ALGORITHMES DE COMPTAGE DE POINTS D'UNE COURBE DÉFINIE SUR UN CORPS FINI

par

Pierrick Gaudry

1. Introduction : énoncé du problème

Soit $q = p^n$ un puissance d'un nombre premier p et soit $\mathbb{F}_q$ le corps fini à q éléments. Soit $\mathcal{C}$ une courbe algébrique projective lisse de genre g définie sur le corps $\mathbb{F}_q$. La fonction Zêta de $\mathcal{C}$ est définie par

$$Z(t) = \exp\left(\sum_{k\geqslant 1} N_k \frac{t^k}{k}\right),$$

où N_k est le nombre de points de $\mathcal{C}$ définis sur $\mathbb{F}_{q^k}$, que l'on notera aussi $\#\mathcal{C}(\mathbb{F}_{q^k})$. Le théorème de Weil (dont on peut trouver une preuve dans [**64**] par exemple) affirme que $Z(t)$ est en fait une fraction rationnelle, ce qui signifie entre autres que connaître les premiers termes de la série suffit pour calculer tous les autres termes. Plus précisément, $Z(t)$ s'écrit

$$Z(t) = \frac{L(t)}{(1-t)(1-qt)},$$

où $L(t) = a_0 + a_1 t + \cdots + a_{2g} t^{2g}$ est un polynôme de degré $2g$ dont les coefficients sont des entiers vérifiant

$$a_0 = 1, a_{2g} = q^g, \text{ et } a_{2g-i} = q^{g-i} a_i, \text{ pour } 0 \leqslant i \leqslant g.$$

De plus, les inverses des racines de $L(t)$ sont de module $\sqrt{q}$ (c'est l'équivalent de l'hypothèse de Riemann pour les corps de fonctions), ce qui donne la borne $|a_i| \leqslant \binom{2g}{i} q^{i/2}$, pour les coefficients de $L(t)$.

La jacobienne de la courbe $\mathcal{C}$ que l'on notera $\mathrm{Jac}(\mathcal{C})$ est une variété abélienne dont les propriétés arithmétiques sont liées à celle de $\mathcal{C}$. En particulier, le numérateur $L(t)$ de $Z(t)$ est le polynôme réciproque du polynôme caractéristique $\chi_\pi(t)$ de l'endomorphisme de Frobenius $\pi : x \mapsto x^q$ agissant sur $\mathrm{Jac}(\mathcal{C})$. Par ailleurs, le nombre de points $\mathbb{F}_q$ rationnels de $\mathrm{Jac}(\mathcal{C})$ est donné par $\chi_\pi(1)$, si bien que l'ordre du groupe des points rationnels de $\mathrm{Jac}(\mathcal{C})$ se déduit de $Z(t)$. La réciproque est vraie dans le cas du genre 1 ou 2, mais la connaissance de $\chi_\pi(1)$ ne suffit plus à déterminer facilement $Z(t)$ lorsque g est supérieur ou égal à 3.

Nous nous intéressons ici au problème de calculer $Z(t)$ ou éventuellement seulement $\chi_\pi(1)$. Ce type de calcul intervient pour les applications suivantes : en théorie des codes correcteurs d'erreurs, pour un genre fixé, le nombre de points sur une courbe indique la qualité du code que l'on peut créer à partir de celle-ci ; en cryptographie, connaître l'ordre du groupe des points de la jacobienne est nécessaire car si celui-ci est friable, le problème du logarithme discret est facile, et donc aucune sécurité n'est à espérer. Et bien sûr, si l'on est capable de calculer efficacement les invariants associés à une courbe, cela ne fait qu'enrichir les logiciels de calcul formel qui sont devenus des calculettes indispensables à bon nombre de mathématiciens.

Il existe des algorithmes qui permettent de calculer $Z(t)$: puisque tous les objets considérés sont finis, il suffit d'énumérer. Évidemment cela ne suffit pas pour traiter des exemples de grande taille. Dans nos mesures de complexité des algorithmes, la taille que nous prendrons comme référence sera le produit $ng \log p$; en effet c'est approximativement le logarithme de l'ordre du groupe considéré. En ce sens nous sortons du paradigme classique où le temps de calcul doit être évalué en fonction de la taille de l'objet donné en entrée, mais nous contournons ainsi une discussion sur le modèle de la courbe $\mathcal{C}$. Cela dit, pour certains algorithmes le degré de l'équation qui est fournie pour la courbe est une donnée plus cruciale que son genre.

L'objectif ultime est l'obtention d'un algorithme dont le temps de calcul soit borné par un polynôme en $ng \log p$. Actuellement des résultats partiels vont dans cette direction, mais l'objectif n'est pas encore atteint si g et $\log p$ tendent tous les deux vers l'infini. Dans cet exposé nous tentons de donner un aperçu de la situation d'aujourd'hui.

2. Survol des algorithmes disponibles

2.1. Les algorithmes ℓ-adiques. — L'algorithme publié par Schoof en 1985 [**59**] pour compter le nombre de points d'une courbe elliptique sur un corps fini fut le premier algorithme a atteindre une complexité polynomiale. Il fut ensuite étendu aux variétés abéliennes par Pila [**54**], puis par Adleman et Huang [**1**] ainsi que Huang et Ierardi [**33**].

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. Par exemple, ce résultat couvre 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.

2.2. 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 [**52, 20, 21, 16**] et finalement une version a été prouvée dans un cadre très général par Heß [**32**]. 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'applications des algorithmes p-adiques, et désormais ces algorithmes sous-exponentiels ne sont plus les plus rapides (du moins asymptotiquement). Toutefois, si l'on est intéressé par la structure du groupe ou par des calculs de logarithme discret, cette approche est la bonne.

2.3. Les algorithmes p-adiques utilisant le relèvement canonique. — En 1999, Satoh [**57**] 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.

2.4. Les algorithmes p-adiques utilisant la cohomologie de Monsky-Washnitzer. — Peu de temps après la publication de l'algorithme de Satoh, Kedlaya [**37**] 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.

2.5. Les algorithmes p-adiques utilisant une approche à la Dwork. — Toujours au début des années 2000, Lauder et Wan [**42**] 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 [**40, 43, 44**], 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.

Une variante plus fructueuse de cette approche est l'utilisation de la théorie de la déformation. Lauder [**41**] a ouvert la voie, et des améliorations et extensions successives [**35, 34, 36, 11**] ont récemment abouti à des algorithmes compétitifs. On trouvera une brève présentation de ces développements récents dans le survol [**12**].

Dans cet exposé, nous ne parlerons pas plus de cette classe d'algorithmes.

2.6. Les implantations. — Dans le diagramme de la figure 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.

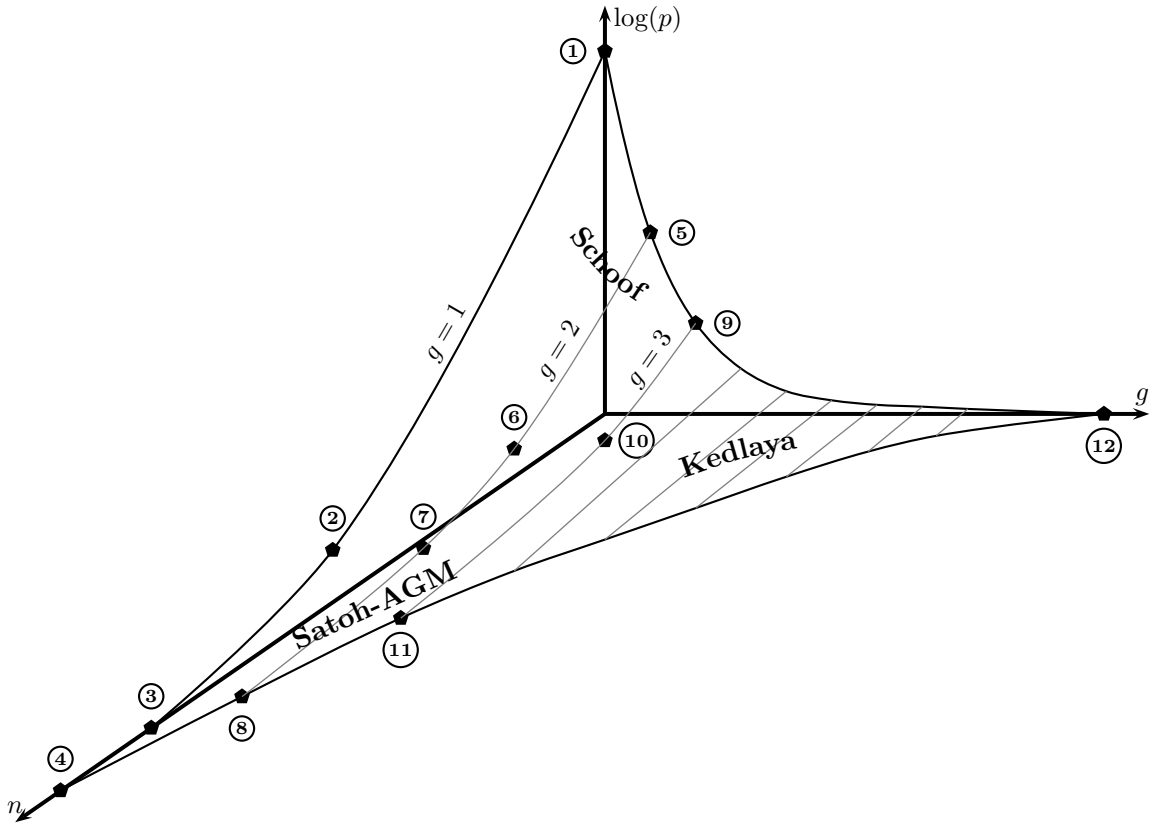
Il convient de noter que le diagramme n'a pas vocation à être précis, par exemple le point 4 correspond en réalité à $n = 130\,020$ et le point 3 à $n = 50\,021$, ce qui n'est manifestement pas à l'échelle. Nous avons préféré faire un schéma le plus lisible possible et qui permette de situer les différents 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 peuvent certainement être poussées plus loin aujourd'hui avec une implantation extrêmement optimisée.

3. Complexité des opérations algébriques élémentaires

Les algorithmes que nous étudions font fréquemment intervenir des objets algébriques de grande taille, tels que de grands entiers, ou des polynômes de grand degré. Dans ce contexte, il est tout à fait pertinent d'utiliser des algorithmes asymptotiquement rapides pour les opérations élémentaires, typiquement à base de transformée de Fourier rapide. La première opération de base que l'on étudie est la multiplication. En effet l'addition est typiquement un ordre de grandeur plus rapide que la multiplication et on négligera les additions lors de l'analyse d'un algorithme.

FIG. 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.



Nous utiliserons la même notation $\mathsf{M}(n)$ pour désigner l'une des deux notions suivantes :
- le nombre d'opérations élémentaires dans un anneau R nécessaires pour multiplier deux polynômes de $R[x]$ de degré au plus n ;
- le nombre d'opérations booléennes nécessaires pour multiplier deux entiers d'au plus n chiffres binaires (bits).

Pour la multiplication d'entiers, on a les estimations suivantes pour $\mathsf{M}(n)$, selon la méthode utilisée. L'algorithme naïf donne $\mathsf{M}(n) \in O(n^2)$. L'algorithme de Karatsuba a une complexité qui donne $\mathsf{M}(n) \in O(n^{\log_2(3)}) = O(n^{1.58\cdots})$. Et finalement l'algorithme de Schönhage-Strassen a une complexité de $\mathsf{M}(n) \in O(n \log n \log\log n)$. Ce dernier algorithme n'est pas simple à implanter et n'est meilleur que les précédents que pour de grandes valeurs de n. Nous renvoyons le lecteur à [**67**] pour une description de ces méthodes.

Des variantes de ces algorithmes s'appliquent aussi pour les polynômes sur un anneau quelconque R. Cela dit, dans les cas simples, il est possible de ramener le problème de multiplication de polynômes à la multiplication d'entiers par la méthode de Kronecker-Schönhage. Par exemple, soient f et g deux polynômes de $\mathbb{Z}[x]$ de degré au plus d et à coefficients positifs de taille au plus n bits que l'on veut multiplier. On commence par calculer $a = f(B)$ et $b = g(B)$, où B est un entier plus grand que les coefficients du produit cherché. Ensuite on calcule le produit $c = ab$ dans $\mathbb{Z}$. Et pour finir, on calcule le polynôme $h = fg$ tel que $c = h(B)$. Cette dernière étape est facile car la taille de B garantit que les coefficients ne se superposent pas : il suffit d'écrire l'entier c en base B. Ainsi, sous réserve que n soit assez grand par rapport à d, on a multiplié f et g en temps $O(\mathsf{M}(nd))$.

Légende pour la figure 1

N°	p	n	g (type)	Réf.	Algorithme	Commentaire
①	$\approx 10^{2500}$	1	1	[**22**]	Schoof	
②	5	569	1	[**57**]	Satoh	Première implantation d'un algorithme p-adique
③	2	50 021	1	[**29**]	AGM	
④	2	130 020	1	[**29**]	AGM	La valeur de n permet l'utilisation de base normale gaussienne
⑤	$\approx 5 \cdot 10^{24}$	1	2	[**27**]	Schoof	
⑥	$2^{32} - 5$	3	2	[**6**]	Schoof	Utilisation de l'opérateur de Cartier-Manin
⑦	2	4 001	2	[**30**]	AGM/Richelot	
⑧	2	32 770	2	[**46**]	AGM/Borchardt	La valeur de n permet l'utilisation de base normale gaussienne
⑨	$\approx 2,8 \cdot 10^{17}$	1	3 (Picard)	[**3, 68**]	Ad-Hoc	Utilisation d'un algorithme exponentiel adapté aux courbes de Picard
⑩	251	7	3 (hyperell.)	[**26**]	Kedlaya	
⑪	2	4 098	3 (hyperell.)	[**46**]	AGM/Borchardt	
⑪	2	5 002	3 (non-hyperell.)	[**56**]	AGM/Borchardt	
⑫	2	1	350 (hyperell.)	[**65**]	Kedlaya	

Cette stratégie se généralise à des anneaux plus compliqués où l'on doit évaluer plusieurs indéterminées en des entiers bien choisis de manière à pouvoir lire le résultat après une opération sur les entiers. Ainsi, si l'on a deux «objets algébriques» de taille n bits, on peut souvent les multiplier en temps $O(\mathsf{M}(n))$ pour la fonction M correspondant à la multiplication d'entiers.

Après la multiplication viennent les opérations élémentaires plus complexes, telles que la division, le PGCD, le PGCD étendu, le résultant de deux polynômes en une variable, l'interpolation ou l'évaluation d'un polynôme en une famille de points. En fait, pour toutes ces tâches il existe des algorithmes (en général du type *diviser pour régner*) dont le temps de calcul est en $O(\mathsf{M}(n))$ ou en $O(\mathsf{M}(n) \log n)$ où n est la taille des objets considérés. Là encore, nous renvoyons à [**67**] qui contient une description de certains de ces algorithmes ainsi que des références pour approfondir dans cette direction.

En conclusion, dans la suite de cet exposé, nous considérerons que les opérations élémentaires sur des objets algébriques de taille n peuvent s'effectuer en $\tilde{O}(n)$ où la notation $\tilde{O}()$ signifie que l'on néglige l'écriture de certains termes logarithmiques.

Notons pour finir que pour la plupart des records mentionnés ci-dessus, ce sont effectivement ces algorithmes asymptotiquement rapides qui sont implantés, si bien que les estimations de complexité que nous ferons reflètent assez bien la réalité.

4. L'algorithme de Schoof et ses généralisations

4.1. 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. Comme dit plus haut, dans le cas où A est la jacobienne d'une courbe $\mathcal{C}$ de genre g, c'est équivalent au calcul de la fonction Zêta de $\mathcal{C}$.

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.

4.2. 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 correspondances 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 [**60**] 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 [**60**] 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 [**15, 45, 47**] 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.

4.3. 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 [**54, 33, 1**].

4.3.1. Le cas du genre 2. — Afin d'illustrer les propos ci-dessus, nous allons développer le cas des courbes de genre 2, qui est à notre connaissance le seul cas non elliptique pour lequel une implantation existe. Toute courbe de genre 2 est hyperelliptique. Nous nous restreignons au cas où il existe un point de Weierstraß rationnel, car dans ce cas la loi de groupe est un peu plus simple. 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. Notons ∞ l'unique place au-dessus du point à l'infini de $\mathcal{C}$. Les points de la jacobienne de $\mathcal{C}$ peuvent être vus comme des classes de diviseurs sur $\mathcal{C}$ modulo les diviseurs principaux. Nous

utiliserons la représentation de Mumford pour stocker un représentant canonique d'une classe de diviseurs : chaque point de $\mathrm{Jac}(\mathcal{C})$ peut être représenté de manière unique par un couple de polynômes $\langle u(x), v(x)\rangle$, où u est unitaire, de degré au plus 2, v est de degré inférieur à celui de u, et de plus u divise $v^2 - f$. La correspondance avec les classes de diviseurs est la suivante : la variété de l'idéal engendré par $u(x)$ et $y - v(x)$ est formée d'au plus deux points sur la courbe $\mathcal{C}$; la classe représentée est celle du diviseur formé par la somme de ces points moins $\deg u$ fois le point ∞. Les détails justifiant la représentation de Mumford peuvent être trouvés dans [**14**].

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 [**8**], 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 [**27**] 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 [**4**]. Une autre version plus récente et libre [**24**] a été implantée en C++ sur la base de la bibliothèque NTL [**62**]. On y trouve en particulier une fonction qui calcule exactement l'idéal I_ℓ tel que décrit dans cette section. À l'aide de ce logiciel, on a pu calculer la ℓ-torsion d'une courbe de genre 2 jusqu'à $\ell = 29$, sur un corps fini premier à environ 10^{25} éléments. Ceci implique de manipuler des polynômes de degré quelques millions, ce qui est monnaie courante avec les ordinateurs et les algorithmes actuels.

4.3.2. 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 [**28**], des équations modulaires sont définies de la manière suivante. La définition de ces équations modulaires est essentiellement la suivante. Soit $D = \langle x^2 + u_1 x + u_0, v_1 x + v_0 \rangle$ 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 [**13**].

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 dans [**19**] mais pour l'instant cela reste pour des ℓ très petits.

Concernant le genre supérieur à 2, 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 [**54, 33, 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.

5. L'algorithme de Satoh et ses généralisations

5.1. 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 [**61**].

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 décrite en Section 3 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.

5.2. Principe général de l'algorithme de Satoh. — Soit E une courbe elliptique définie sur $\mathbb{F}_q$. L'idée de Satoh 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 [**48**].

***Théorème 1* (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*

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

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 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.

5.2.1. 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 :

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

ainsi que les relations galoisiennes :

$$\forall i \in [0, n-1],\ j_{\mathcal{E}_{i+1}} = \sigma(j_{\mathcal{E}_i}). \tag{2}$$

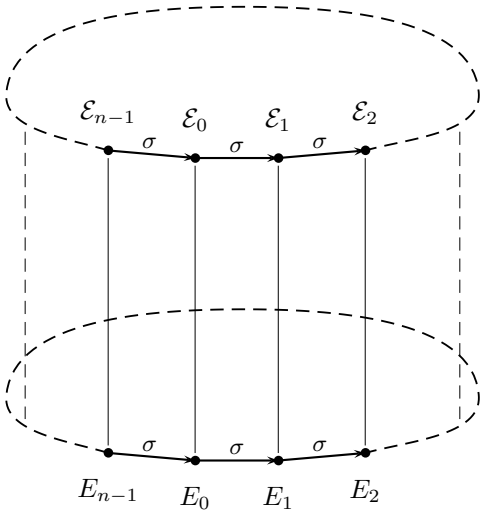


FIG. 2. Cycle et cycle relevé de courbes conjuguées

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 (1) est inversible. Nous renvoyons à l'article original [**57**] 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 donnons 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 (1), l'autre étant la relation galoisienne (2).

5.2.2. 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.

Il reste à étudier la précision à laquelle les calculs doivent être menés. 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 $n/2 + O(1)$.

5.2.3. 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 [**65**]. 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 [**23**] et Skjernaa [**63**]. Ensuite, Vercauteren [**66**] et Mestre [**51**] 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 [**58**] 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 [**38**] 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 $O()$), Gaudry [**25**] a montré que la méthode AGM pouvait être intégrée aux algorithmes asymptotiquement rapides afin de bénéficier des avantages des formules extrêmement compactes. Dans cette optique, Kohel [**39**] et Madsen [**49**] ont proposé des adaptations de la méthode AGM aux petites caractéristiques différentes de 2.

5.2.4. 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 univarié 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.

5.3. L'algorithme de Harley. — Nous donnons quelques détails 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 :

$$v + e \frac{\partial \Phi_p}{\partial x}(X, X^\sigma) + e^\sigma \frac{\partial \Phi_p}{\partial y}(X, X^\sigma) \equiv 0 \mod p^k. \tag{3}$$

On a supposé X connu et σ calculable efficacement, de sorte que la seule inconnue de l'équation (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.

5.3.1. 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 $X + p^{k/2}E$.

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.

5.3.2. Calcul efficace du Frobenius et de la norme. — 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 $z = z_0 + z_1 x + \cdots + z_{n-1}x^{n-1}$ 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 $\sigma(z) = z_0 + z_1 x^p + \cdots + z_{n-1}x^{p(n-1)}$. 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. Une fois ce relèvement calculé, 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 [**58**]) ayant une complexité moins bonne mais plus efficace pour les tailles atteignables aujourd'hui.

Nous sommes désormais en mesure de donner la complexité totale de l'algorithme. 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.

5.4. Les algorithmes de Mestre pour le genre supérieur. — 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 [**53**] 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.

5.4.1. 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 [**55, 5**]. 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 [**50**] 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 [**9**] permettrent de s'affranchir de la contrainte $p = 2$.

5.4.2. 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 matricelle.

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 [**46**].

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 [**56**] 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.

6. L'algorithme de Kedlaya

Tout comme l'algorithme de Satoh, l'algorithme de Kedlaya 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).

6.1. 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.

6.2. 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 2. — *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. Réécrire $\omega_{i,j}$ 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.

6.2.1. 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 $u_{n+1} = \frac{1}{r}((r+1)u_n - a u_n^{r+1})$ 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.

6.2.2. 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.

6.2.3. 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.

6.2.4. 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 [**26**], on peut faire baisser la complexité à $\tilde{O}(pn^3g^4)$. Plus récemment, en s'inspirant de [**7**], Harvey [**31**] est parvenu à ramener la dépendance en p à $\sqrt{p}$.

6.3. 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 [**18**]. Dans cet algorithme, de manière similaire à l'algorithme original de Keldaya, 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 $\tilde{O}(g^4n^3)$ pour une courbe hyperelliptique générique de genre g sur $\mathbb{F}_{2^n}$.

Une deuxième extension de l'algorithme de Kedlaya [**17**] 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 $\tilde{O}(g^5n^3)$.

Ce dernier algorithme a encore été étendu dans [**10**] 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.

7. Conclusion

Le calcul du nombre de points d'une courbe sur un corps fini est un sujet qui a été très actif ces dernières années, notamment à cause des applications à la cryptographie. En ce qui concerne les courbes elliptiques, on peut considérer que le problème est désormais résolu de manière très satisfaisante : des nombres de points de quelques milliers de bits, voire quelques dizaines de milliers de bits en petite caractéristique sont désormais calculables.

Pour les courbes de genre supérieur à 1, on est passé en quelques années d'une situation où l'on ne savait essentiellement pas faire autre chose que d'utiliser des algorithmes de complexité exponentielle à une situation où l'on peut traiter des cas très grands, notamment en petite caractéristique. Le cas de la grande caractéristique reste problématique, déjà pour le cas le plus simple du genre 2. Les progrès effectués sur l'algorithme de Schoof ne sont pas suffisants pour que l'on puisse considérer que le problème est résolu. En particulier, la question de l'existence d'un algorithme de

comptage de points qui fonctionnerait en temps polynomial en le genre et la caractéristique est encore complètement ouverte. Une idée fondamentalement nouvelle reste à découvrir pour changer la donne sur ce point.

Références

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

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

[5] 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.

[6] 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.

[7] 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.

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

[9] R. Carls and D. Lubicz. A p-adic quasi-quadratic point counting algorithm. *International Mathematics Research Notices*, pages 698–735, 2009.

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

[11] 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.

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

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

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

[15] 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.

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

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

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

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

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

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

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

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

[24] P. Gaudry. *NTLJac2, Tools for genus 2 Jacobians in NTL.* `http://www.loria.fr/~gaudry/NTLJac2/`.

[25] 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.

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

[27] 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.

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

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

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

[31] D. Harvey. Kedlaya's algorithm in larger characteristic. *International Mathematics Research Notices*, 2007:Article ID rnm095, 29 pages, 2007.

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

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

[34] H. Hubrechts. Point counting in families of hyperelliptic curves in characteristic 2. *LMS Journal of Computation and Mathematics*, 10:207–234, 2007.

[35] H. Hubrechts. Point counting in families of hyperelliptic curves. *Found. Comput. Math.*, 8:137–169, 2008.

[36] H. Hubrechts. Quasi-quadratic elliptic curve point counting using rigid cohomology. *J. Symbolic Comput.*, 44:1255–1267, 2009.

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

[38] 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.

[39] 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.

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

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

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

[43] 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.

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

[45] 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.

[46] R. Lercier and D. Lubicz. A quasi quadratic time algorithm for hyperelliptic curve point counting. *J. Ramanujan Math. Soc.*, 12:399–423, 2006.

[47] 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.

[48] 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`.

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

[50] 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.

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

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

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

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

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

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

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

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

[66] 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.

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

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

Pierrick Gaudry