characteristic 2. To appear in LMS Journal of Computation and Mathematics.

[92] H. Hubrechts. Quasi-quadratic elliptic curve point counting using rigid cohomology. To appear in MEGA 2007.

[93] T. Iijima, M. Shimura, J. Chao, and S. Tsujii. An extension of GHS Weil descent attack. *IEICE Transactions*, 88-A(1) :97–104, 2005.

[94] M. Jacobson, N. Koblitz, J. Silverman, A. Stein, and E. Teske. Analysis of the Xedni calculus attack. *Des. Codes Cryptogr.*, 20 :41–64, 2000.

[95] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *J. of Cryptology*, 16 :239–247, 2003.

[96] M. Kawazoe and T. Takahashi. Pairing-friendly hyperelliptic curves with ordinary jacobians of type $y^2 = x^5 + ax$. Cryptology ePrint Archive : Report 2008/026.

[97] Kiran S. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *J. Ramanujan Math. Soc.*, 16(4) :323–338, 2001.

[98] H. Kim, J. Park, J. Cheon, J. Park, J. Kim, and S. Hahn. Fast elliptic curve point counting using Gaussian normal basis. In C. Fieker and D. R. Kohel, editors, *ANTS-V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 292–307. Springer–Verlag, 2002.

[99] D. Kohel. The AGM-$X_0(N)$ Heegner point lifting algorithm and elliptic curve point counting. In C. Laih, editor, *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Comput. Sci.*, pages 124–136. Springer-Verlag, 2003.

[100] D. Kohel. Complex multiplication and canonical lifts. In R. Rolland J. Chaumine, J. Hirschfeld, editor, *Algebraic geometry and its applications – Proceedings of the first SAGA conference*, pages 67–83. Woeld Scientific, 2008.

[101] V. Kovtun and T. Wollinger. Fast explicit formulae for genus 2 hyperelliptic curves using projective coordinates. Cryptology ePrint Archive : Report 2008/056.

[102] T. Lange. Montgomery addition for genus two curves. In D. Buell, editor, *ANTS-VI*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 309–317. Springer–Verlag, 2004.

[103] A. Lauder. Rigid cohomology and p-adic point counting. To appear in Journal de Théorie des Nombres de Bordeaux.

[104] A. Lauder. Computing zeta functions of Kummer curves via multiplicative characters. *Found. Comput. Math.*, 3 :273–295, 2003.

[105] A. Lauder. Deformation theory and the computation of zeta functions. *Proc. London Math. Soc.*, 88 :565–602, 2004.

[106] A. Lauder. A recursive method for computing zeta functions of varieties. *LMS Journal of Computation and Mathematics*, 9 :222–267, 2006.

[107] A. Lauder and D. Wan. Counting points on varieties over finite fields of small characteristic,. Preprint 2001.

[108] A. Lauder and D. Wan. Computing zeta functions of Artin-Schreier curves over finite fields. *LMS Journal of Computation and Mathematics*, 5 :34–55, 2002.

[109] A. Lauder and D. Wan. Computing zeta functions of Artin-Schreier curves over finite fields II. *J. Complexity*, 20 :331–349, 2004.

[110] R. Lercier. Computing isogenies in F_{2^n}. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 197–212. Springer Verlag, 1996.

[111] R. Lercier and D. Lubicz. A quasi quadratic time algorithm for hyperelliptic curve point counting. To appear in J. Ramanujan Math. Soc.

[112] R. Lercier and F. Morain. Computing isogenies between elliptic curves over F_{p^n} using Couveignes's algorithm. *Math. Comp.*, 69(229) :351–370, January 2000.

[113] D. Lorenzini. *An invitation to arithmetic geometry*, volume 9 of *Graduate Studies in Mathematics*. AMS, 1996.

[114] J. Lubin, J. P. Serre, and J. Tate. Elliptic curves and formal groups. In *Lecture notes prepared in connection with the seminars held at the Summer Institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massach usetts, July 6-July 31, 1964*, 1964. Scanned copies available at `http://www.ma.utexas.edu/users/voloch/lst.html`.

[115] M. Madsen. A p-adic point counting algorithm for elliptic curves on Legendre form. *Finite Fields and Their Applications*, 11 :71–88, 2005.

[116] J. I. Manin. The Hasse-Witt matrix of an algebraic curve. *Trans. Amer. Math. Soc.*, 45 :245–264, 1965.

[117] K. Matsuo, J. Chao, and S. Tsujii. An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In C. Fiecker and D. Kohel, editors, *ANTS-V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 461–474. Springer-Verlag, 2002.

[118] M. Maurer, A. Menezes, and E. Teske. Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree. *LMS Journal of Computation and Mathematics*, 5 :127–174, 2002.

[119] U. M. Maurer and S. Wolf. Diffie-Hellman oracles. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Comput. Sci.*, pages 268–282. Springer-Verlag, 1996.

[120] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39(5) :1639–1646, September 1993.

[121] A. Menezes and M. Qu. Analysis of the Weil descent attack of Gaudry, Hess and Smart. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *LNCS*, pages 308–318. Springer-Verlag, 2001.

[122] A. Menezes and E. Teske. Cryptographic implications of Hess' generalized GHS attack. *Appl. Algebra Engrg. Comm. Comput.*, 16 :439–460, 2006.

[123] A. Menezes, E. Teske, and A. Weng. Weak fields for ECC. In T. Okamoto, editor, *Topics in Cryptology - CT-RSA 2004*, volume 2964 of *LNCS*, pages 366–386. Springer-Verlag, 2004.

[124] A. Menezes, Y.-H. Wu, and R. Zuccherato. An elementary introduction to hyperelliptic curves. In *Algebraic aspects of cryptography*, by N. Koblitz, pages 155–178, Springer-Verlag, 1997.

[125] J.-F. Mestre. Algorithmes pour compter des points de courbes en petite caractéristique et en petit genre. Exposé donné à Rennes en mars 2002. Notes rédigées par D. Lubicz.

[126] J.-F. Mestre. Utilisation de l'AGM pour le calcul de $E(F_{2^n})$. Lettre adressée à Gaudry et Harley, Décembre 2000.

[127] J.-F. Mestre. Construction de courbes de genre 2 à partir de leurs modules. In T. Mora and C. Traverso, editors, *Effective methods in algebraic geometry*, volume 94 of *Progr. Math.*, pages 313–334. Birkhäuser, 1991. Proc. Congress in Livorno, Italy, April 17–21, 1990.

[128] V. Miller. Short programs for functions on curves. Draft, 1986.

[129] V. Miller. The Weil pairing and its efficient calculation. *J. of Cryptology*, 17 :235–261, 2004.

[130] F. Momose and J. Chao. Classification of Weil restrictions obtained by $(2, ..., 2)$ coverings of P^1. Cryptology ePrint Archive : Report 2006/347.

[131] F. Momose and J. Chao. Scholten forms and elliptic/hyperelliptic curves with weak Weil restrictions. Cryptology ePrint Archive : Report 2005/277.

[132] V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Math. Comp.*, 68(226) :807–822, 1999.

[133] A. Muzereau, N. Smart, and F. Vercauteren. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS J. Comput. Math*, 7 :50–72, 2004.

[134] K. Nagao. Decomposed attack for the jacobian of a hyperelliptic curve over an extension field. Cryptology ePrint Archive : Report 2007/112.

[135] F. Oort. Lifting algebraic curves, abelian varieties, and their endomorphisms to characteristic zero. In *Algebraic geometry, Bowdoin, 1985 (Brunswick, Maine, 1985)*, volume 46 of *Proc. Sympos. Pure Math.*, pages 165–195. AMS, 1987.

[136] J. Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comp.*, 55(192) :745–763, October 1990.

[137] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT–24 :106–110, 1978.

[138] F. Richelot. Essai sur une méthode générale pour déterminer les valeurs des intégrales ultra-elliptiques, fondée sur des transformations remarquables de ces transcendantes. *C. R. Acad. Sci. Paris*, 2 :622–627, 1836.

[139] C. Ritzenthaler. Point counting on genus 3 non hyperelliptic curves. In D. Buell, editor, *ANTS-VI*, volume 3076 of *Lecture Notes in Comput. Sci.*, pages 379–394. Springer–Verlag, 2004.

[140] H. G. Rück. On the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 68(226) :805–806, 1999.

[141] T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, 15 :247–270, 2000.

[142] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comment. Math. Helv.*, 47(1) :81–92, 1998.

[143] T. Satoh, B. Skjernaa, and Y. Taguchi. Fast computation of canonical lifts of elliptic curves and its application to point counting. *Finite Fields and Their Applications*, 9 :89–101, 2003.

[144] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7 :281–292, 1971.

[145] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44 :483–494, 1985.

[146] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7 :219–254, 1995.

[147] I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Preprint, 2004.

[148] I. A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p. *Math. Comp.*, 67(221) :353–356, January 1998.

[149] J.-P. Serre. *Corps locaux*. Hermann, 1968.

[150] D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math. vol. 20*, pages 415–440. AMS, 1971.

[151] V. Shoup. *NTL : A library for doing number theory*. Distributed at `http://www.shoup.net/ntl/`.

[152] J. Silverman. The Xedni calculus and the elliptic curve discrete logarithm problem. *Des. Codes Cryptogr.*, 20 :5–40, 2000.

[153] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer–Verlag, 1986.

[154] Berit Skjernaa. Satoh's algorithm in characteristic 2. *Math. Comp.*, 72 :477–487, 2003.

[155] N. Smart. On the performance of hyperelliptic cryptosystems. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Comput. Sci.*, pages 165–175. Springer–Verlag, 1999.

[156] N. Smart and S. Siksek. A fast Diffie-Hellman protocol in genus 2. *J. of Cryptology*, 12 :67–73, 1999.

[157] B. Smith. Isogenies and the discrete logarithm problem in jacobians of genus 3 hyperelliptic curves. In N. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Comput. Sci.*, pages 163–180. Springer-Verlag, 2008.

[158] A.-M. Spallek. *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen*. PhD thesis, Universität Gesamthochschule Essen, July 1994.

[159] M. Stam. On Montgomery-like representations for elliptic curves over $GF(2^k)$. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *Lecture Notes in Comput. Sci.*, pages 240–254. Springer-Verlag, 2003.

[160] A. Stein and H. Williams. Some methods for evaluating the regulator of a real quadratic function field. *Experiment. Math.*, 8(2) :119–133, 1999.

[161] H. Stichtenoth. *Algebraic function fields and codes*. Springer–Verlag, 1993.

[162] V. Strassen. Einige Resultate über Berechnungskomplexität. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 78 :1–8, 1976.

[163] A. Sutherland. A generic approach to searching for Jacobians. To appear in Math. Comp.

[164] E. Teske. An elliptic curve trapdoor system. *J. of Cryptology*, 19 :115–133, 2006.

[165] N. Thériault. Weil descent attack for Artin-Schreier curves. Preprint.

[166] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In C. Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Comput. Sci.*, pages 75–92. Springer-Verlag, 2003.

[167] N. Thériault. Weil descent attack for Kummer extensions. *J. Ramanujan Math. Soc.*, 18 :281–312, 2003.

[168] P. van Wamelen. Examples of genus two CM curves defined over the rationals. *Math. Comp.*, 68(225) :307–320, January 1999.

[169] F. Vercauteren. *Computing zeta functions of curves over finite fields*. PhD thesis, Katholieke Universiteit Leuven, 2003.

[170] F. Vercauteren, B. Preneel, and J. Vandewalle. A memory efficient version of Satoh's algorithm. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 1–13. Springer-Verlag, 2001.

[171] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

[172] A. Weng. Constructing hyperelliptic curves of genus 2 suitable for cryptography. *Math. Comp.*, 72 :435–458, 2003.

[173] A. Weng. Extensions and improvements for the CM method for genus two. *Fields Institute Comm.*, 41 :379–389, 2004.

[174] A. Weng. A low-memory algorithm for point counting on Picard curves. *Des. Codes Cryptogr.*, 38 :383–393, 2006.

[175] T. Wollinger, J. Pelzl, and C. Paar. Cantor versus Harley : Optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems. *IEEE Trans. Computers*, 54(7) :861–872, 2005.

[176] N. Yui. On the jacobian varietes of hyperelliptic curves over fields of characteristic $p > 2$. *J. Algebra*, 52 :378–410, 1978.

Résumé

Dans ce mémoire, nous présentons divers travaux sur le thème de l'algorithmique des courbes algébriques en vue d'applications à la cryptologie. Nous décrivons des algorithmes pour le calcul de logarithmes discrets, problème dont la difficulté est à la base de la sécurité des cryptosystèmes s'appuyant sur les courbes. Une première classe d'algorithmes regroupe les techniques du type «calcul d'index» ; une seconde les méthodes liées à la restriction de Weil.

Viennent ensuite des algorithmes permettant le calcul du nombre de points d'une courbe définie sur un corps fini. Ceux-ci se répartissent en trois catégories : l'algorithme de Schoof et ses généralisations, les algorithmes p-adiques s'appuyant sur un relèvement canonique, et les méthodes p-adiques issues de l'algorithme de Kedlaya.

Nous traitons d'autres aspects pouvant être utiles lors de la conception de cryptosystèmes à bases de courbes, en particulier des formules efficaces pour la loi de groupe en genre 2, issues de la théorie des fonctions Thêta.

Pour finir, nous mentionnons des travaux liés à l'arithmétique efficace et son implantation logicielle, notamment des travaux sur l'algorithme de Schönhage-Strassen et sur une bibliothèque pour les corps finis.

Abstract

In this memoir, we present various works on the theme of algorithms for algebraic curves with a view towards cryptology. We describe algorithms for computing discrete logarithms, which is a problem whose difficulty is the key to the security of curve-based cryptosystems. A first class of algorithms contains «index-calculus»-like techniques ; a second class contains methods relying on the Weil restriction.

Algorithms for counting points of a curve defined over a finite field come next. They can be sorted in three categories : Schoof's algorithm and its generalizations, p-adic algorithms based on a canonical lift, and p-adic methods following Kedlaya's algorithm.

We deal with other aspects that can be useful when designing a curve-based cryptosystem, in particular some efficient formulae for the group law in genus 2, based on the theory of Theta functions.

Finally, we mention works related to efficient arithmetic and its implementation, namely a work on the algorithm of Schönhage-Strassen and a work on a finite field library.