Table des matières

Introduction

Tum Cotta, inquit [...] : Sed etiamsi est aliquanto spissius, aut si ego sum tardior, profecto nunquam conquiescam, neque defatigabor antequam [...] percepero. [a]

Cicéron, *De Oratore.*

---

a. À cet instant, Cotta l'interrompit [...] : *Mais quand bien même ils auraient encore plus d'obscurité, quand je me sentirais mal secondé par mon intelligence, je ne me découragerai pas, et je suis décidé à ne prendre ni repos ni relâche, avant [...] d'avoir achevé de comprendre.*

Cotta aurait connu les logarithmes discrets qu'il ne les aurait pas trouvé moins obscurs. Et, comme ceux d'entre nous dont la curiosité est toujours aiguillonnée par l'énigme et la complexité, je rêve à croire qu'il ne se serait pas *découragé* devant le problème suivant :

Soit $(G, \times)$ un groupe cyclique et g un générateur de G. Considérons alors l'application $x \mapsto g^x$, qui n'est autre que l'exponentiation discrète à la puissance x en base g, l'élément x étant un entier. Cette opération partage de nombreuses propriétés avec l'exponentiation ordinaire, comme l'égalité $g^{x+y} = g^x \cdot g^y$. L'inverse de cette opération consiste, étant donné un élément h dans G, à déterminer un entier x tel que :

$$h = g^x.$$

L'exponentiation discrète étant périodique, de période $|G|$, cet entier x n'est pas unique. En revanche $x \bmod |G|$ est bien déterminé : il sera appelé le *logarithme discret* de h en base g et noté $\log_g(h)$. Par abus de langage, on confond souvent ce logarithme avec son unique représentant dans l'intervalle $[\![0, |G| - 1]\!]$. De plus, on dit de tout entier x satisfaisant $g^x = h$ que c'est un logarithme discret de h. Les logarithmes discrets de h sont donc les entiers appartenant à la classe de $\log_g(h)$ modulo $|G|$. Comme pour les logarithmes

classiques, nous disposons d'une égalité faisant état de la relation entre la multiplication de deux éléments dans le groupe et l'addition des logarithmes correspondants. Plus précisément :

$$\log_g(h \cdot j) \equiv \log_g(h) + \log_g(j) \pmod{|G|}.$$

On considère que le problème du logarithme discret est résolu dans G dès lors que pour tout élément de G il est possible de déterminer efficacement son logarithme discret – ou l'un de ses représentants. Ainsi, lorsque l'ordre du groupe G est connu, résoudre le problème du logarithme discret dans G consiste simplement à expliciter l'inverse de l'isomorphisme :

$$\begin{array}{ccc} \mathbb{Z}/|G|\mathbb{Z} & \mapsto & G \\ x & \mapsto & g^x \end{array}$$

sous une forme calculatoirement efficace. Un sous problème intéressant consiste à retrouver le logarithme discret d'une partie non négligeable des éléments de G. Par exemple, les standards habituels de sécurité cryptographique stipulent que, pour que l'on puisse prétendre avoir résolu le problème pour une famille de groupes, la fraction des éléments dont on connait un logarithme au sein de l'un de ces groupes G doit être asymptotiquement au moins aussi grande que l'inverse d'un polynôme en $\log(|G|)$.

La description du groupe G détermine la difficulté du problème ; en effet, si G est exactement le groupe additif $(\mathbb{Z}/n\mathbb{Z}, +)$, calculer des logarithmes discrets dans celui-ci est immédiat, puisqu'il s'agit, étant donné h et g deux éléments du groupe, de retrouver un entier x tel que $h = g \cdot x$. Un calcul de pgcd nous donne aisément et efficacement l'inverse de g dans le groupe, et nous pouvons conclure en calculant simplement $x = g^{-1} \cdot h$. En revanche, dans le modèle du groupe générique, qui sera décrit au paragraphe 1.5, des résultats stipulent qu'il s'agit bel et bien d'un problème difficile : dans un groupe générique, le calcul d'un logarithme discret coûte de l'ordre de $\sqrt{p}$ opérations, où p désigne le plus grand facteur de l'ordre du groupe. Si cela reste plus rapide qu'une recherche exhaustive sur toutes les valeurs possibles, ce résultat montre qu'un tel calcul demeure absolument infaisable, même pour des groupes de taille modérée. Ma thèse porte ainsi sur l'étude du problème du logarithme discret dans certains groupes – les groupes multiplicatifs des éléments inversibles d'un corps fini – problème qui cristallise un des piliers de la cryptographie moderne. Si les efforts pour résoudre celui-ci concentrera toute notre attention dans le corps de ce manuscrit, commençons par retracer le lien étroit qui relie ces logarithmes particuliers à la *sciences des écritures secrètes.*

0.1 Cryptographie asymétrique et problèmes difficiles

0.1.1 Alan Turing ou la cryptographie moderne

Jusqu'à la fin du XIX^{e} siècle la cryptographie s'attache principalement à décrire des procédés de chiffrement, c'est-à-dire à convertir un message brut et clair en un texte illisible auquel seul un petit nombre d'élus sont en mesure de rendre le sens initial. L'idée naïve consiste par exemple à changer chaque lettre par une autre, ou à réarranger l'ordre de celles-ci. Parallèlement, les premières cryptanalyses ne se font pas attendre : à chaque nouvelle méthode de chiffrement employée, les adversaires reprennent leurs illégitimes tentatives d'en *briser le code*. Ce n'est qu'avec l'avènement de la radio en 1903 et les enjeux de la première guerre mondiale que la cryptographie marque son premier tournant. L'électricité et l'automatisation permettent un chiffrement progressivement plus travaillé, complexifiant celui qu'un soldat ou qu'un officier sur le champ de bataille ne parvenait à effectuer seul jusqu'à lors. En plus de la confidentialité des messages – l'information n'est accessible qu'à ceux auxquels l'accès est autorisé – on cherche, sans succès d'abord, à garantir leur authenticité – l'identité de celui qui nous écrit est certifiée – tout comme leur intégrité – une vérification atteste que les données n'ont pas été altérées pendant la transmission du message. Enfin, l'horreur de la seconde guerre mondiale et la nécessité de ramener la paix lèvent le jour sur le premier des cryptographes modernes : Alan Turing. En achevant la cryptanalyse de la machine Enigma utilisée par les allemands, Turing crée non seulement de toutes pièces ce que l'on considère comme le premier des ordinateurs, mais ouvre par ailleurs le champ d'une cryptographie nouvelle, portée par l'électronique. L'informatique balbutie. La cryptographie quitte l'enfance.

Un souci majeur contrarie pourtant son développement pendant trois décennies successives. Pour être honnête, bien qu'entrée dans l'ère moderne, la cryptographie bute encore sur un obstacle de longue date. Pour mieux saisir cette contrariété, nous reprenons l'image classique du coffre et du cadenas : admettons que le chiffrement corresponde à la dissimulation d'un message dans un coffre, suivi du verrouillage de celui-ci par une clef donnée. Le déchiffrement doit alors permettre à tout détenteur de cette même clef, après voyage du coffre, de l'ouvrir pour révéler son contenu. La transmission du coffre-message peut prétendre aux plus hautes mesures de sécurité, qu'en est-il du voyage de la clef ? En effet, loin de l'objet matériel, celle-ci revêt très tôt une réalité plus abstraite. Qu'il s'agisse d'une correspondance de symboles, d'un mot, ou dorénavant d'un nombre plus ou moins grand d'octets, le souci de son transfert repose alors sur l'existence de méthodes permettant de faire circuler en amont du protocole cette information par des voies non sécuri-

sées... Ce qui est précisément le problème initial, qui, en un sens, n'est donc toujours pas résolu !

0.1.2 L'avènement de la clef publique

New Directions in Cryptography bouleverse l'année 1976 [DH76]. Petite révolution dans le domaine comme son titre le laisse présager, cet article fondateur de Diffie et Hellman impulse un changement de paradigme en formalisant la notion de cryptosystème asymétrique, c'est-à-dire de protocole où l'expéditeur et le destinataire n'ont aucune clef commune préalable. Exit alors le souci de sa transmission. Le schéma toujours d'actualité recommande la création non plus d'une clef partagée mais de deux clefs différentes associées ; l'une d'entre elle n'est pas secrète et peut être distribuée librement, c'est la clef publique, la seconde, privée donc, ne quitte en revanche jamais son propriétaire. Ainsi pour chiffrer un message il convient d'utiliser la clef publique de son destinataire, tandis que ce dernier se sert uniquement de la clef privée associée, sa clef, pour déchiffrer. Le cryptosystème RSA [RSA78] dessine l'année suivante une illustration exemplaire du chiffrement asymétrique. Témoignage de leur importance respective, ces deux résultats ont valu à leurs auteurs l'attribution du prix Turing, en 2002 pour Rivest, Shamir et Adleman, et en 2015 pour Diffie et Hellman.

0.1.3 A la recherche des problèmes difficiles

Les protocoles asymétriques utilisés lorsque l'on souhaite chiffrer et déchiffrer des messages, signer un document numérique ou s'authentifier auprès de sa banque, sont tous construits à partir d'hypothèses calculatoires réputées difficiles. Bien choisir ces hypothèses est plus délicat qu'il n'y parait : il faut disposer d'un problème difficile à résoudre pour un attaquant [4] alors que le déchiffrement doit être facile pour un utilisateur légitime muni de la bonne clef. On comprend dès lors la nécessité de disposer d'un cryptosystème pour lequel il est difficile de trouver la bonne clef, alors que par construction il est nécessairement facile de tester si une clef donnée est correcte. Du point de vue de la théorie de la complexité, cette double condition nécessaire explique pourquoi la cryptographie moderne n'a de sens que sous l'hypothèse que les deux classes bien connues **NP** et **P** soient distinctes. Rappelons que

4. En particulier, lorsque l'on cherche un problème difficile, il faut se souvenir qu'un attaquant disposant de ressources calculatoires immenses peut toujours procéder au déchiffrement, en essayant simplement toutes les clefs possibles. Il s'agit donc de chercher un problème dont les meilleures attaques connues ne sont pas plus performantes que cette méthode de référence, dite par force brute, dont on ne peut s'abstraire.

la classe **P** correspond aux problèmes résolubles en temps polynomial tandis que la classe **NP** regroupe ceux pour lesquels une solution, une fois donnée, est vérifiable en temps polynomial. En effet, notre problème est dans **NP** car il est facile de tester si une clef est correcte, et pour prétendre être difficile, il ne faut pas qu'il soit dans **P**, afin que la bonne clef ne puisse pas aisément être retrouvée. Or, savoir si **P** et **NP** sont deux classes égales ou distinctes est une question non résolue. Pire, elle est considérée comme le problème ouvert le plus important du domaine, importance soulignée par son appartenance aux sept problèmes à 1 million de dollars du *Clay Millenium Challenge*.

Par conséquent, aujourd'hui nous ne savons pas s'il existe de véritables problèmes difficiles sur lesquels appuyer les fondements de la cryptographie. Faute de mieux, la communauté choisit de porter son attention sur des questions qui ont été suffisamment étudiées pour que l'absence de solution soit un bon indice de leur difficulté. Pour cette raison, la cryptographie à clef publique, ou cryptographie asymétrique, se base généralement sur des problèmes issus des mathématiques. Comme l'essentiel des calculs se fait par ailleurs sur des ordinateurs qui préfèrent manipuler un monde d'objets discrets plutôt qu'un monde continu, c'est du vaste champ de la théorie des nombres qu'ont été sélectionnés deux candidats incontournables : le problème de la factorisation des entiers et celui du logarithme discret.

0.2 Où trouve-t-on des logarithmes discrets ?

L'histoire du logarithme discret en théorie des nombres précède largement la découverte du protocole de Diffie et Hellman. De fait, le calcul de ces logarithmes attire déjà l'attention de Kraïtchik [Kra22, Chap. V] au début du siècle dernier. Hommage posthume, le nom de la méthode sur laquelle nous nous appuierons tout au long de ce manuscrit – la méthode du calcul d'indice – tire son nom de ce premier ouvrage. Pour Kraïtchik, l'indice de g^x dans le groupe engendré par g désigne ainsi exactement celui que nous appelons aujourd'hui le logarithme discret de g^x, c'est-à-dire l'entier x. Autrement dit, de manière amusante, nous avons assisté à un lent glissement de la sémantique du terme : si *un calcul d'indice* ne représentait rien d'autre il y a un siècle qu'un calcul de logarithme discret, actuellement la *méthode du calcul d'indice* couvre certes une vaste famille destinée à calculer ces logarithmes, mais non la totalité des algorithmes de résolution du problème.

Néanmoins, la plus ancienne apparition dont on trouve trace précède le manuscrit de Kraïtchik. Dès 1849 certains de nos logarithmes discrets se dissimulent déjà sous les logarithmes de Zech dont la définition est la suivante :

Définition 0.2.1. Si α est l'élément primitif d'un corps fini, le logarithme de Zech de n en base α est défini comme l'entier $\log_\alpha(n)$ tel que $\alpha^{\log_\alpha(n)} = 1+\alpha^n$.

Le logarithme de Zech d'un entier est donc le logarithme discret d'un élément particulier du corps fini. L'intérêt des logarithmes de Zech réside dans la traduction des additions dans le corps en question par une simple exponentiation. En effet, l'addition $\alpha^m + \alpha^n$ qui est égale à $\alpha^m(1 + \alpha^{m-n}) = \alpha^{m+\log_\alpha(n-m)}$ peut ainsi se réécrire comme α^m à la puissance $\log_\alpha(n-m)$. Le calcul préalable de tables de logarithmes de Zech permettait l'exécution plus rapide d'opérations arithmétiques dans des corps finis.

En cryptographie, l'intérêt que l'on accorde à ce problème remonte lui aussi quelques années avant les avancées de Diffie et Hellman. Par exemple, la sécurité des cryptosystèmes à clefs secrètes qui impliquent des registres à décalage à rétroaction linéaire (ou LFSR, de l'anglais *linear feedback-shift registers*) est étroitement liée aux calculs de logarithmes discrets dans des corps finis binaires. Plus précisément, localiser la position d'une sous-séquence donnée au sein de la sortie d'un LFSR peut se traduire par un problème de calcul de logarithme discret dans le corps fini défini par le polynôme de feedback – sous réserve que celui-ci soit bien entendu irréductible, ce qui est souvent le cas.

Sans surprise, l'utilisation la plus classique du problème du logarithme discret remonte en 1976, à la présentation du protocole de Diffie et Hellman [DH76]. Cette découverte justifie à elle seule une longue parenthèse ; aussi nous repoussons les détails au paragraphe 0.2.1. Plus tard, en 2000, l'introduction des couplages au sein de la discipline (versions journal [Jou04, BF03]) ravive encore l'attention apportée au même problème, bien que cette fois-ci les groupes dans lequels les logarithmes sont recherchés présentent une structure atypique. Il s'agit par exemple de certains corps finis comportant un degré d'extension composé ou une caractéristique de taille moyenne. L'utilisation des protocoles basés sur les couplages fait l'objet du paragraphe 0.2.2.

0.2.1 Echange de clef de Diffie-Hellman

Ainsi les projecteurs se tournent en 1976 sur nos logarithmes, auparavant relativement discrets. Diffie et Hellman proposent cette année là un mécanisme d'échange de clef entre deux entités sans connaissance d'un secret commun préalable [DH76], résolvant par conséquent cette question de longue date : comment communiquer de manière sécurisée avec une personne que l'on n'a précédemment jamais vue ?

Echange de clef

L'approche classique de l'échange de clef de Diffie et Hellman, toujours utilisée de nos jours, s'articule autour des grandes lignes suivantes. Si Alice et Bob souhaitent créer une clef commune, ils s'accordent en premier lieu sur un groupe cyclique G et un générateur g de ce groupe. Plusieurs utilisateurs peuvent choisir un même groupe et un même générateur – et c'est à propos l'usage courant – bien que cet emploi commun puisse mener à une légère diminution de la sécurité du système. Alice choisit ensuite une entier aléatoire a, calcule g^a puis transmet ce dernier à Bob par l'intermédiaire d'un canal public. Pendant ce temps Bob choisit de même un entier aléatoire b pour envoyer le résultat de g^b à Alice. Alice et Bob sont maintenant en possession d'une valeur commune qu'il leur suffit de calculer indépendamment, chacun pour soi :

$$(g^b)^a = g^{a \cdot b} = (g^a)^b.$$

C'est cette valeur commune qui fera office de secret partagé pour toutes les utilisations ultérieures.

Hypothèses de difficulté

La sécurité du système dépend de l'hypothèse faite selon laquelle une tierce personne qui intercepterait l'échange, et connaitrait ainsi les valeurs de g, g^a, et g^b, serait incapable de calculer le secret commun. En particulier, cette hypothèse suppose que l'espion en question ne sait pas résoudre le problème du logarithme discret dans G. En effet, si le problème du logarithme discret dans ce groupe est simple à résoudre, il peut donc calculer a (ou b) et retrouver le secret commun $g^{a \cdot b}$. Cependant, le lien entre le problème du logarithme discret à proprement parler et celui de calculer g^{ab} à partir de g, g^a, et g^b, qui est repertorié sous le nom de problème de Diffie-Hellman calculatoire (CDH, pour *computation Diffie-Hellman* problem en anglais) demeure complexe. Nous savons seulement qu'étant donné un algorithme qui résout le problème de Diffie-Hellman calculatoire, il est possible de calculer efficacement des logarithmes discrets, bien que cela nécessite de trouver en amont un objet auxiliaire qui rende ce calcul possible. Plus précisément, supposons que la cardinalité de G soit première, alors nous avons besoin de considérer une courbe elliptique définie sur $\mathbb{F}_{|G|}$ telle que l'ordre de celle-ci ne possède que des petits facteurs premiers. On ne sait malheureusement ni construire cette courbe, ni même en prouver l'existence.

Cependant, il n'existe pour l'heure aucun groupe pour lequel nous sachions résoudre le problème CDH sans être capable de résoudre le problème

du logarithme discret. Afin d'étudier et de prouver la sécurité d'un grand nombre de protocoles cryptographiques, il est par ailleurs souvent nécessaire de considérer le problème de décision associé : étant donné g, g^a, g^b et h, décider si h est la valeur de g^{ab} ou non. Ce problème est repertorié comme le problème de Diffie-Hellman décisionnel (DDH, pour *decision Diffie-Hellman* problem en anglais).

D'autres problèmes calculatoires ou de décision liés au problème du logarithme discret ont été introduits au fil des années comme fondations alternatives pour différents cryptosystèmes. Néanmoins la comparaison de ces hypothèses variées n'est pas aisée. Aussi, dans une tentative de simplification du paysage, pour comprendre les liens qui unissent tous ces problèmes et pour faciliter une vue d'ensemble plus claire, Boneh, Boyen et Goh [BBG05] ont proposé l'hypothèse Uber (*Uber assumption*) qui regroupe l'ensemble de ces variantes. En outre l'hypothèse Uber est prouvée sûre dans le modèle du groupe générique.

Le problème du logarithme discret ou DLP (de l'anglais *Discrete Logarithm Problem*) reste pourtant fondamental en tant que tel. D'un point de vue mathématique il cristallise en effet une question bien plus naturelle que celles soulevées par les problèmes ci-dessus évoqués ; en pratique, aucun de ceux-ci n'a par ailleurs été résolu indépendamment du DLP.

Espion passif VS espion actif

Toute la description précédente repose sur une hypothèse extrêmement importante quoiqu'implicite : l'attaquant, passif, ne peut qu'écouter les communications qui transitent entre Alice et Bob. Dans le cas d'un espion actif la sécurité s'écroule en revanche. C'est le cas si l'on tente de monter une attaque de l'homme au milieu, c'est-à-dire lorsqu'une tierce personne malveillante feint d'être Bob lorsqu'elle communique avec Alice, et d'être Alice lorsqu'elle communique avec Bob. Plus précisément, elle commencera par échanger dans une phase initiale deux clefs indépendantes, l'une avec Alice, l'autre avec Bob. Elle transmettra ensuite tous les messages reçus à leur destinataire légitime après les avoir chiffrés de nouveau avec la clef de celui-ci. Cette manière de procéder lui permet alors de déchiffrer la totalité des communications ayant cours, sans être détectée. Un point crucial lors de la conception de cryptosystèmes basés sur les logarithmes discrets consiste donc à réfléchir à la mise en place de mesures de sécurité allant à l'encontre de telles incursions.

0.2.2 Cryptosystèmes basés sur les couplages

Les années 80 soulèvent progressivement une question naturelle, suite à l'échange de clef de Diffie-Hellman : existe-t-il un protocole de création de clef, en un tour seulement, qui permettrait d'accorder trois entités distinctes, et non plus deux, tout en résistant aux écoutes d'une personne extérieure ? C'est l'introduction en 2000 par Joux [Jou04] d'une méthode simple basée sur les couplages bilinéaires qui permet de répondre pour la première fois par l'affirmative à cette question ouverte. En effet, la création d'une clef commune entre plus de deux entités nécessitait jusqu'ici au moins deux tours d'interactions entre les participants ; le protocole de Burmester-Desmedt [BD94] illustre ainsi une solution classique employée à l'époque pour un nombre arbitrairement grand d'utilisateurs.

Exposons les grandes lignes du protocole de Joux qui ne demande qu'un seul tour. Si Alice, Bob et Charlie souhaitent créer une clef secrète commune, ils s'accordent au préalable sur un groupe additif $G_1 = \langle P \rangle$ d'identité $\mathcal{O}$, ainsi que sur un second groupe multiplicatif G_2 de même ordre et d'identité 1. Parallèlement, ils s'accordent sur un couplage bilinéaire du premier groupe vers le second. Rappelons la définition :

Définition 0.2.2. Un couplage bilinéaire symétrique[5] d'un groupe G_1 vers un groupe G_2 est une application :

$$e : G_1 \times G_1 \to G_2$$

qui satisfait les conditions suivantes :

1. e est bilinéaire : $\forall R, S, T \in G_1, \quad e(R+S,T) = e(R,T) \cdot e(S,T)$ et $e(R,S+T) = e(R,S) \cdot e(R,T)$.
2. e est non dégénérée : Si $\forall R \in G_1, e(R,S) = 1,$ alors $S = \mathcal{O}$.

Alice choisit aléatoirement un entier secret a modulo l'ordre du groupe G_1 et transmet la valeur de aP à ses deux correspondants. De manière similaire et simultanée, Bob et Charlie choisissent chacun leur entier secret b ou c puis transmettent les valeurs de bP ou cP. Alice (respectivement Bob comme Charlie) est ainsi capable de calculer leur secret commun, à savoir :

$$K = e(bP, cP)^a = e(P,P)^{abc}$$

5. Les couplages asymétriques du type $e : G_1 \times G_1' \to G_2$ peuvent être aussi l'objet d'une étude plus poussée ; néanmoins par simplicité nous ne décrirons ici que le cas symétrique.

Hypothèses de difficulté

Nous avons vu que la sécurité des protocoles basés sur l'idée de Diffie et Hellman reposait le plus souvent sur la difficulté des problèmes CDH et DDH. De manière analogue, la sécurité des protocoles qui s'appuient sur les couplages dépendent de la difficulté à calculer $e(P,P)^{abc}$ étant donnés P, aP, bP et cP. Ce problème est répertorié sous le nom de problème de Diffie-Hellman bilinéaire calculatoire ou CBDH (de l'anglais *Computational Bilinear Diffie-Hellman problem*) aussi parfois simplement BDH, qui existe aussi sous sa forme décisionnelle. On parle alors du problème DBDH. L'intractabilité du problème CBDH est en revanche mal comprise, bien que la difficulté de CBDH soit généralement considérée comme égale à celle du plus simple des deux problèmes de logarithme discret parmi les deux groupes G_1 et G_2. En effet, si le problème du logarithme discret dans G_1 peut être résolu efficacement, un attaquant souhaitant calculer K peut retrouver a depuis aP et connaitre la valeur de $e(bP,cP)^a$. De même, si le problème du logarithme discret dans G_2 est résolu, il peut retrouver la valeur du produit bc à partir du couplage $e(bP,cP) = e(P,P)^{bc}$, connaitre ensuite les coordonnées du point bcP et finalement retrouver K, qu'il calcule comme étant égal au couplage $e(aP,bcP)$.

Réduction du DLP dans G_1 au DLP dans G_2

Une conséquence directe de l'existence de ce couplage bilinéaire réside par ailleurs dans la réduction efficace du problème du logarithme discret dans G_1 au problème du logarithme discret dans G_2. En effet, supposons que Q soit un élément de G_1 tel que $Q = xP$, alors nous obtenons l'égalité $e(P,Q) = e(P,xP) = e(P,P)^x$. Ainsi, calculer le logarithme de $e(P,Q)$ dans G_2 nous permet de retrouver la valeur de l'entier x. Cette réduction a été décrite pour la première fois par Menezes, Okamoto, et Vanstone [MOV93] pour démontrer que les courbes elliptiques supersingulières étaient bien plus faibles que leurs homologues aléatoires, puisque le problème du logarithme discret peut être transféré d'une courbe supersingulière vers un corps fini relativement petit à l'aide d'un couplage.

A la suite du protocole tri-parties et de la publication du résultat de Menezes, Okamoto, et Vanstone, la communauté cryptographique s'est intéressée de plus près à diverses applications de ces mêmes couplages. A ce sujet, deux protocoles remarquables se distinguent : le schéma de chiffrement basé sur l'identité (IBE comme *identity-based encryption scheme* en anglais) de Boneh et Franklin [BF03] et celui de signature courte de Boneh, Lynn et Shacham [BLS04]. Dès lors, une recherche accrue s'est manifestée aussi bien pour

le design et l'implémentation que pour l'analyse des protocoles cryptographiques reposant sur des couplages bilinéaires, dont l'ensemble de départ [6] n'est autre qu'un sous-groupe d'une courbe elliptique. Notons qu'une part non négligeable des travaux actuels se préoccupe aussi des couplages bilinéaires de variétés abéliennes plus générales, comme les courbes hyperelliptiques. En pratique, la plupart de ces protocoles utilisent un corps fini comme ensemble d'arrivée du couplage sous-jacent, ce qui présente donc un cas concret affecté par la résolution du problème étudié dans cette thèse.

0.2.3 Protocoles variés

Outre l'échange de clef de Diffie-Hellman et les cryptosystèmes basés sur les couplages, le logarithme discret représente à lui seul un problème à la fondation d'une plage de protocoles parfois exotiques. Historiquement, il serait curieux de ne pas rappeler le rôle essentiel joué par El Gamal [Gam85] en 1985. Peu après l'invention du cryptosystème RSA, celui-ci révèle en effet que le problème du logarithme discret peut non seulement être utile à un échange de clef, mais aussi au chiffrement des messages et à leur signature. Une publicité efficace pour nos logarithmes à l'époque. Quatre ans plus tard, Schnorr [Sch89] propose quant à lui un protocole d'identification basé sur une preuve *zero-knolewdge* de la connaissance d'un logarithme discret. Ce protocole peut être lui-même transformé en un schéma de signature, dite signature de Schnorr, à l'aide de la transformation de Fiat-Shamir [FS86].

L'objet de cette thèse n'est pas de couvrir une liste exhaustive de tous les cryptosystèmes basés sur le problème du logarithme discret, qui sont nombreux, mais nous pouvons mentionner le chiffrement de Paillier [Pai99] publié en 1999 malgré tout. Ce système s'appuie sur le groupe $\mathbb{Z}_{N^2}^*$, où $N = pq$ est un entier RSA de factorisation non donnée. En particulier, cet exemple illustre un protocole dont la sécurité repose sur le problème du logarithme discret dans un groupe de cardinalité inconnue. Il possède en outre la propriété intéressante d'être additivement homomorphique : le produit du chiffrement par Paillier de deux messages est égal au chiffrement de leur somme.

Un autre caractère marquant du logarithme discret se retrouve dans la facilité que l'on rencontre à construire des protocoles d'échange de clef munis de propriétés additionnelles, comme l'authentification simultanée de cet échange : la vérification de l'identité du correspondant est effectuée à l'intérieur même du protocole d'échange de clef.

6. Les ensembles de départ G_1 et G_1' pour les couplages asymétriques.

0.3 En quoi l'utilisation des logarithmes discrets est-elle bénéfique ?

Une grande partie des protocoles fournis par la cryptographie à clef publique, comme les signatures numériques ou l'échange de clefs, peut être réalisée à l'aide de la méthode RSA et de ses variantes. Les cryptosystèmes qui reposent sur les couplages sont une exception notable à ce constat. Cependant, même pour des protocoles plus classiques, l'utilisation de logarithmes discrets en guise de primitive plutôt que de la factorisation sous-jacente à RSA offre quelques bénéfices à souligner.

0.3.1 Avantages techniques

Réduction des tailles de clefs. L'intérêt principal provient du fait que la complexité des algorithmes actuels pour résoudre le problème du logarithme discret sur une courbe elliptique générale (ECDLP) est bien plus élevée que celle pour la factorisation d'un entier de taille comparable. Par conséquent, les cryptosystèmes se référant à des courbes elliptiques proposent actuellement la possibilité d'utiliser des clefs de tailles bien plus petites que celles requises par RSA, ou le logarithme discret sur les corps finis, pour un niveau de sécurité similaire. Par exemple, pour atteindre un niveau de sécurité équivalent au choix d'un module RSA de taille 2048 bits, l'Agence nationale de la sécurité des systèmes d'information (ANSSI) recommande le choix d'un corps fini $\mathbb{F}_p$ d'ordre premier p de 2048 bits, alors que l'emploi d'un sous-groupe d'une courbe elliptique d'ordre premier de taille 256 bits suffit [ANS14]. En plus d'autoriser une forte réduction des tailles de clef pour les protocoles asymétriques, *le problème du logarithme discret sur courbe elliptique [...] permet d'obtenir, pour des tailles de paramètres réduites, une sécurité comparable à celle exigée pour des primitives symétriques.* A l'heure actuelle, on considère de ce fait que les systèmes basés sur les courbes elliptiques surpassent les systèmes plus classiques. Néanmoins, le débat est relancé par les prises de positions récentes des agences américaines : la NSA (*National Security Agency*) conseille par exemple l'abandon de la cryptographie basée sur les courbes elliptiques.

Confidentialité persistante. Lorsque l'on utilise RSA en vue de procéder à un échange de clef, l'approche habituelle consiste à ce que l'une des deux parties choisisse une clef secrète aléatoire, la chiffre avec la clef publique RSA de son correspondant, puis l'envoie par un canal non sécurisé. Un adversaire qui enregistre toutes les communications pourra donc décrypter tous les échanges passés s'il prend possession un jour de la clef privée associée à celle utilisée

pour le transfert de la clef commune.

S'appuyer sur le problème du logarithme discret permet en revanche de contourner cet écueil, en facilitant la création de schémas de confidentialité persistante parfaite (*perfect forward secrecy* en anglais), pour lesquels la révélation de secrets de long terme ne permet pas de rendre intelligibles des échanges passés [DOW92]. Par exemple, dans l'échange de Diffie-Hellman, les secrets a et b d'Alice et Bob sont éphémères, tout comme le secret commun g^{ab} qu'ils utilisent pour créer leur clef de session partagée. Comme celle-ci est temporaire, si un adversaire parvient à intercepter ultérieurement l'une d'entre elle, il ne pourra cependant décrypter que le message correspondant. Ainsi, un intrus qui pénétrerait l'ordinateur d'Alice ou celui de Bob ne serait pas capable de reconstituer les clefs des communications précédentes, et donc de décrypter les échanges passés. En pratique, il est aussi possible de combiner des secrets éphémères et à longs termes, c'est-à-dire des clefs privées, pour fournir des schémas offrant simultanément une confidentialité persistante parfaite, et de la vérification d'identité.

0.3.2 Diversité algorithmique

L'histoire de la cryptographie nous apprend qu'il n'est pas prudent de fonder l'intégralité de nos protocoles sur une unique hypothèse de sécurité, puisque la mise en défaut de celle-ci serait de nature à briser simultanément tous les systèmes existant. C'est pour cette raison que les travaux actuels de recherche se focalisent à la fois sur la diversité des cryptosystèmes, et sur la variété des problèmes sous-jacents ; l'important réside de ce fait dans la génération de candidats sérieux pour remplacer les cryptosystèmes déjà sur le marché, en cas d'effondrement de l'un ou l'autre des problèmes réputés difficiles. Les schémas reposant sur le problème du logarithme discret proposent ainsi une alternative à ceux dérivés directement de la méthode RSA ou d'autres algorithmes dont la sécurité se rattache à la difficulté de factoriser de grands entiers.

Pour être tout à fait honnête il est néanmoins important de remarquer que la factorisation comme le logarithme discret sont des problèmes dont la complexité s'écroulerait à l'avénement d'ordinateurs quantiques performants. Il est donc utile d'étudier des cryptosystèmes un cran plus éloignés encore, en s'appuyant sur les codes correcteurs d'erreurs, les fonctions de hachage, ou les réseaux, par exemple.

0.4 Contributions

0.4.1 Logarithme discret, algèbre linéaire presque creuse et malléabilité de l'entropie du Bitcoin

Comparé à la factorisation, le problème du logarithme discret offre une plus grande flexibilité, puisqu'il autorise de nombreuses possibilités pour le groupe G. En pratique, les protocoles s'appuient naturellement sur deux catégories de groupes : le groupe des points rationnels d'une courbe elliptique définie sur un corps fini, ou le groupe multiplicatif des éléments inversibles d'un corps fini. Dans ce dernier, qui constitue l'objet de cette thèse, la structure plus riche facilite la résolution du problème. L'article suivant propose un survol large du problème du logarithme discret :

1. [JOP14] **The Past, evolving Present and Future of Discrete Logarithm,**
   avec Antoine Joux et Andrew Odlyzko,
   Open Problems in Mathematical and Computational Science Book, Springer, 2014.

Soit $\mathbb{F}_{p^n}$ un corps fini de caractéristique première p et de degré d'extension n. Le problème qui nous intéresse consiste donc, étant donné un élément arbitraire $h \in \mathbb{F}_{p^n}$ non nul et un générateur g de $\mathbb{F}_{p^n}^*$, à retrouver un entier x tel que :

$$g^x = h \text{ modulo } |\mathbb{F}_{p^n}^*|.$$

Actuellement, deux familles d'algorithmes se distinguent pour résoudre ce problème, selon la taille relative de la caractéristique et du degré de l'extension.

Pour ceux dont la caractéristique peut être considérée comme petite – en un sens que nous détaillons plus loin – les algorithmes par représentation de Frobenius sont à privilégier. Les articles qui traitent ce cas de figure sont les suivants :

2. [JP16b] **Technical history of discrete logarithms in small characteristic finite fields. The road from subexponential to quasipolynomial complexity,**
   avec Antoine Joux,
   Journal of Designs, Codes and Cryptography, 2016.
3. [JP14] **Improving the Polynomial time Precomputation of Frobenius Representation Discrete Logarithm Algorithms : Simplified Setting for Small Characteristic Finite Fields,**
   avec Antoine Joux,
   Asiacrypt 2014.

Le premier [JP16b] propose de retracer l'historique technique des différentes méthodes qui ont permis d'obtenir une complexité dite quasipolynomiale, c'est-à-dire en $\log(p^n)$. Le second [JP14] présente quant à lui une variante simplifiée d'un algorithme par représentation de Frobenius. En plus de faire abstraction de détails techniques devenus inutiles, l'apport de nouvelles idées concernant la collecte des relations permet d'abaisser la complexité des phases polynomiales de l'algorithme de $O(q^7)$ à $O(q^6)$, où q est une puissance de p très inférieure à p^n qui correspond à la cardinalité du corps de base. En pratique, soulignons que ces phases, bien que polynomiales et non quasipolynomiales, demeuraient le frein majeur des différents records de calcul effectués jusqu'à lors. La simplification des paramètres comme nous la proposons permet de redescendre la complexité asymptotique d'un calcul de logarithmes discrets dans un corps fini général à celle connue jusqu'ici pour les extensions particulières de Kummer, ou de Kummer tordues. Nous illustrons ce résultat par un calcul de logarithme discret en caractéristique 3 : il s'agit de $\mathbb{F}_{3^{5*497}}$, un corps fini qui n'est pas l'extension particulière d'un autre sous-corps, comme l'était les corps cibles précédemment atteints.

Lorsque la caractéristique devient plus large, les meilleurs algorithmes reposent sur des variations du crible par corps de nombres (NFS, pour *Number Field Sieve* en anglais). Trois publications portent sur cet ensemble de corps finis :

4. [BP14] **The Multiple Number Field Sieve for Medium and High Characteristic Finite Fields,**
   avec Razvan Barbulescu,
   LMS Journal of Computation and Mathematics et présenté à *ANTS XI, 2014, Corée du Sud.*

5. [Pie15] **The Multiple Number Field Sieve with Conjugation and Generalized Joux-Lercier Methods,**
   Eurocrypt 2015.

6. [JP16a] **Nearly Sparse Linear Algebra, and Applications to Discrete Logarithms Computations,**
   avec Antoine Joux,
   Review Volume, *Contemporary Developments in Finite Fields and Applications, 2016 © World Scientific Publishing Company.*

Le premier [BP14] propose de n'utiliser non plus deux corps de nombres, comme dans le cas du crible NFS classique, mais un grand nombre. Cette idée permet d'améliorer les complexités asymptotiques des algorithmes aussi bien pour les corps finis de moyenne que de grande caractéristiques. Le deuxième article [Pie15], présente quant à lui une manière de combiner cette idée avec

une nouvelle sélection polynomiale apparue entre-temps. Le tout résulte une nouvelle fois en un gain asymptotique, applicable cependant pour les corps finis de moyenne caractéristique seulement. Pour ces corps qui incarnent en revanche le cas le plus difficile des trois catégories de corps finis considérés dans notre problème, nous montrons qu'il faut asymptotiquement accroitre la taille en bits des corps considérés de 6 % pour prétendre à un niveau de sécurité équivalent à celui que l'on pouvait atteindre avant la découverte de cet algorithme.

Le troisième et dernier article [JP16a] qui intervient pour cette double plage de caractéristiques a en réalité un impact plus large que le simple calcul de logarithme discret. Si l'algèbre linéaire, comme souvent, agit dans les soubassements de la cryptographie, la nature même des équations que l'on rencontre dans notre domaine présente en revanche quelques spécificités. Ainsi sommes-nous couramment confrontés au traitement de systèmes d'équations linéaires dont la majorité des entrées sont nulles, systèmes dont les matrices associées sont alors qualifiées de matrices creuses. [JP16a] élargit l'étude classique aux matrices presque creuses, caractérisées par la concaténation d'une matrice creuse et d'un plus ou moins grand nombre de colonnes denses, disons ∂. Si y est un vecteur fixé, nous cherchons donc à résoudre le système :

$$A.x = y$$

où A est une matrice presque creuse, éventuellement rectangulaire, dont la dimension la plus large est notée N.

Isoler les colonnes denses de la matrice A permet d'obtenir une complexité asymptotique en $O(\ell N^2) + \tilde{O}(\max(\partial, c)^{\omega-1} N)$, avec ℓ le nombre maximum d'entrées non nulles par ligne dans sa partie creuse, c le nombre de processeurs à disposition, et ω l'exposant le plus bas obtenu dans la complexité de la multiplication de deux matrices. Ceci abaisse alors la complexité actuelle du problème, résolue jusqu'ici par Block Wiedemann en $O((\ell+\partial)N^2)+\tilde{O}(c^{\omega-1}N)$ opérations. Par ailleurs, et contrairement aux algorithmes précédents, nous montrons sous des conditions précises de choix des vecteurs initiaux que notre méthode permet non seulement de produire une solution aléatoire du système, mais décrit une base complète de l'ensemble des solutions.

Un fait notable est le suivant : lorsque le nombre de colonnes denses est inférieur au nombre de processeurs, ces colonnes denses ne coûtent rien ! Cette remarque est utile dans la mesure où ce cas particulier se présente précisément pour le calcul de logarithmes dans des corps de moyenne ou grande caractéristiques. [JP16a] permet ainsi d'accélérer la résolution d'une des trois phases du crible par corps de nombres, qui consiste précisément en la recherche d'un élément non trivial du noyau d'une telle matrice, dont le

nombre de colonnes denses est en pratique toujours plus petit que le nombre de processeurs dont on peut disposer.

Un quatrième article traite aussi du sujet dans le cas particulier où la caractéristique (moyenne ou grande) peut de surcroit être considérée comme creuse. Néanmoins, ce résultat n'apparait pas dans cette thèse car il a déjà fait l'objet de mon manuscrit de master.

7. [JP13] **The Special Number Field Sieve in Finite Fields, Application to Pairing-Friendly Constructions,**
avec Antoine Joux,
Pairing 2013.

De la même manière, j'ai choisi de ne pas traiter un résultat récent, qui, quoiqu'écrit pendant ma thèse et toujours en cryptographie, se situe dans un cadre trop lointain des logarithmes discrets. Par souci de cohérence il n'est donc donné aucun détail à son sujet dans le corps de ce manuscrit. A titre informatif tout de même, il s'agit de l'article suivant :

8. [PW16] **Malleability of the Blockchain's Entropy,**
avec Benjamin Wesolowski,
Présenté à *ArcticCrypt Conference 2016,* en cours de publication.

La génération de nombres aléatoires publics et fiables est nécessaire à de nombreux protocoles de cryptographie. Une idée suggérée parmi d'autres consisterait à s'appuyer sur l'imprédictabilité inhérente des chaines de blocs pour bâtir une source publique d'aléa. C'est ici qu'interviennent les monnaies électroniques décentralisées, comme le Bitcoin. En effet l'entropie de la chaine du registre public sous-jacent au Bitcoin a par exemple été utilisée pour créer des lotteries, puis proposée par la suite pour d'autres applications variant de la rédaction de contrats intelligents à l'audit d'élections. La malléabilité de l'entropie des chaines de blocs montre cependant qu'un adversaire peut manipuler ces nombres aléatoires, même s'il celui-ci est tenu par des contraintes financières étroites et qu'il ne dispose que d'une puissance de calcul limitée. Dans [PW16] nous analysons la probabilité de succès d'un adversaire malveillant en fonction de ces paramètres, pour conclure qu'une telle génération de nombres aléatoires publics est à rejeter tout à fait. En passant, cette analyse propose une nouvelle illustration des mots de Dyck tout en suscitant une généralisation de ceux-ci.

0.4.2 Organisation du manuscrit

La première partie de cette thèse s'attache à donner des résultats généraux sur le problème du logarithme discret. Nous y étudions ainsi les classes

de complexité auxquelles le problème appartient, détaillons les algorithmes classiques pour résoudre le problème en l'absence d'indication essentielle sur le groupe, puis évoquons le modèle du groupe générique, le tout au sein du chapitre 1. Le chapitre 2 détaille ensuite l'articulation de la *méthode de calcul d'indice*, à la base de tous les algorithmes actuels pour résoudre le problème du logarithme discret dans les corps finis ou sur des courbes algébriques. Cette partie est issue essentiellement de l'article de survol [JOP14].

Une deuxième partie est consacrée au problème du logarithme discret dans les corps finis de petite caractéristique. Le chapitre 3 reprend ainsi les avancées techniques qui ont permis la transformation des algorithmes sous-exponentiels en des algorithmes quasipolynomiaux. Il est issu en majorité de l'article [JP16b]. Sur ces fondations, nous reconstruisons au chapitre 4 l'algorithme de [JP14] qui permet d'obtenir une amélioration de la complexité des phases polynomiales, c'est-à-dire de récolte des relations, d'algèbre linéaire et d'extension de la base de facteurs. Il s'agit de la méthode la plus efficace à ce jour pour calculer des logarithmes discrets sur cette plage de corps finis.

Enfin, dans une troisième et dernière partie nous nous attachons aux corps finis de moyenne et grande caractéristiques. Le chapitre 5 combine les résultats des articles [BP14] et [Pie15] qui constituent ainsi les meilleurs algorithmes actuels pour résoudre le problème, en terme de complexité asymptotique, et lorsque l'on ne peut rien dire sur la caractéristique du corps, ni sur son degré d'extension. Le chapitre 6 présente enfin un algorithme pour résoudre des problèmes d'algèbre linéaire sur des matrices que l'on qualifiera de presque creuses. Une des applications de cette méthode proposée dans [JP16a] consiste à améliorer en pratique la phase d'algèbre linéaire des calculs de logarithmes discrets dans ces corps finis de moyenne et grande caractéristiques.

Première partie

Algorithmes de calcul de logarithmes discrets

Chapitre 1

Logarithme et groupes généraux

Sommaire

L'objet de ce premier chapitre consiste en la présentation de résultats généraux qui s'appliquent au problème du logarithme discret quelque soit le groupe considéré. Il n'est ainsi requis qu'un tout petit nombre de propriétés concernant celui-ci : nous supposons simplement que la représentation du groupe est compacte, puis que la loi associée est explicitement donnée et que l'on détient un algorithme efficace pour effectuer les opérations de cette loi. Un tel groupe sera dit *général*. En sus, nous serons souvent amenés à supposer l'ordre du groupe connu, ainsi, éventuellement, que sa factorisation – mais ceci sera toujours explicitement notifié.

Cette dernière hypothèse n'est pas très contraignante : pour de très nombreux groupes utilisés en pratique, la factorisation de l'ordre s'obtient facilement. L'exemple le plus classique se compose du cas du groupe des points rationnels d'une courbe elliptique, groupe pour lequel l'ordre peut être efficacement obtenu à l'aide des algorithmes récents de comptage de points. Par ailleurs, les cryptographes chargés de la conception de ces systèmes choisissent fréquemment des courbes pour lesquelles l'ordre consiste en un petit multiple d'un nombre premier. Il est alors aisé de comprendre que la factorisation de la cardinalité du groupe devient accessible, ce qui rend réaliste l'hypothèse classique faite selon laquelle cette factorisation est une donnée initiale du problème.

De manière amusante nous montrons aussi au paragraphe 1.1.3 que, si l'on sait calculer efficacement des logarithmes discrets au sein d'un groupe G, il est possible de retrouver la cardinalité de celui-ci.

Plan du chapitre. Nous commençons au paragraphe 1.1 par des résultats de complexités généraux qui permettent de comprendre la difficulté du problème du logarithme discret en étudiant ses liens avec certaines classes de complexités théoriques usuelles. Nous travaillons ensuite avec un groupe G général dont nous supposons connue la factorisation de l'ordre. Ceci autorise la démonstration du fait suivant : le calcul de logarithmes discrets dans G n'est pas plus difficile que celui de logarithmes discrets dans chacun des sous-groupes de G d'ordre premier. Il s'agit du résultat de Pohlig–Hellman [PH78] qui propose une méthode constructive pour calculer des logarithmes discrets dans le groupe tout entier à partir d'un petit nombre de calculs de logarithmes dans les sous-groupes de cardinalité première. Cet algorithme est décrit au paragraphe 1.2. Nous détaillons au paragraphe 1.3.2 l'algorithme Rho de Pollard [Pol78] qui permet de résoudre le problème dans G en $O(\sqrt{|G|})$ opérations. En combinant les algorithmes de Pohlig–Hellman et Rho de Pollard il est par conséquent possible d'élaborer une méthode pour calculer des logarithmes discrets en $O(\sqrt{p})$ opérations, où p est le plus grand facteur premier

intervenant dans la factorisation de l'ordre d'un groupe général.

Enfin, nous discutons au paragraphe 1.4 du souci de passage à l'échelle, c'est-à-dire de la possibilité de calculer un grand nombre de logarithmes discrets indépendants au sein du même groupe général, en amortissant une partie des coûts de calcul. Nous clôturons ce chapitre par la présentation du modèle du groupe générique au paragraphe 1.5 proposé par Shoup dans [Sho97b], modèle pour lequel on connait des algorithmes optimaux.

1.1 Classes de complexité

Pour décrire le niveau exact de difficulté d'un problème calculatoire, l'approche classique consiste à étudier les classes de complexité auxquelles il appartient. A cette fin, nous considérons traditionnellement des problèmes de décision, c'est-à-dire des problèmes qui peuvent être formulés de telle sorte que l'on puisse répondre simplement par oui ou par non. Le problème du logarithme discret tel quel n'est pas un problème de décision. Aussi, avant de déterminer les classes de complexités qui lui sont reliées, la première étape repose sur l'introduction du problème de décision qui lui sera le plus proche, et dont la difficulté sera essentiellement équivalente au calcul de logarithmes discrets. Si plusieurs choix sont possibles, notre attention se portera sur le problème suivant :

Problème 1.1.1 (Log Range Decision). **Etant donné un groupe cyclique** G **et un triplet** (g, h, B) :

— Renvoyer **OUI** s'il existe $x \in [\![0, B]\!]$ tel que $h = g^x$.
— Sinon renvoyer **NON**.

En effet, un algorithme ou un oracle qui résout ce problème peu être utilisé pour calculer des logarithmes discrets via une recherche dichotomique. Ceci demande alors un nombre logarithmique d'appels à *Log Range Decision.* En d'autres termes, le nombre d'appels à l'oracle est polynomial en la taille en bits de l'entrée. Réciproquement, il est clair que la connaissance d'un algorithme de résolution du problème du logarithme discret permet de répondre à la question posée par ce problème de décision. Par conséquent, la difficulté du problème *Log Range Decision* est essentiellement la même que celle du problème du logarithme discret lui-même. Nous donnons dans la suite de cette section quelques résultats d'appartenance à des classes de complexité bien connues.

1.1.1 Log Range Decision appartient à NP ∩ co-NP

Rappelons qu'un problème est dans **NP** s'il peut être décidé sur une machine de Turing non-déterministe en temps polynomial par rapport à la taille de l'entrée, c'est-à-dire qu'il existe un témoin qui permet de décider en temps polynomial si une solution donnée par exemple par un oracle convient à ce problème. La classe **co-NP** représente la classe complémentaire de la classe **NP**, au sens de la théorie de la complexité. Dit autrement, un problème est dans **co-NP** s'il existe un certificat qui permet de vérifier en temps polynomial qu'une instance qui ne répond pas au problème donnera la réponse NON.

Pour montrer que le problème est dans **NP**, supposons qu'il existe un entier $x \in [\![0, B]\!]$ tel que $h = g^x$, alors cet entier x lui-même est un témoin de ce fait, qui se teste facilement en temps polynomial.

Par ailleurs, lorsque g est un générateur de G et que le groupe en question est d'ordre connu, donner un logarithme discret de h en base g est aussi un témoin satisfaisant pour prouver que la réponse est NON. Dans ce cas particulier, le problème appartient donc à la classe **co-NP**. La situation est cependant plus complexe dans le cas général. En effet, pour poursuivre, nous avons besoin d'un générateur g_0 de G, ainsi que de la connaissance de l'ordre $|G|$ du groupe, et de sa factorisation. Au demeurant, puisqu'il y a $\varphi(|G|)$ générateurs de G, c'est à dire un grand nombre, g_0 se trouve aisément par test de candidats pris aléatoirement dans l'ensemble. Ce générateur g_0 étant connu, la connaissance du logarithme discret de g et celui de h en base g_0 suffit alors à déterminer si h appartient au sous-groupe généré par g ou non, et, au besoin, à montrer qu'aucun des logarithmes discrets de h en base g n'appartient à l'intervalle $[\![0, \cdots, B]\!]$. Nous en déduisons que le problème *Log Range Decision* se situe bien dans **co-NP**, même dans le cas général où g n'est pas un générateur du groupe.

1.1.2 Log Range Decision appartient à BQP

Un autre résultat de complexité théorique essentiel concernant le logarithme discret est avancé par Shor en 1997, lorsqu'il propose un algorithme quantique efficace pour résoudre le problème au sein de n'importe quel groupe général, sous la seule hypothèse donc que les opérations du groupe puissent se faire de manière efficace. Cette méthode que l'on peut retrouver dans l'article [Sho97a] et qui s'appuie sur les transformations de Fourier quantiques prouve ainsi que le problème appartient à la classe de complexité **BQP**. Cette classe, dont l'acronyme provient de l'anglais *Bounded-error Quantum Polynomial time*, correspond à l'ensemble des problèmes pour lesquels il existe un algorithme qui permet de répondre au problème, en temps polynomial et sur

une machine quantique, avec une probabilité d'erreur bornée. Lorsque l'on souhaite donner une définition formelle de **BQP**, une probabilité d'erreur inférieure à 1/3 est communément admise.

1.1.3 Retrouver l'ordre $|G|$ du groupe à l'aide d'un calcul de logarithme discret

Lorsque l'on considère le problème du logarithme discret, on suppose souvent, même implicitement, que la cardinalité du groupe est connue a priori. La raison principale repose simplement sur le fait que l'ordre, comme la factorisation de celui-ci, sont facilement calculables en pratique pour les groupes utilisés. Néanmoins, un argument intéressant consiste à relever le fait suivant : lorsque le calcul d'un logarithme discret est réalisable au sein d'un groupe dont on ne connait pas l'ordre, retrouver a posteriori l'ordre de celui-ci devient alors envisageable.

En effet, supposons que l'on ne connaisse qu'un ordre de grandeur de cette cardinalité. Par exemple, nous pouvons faire l'hypothèse d'avoir obtenue une information sur le nombre de bits de $|G|$, c'est-à-dire que nous partons de l'entier n tel que $|G| \in [2^{n-1}, 2^n - 1[$.

Deux cas se présentent maintenant. Soit l'algorithme dont on dispose nous donne une valeur normalisée entre 0 et $|G| - 1$. Dans ce contexte, à partir d'un générateur g de G, nous déduisons que l'ordre du groupe cherché divise $2^n - \log_g(g^{2^n})$. Cet entier ne peut être nul car $2^n > |G| - 1$. Par ailleurs, on constate qu'il n'est pas non plus un multiple de $|G|$ différent de celui-ci : en effet, de $2^{n-1} \leq |G|$ il vient $2^n \leq 2|G|$, donc $2^n - \log_g(g^{2^n})$ est clairement plus petit que le plus petit des multiples propres de $|G|$. Aussi, de l'égalité :

$$|G| = 2^n - \log_g(g^{2^n}),$$

nous sommes en mesure de calculer l'ordre de G par un simple calcul de logarithme discret.

Le second cas de figure – lorsque l'algorithme est autorisé à renvoyer n'importe quel logarithme discret, certes correct, mais non normalisé – est plus complexe. Prenons b un entier aléatoire entre 0 et un certain multiple de $2^n - 1$, par exemple $10 \cdot (2^n - 1)$. Notons a la valeur obtenue lorsque l'on calcule le logarithme discret de g^b. Comme $g^b = g^a$ dans le groupe, nous en déduisons comme précédemment que l'écart $|b - a|$ est un multiple de l'ordre cherché – éventuellement nul. Si pour ce b choisi l'entier a renvoyé par l'algorithme lui est égal, alors nous recommençons le procédé en initialisant par un autre choix de b. Supposons alors que $a \neq b$. Comme $|b - a|$ est un des dix multiples parmi $|G|, \cdots, 10 \cdot |G|$ il est aisé de retrouver lequel parmi les dix

diviseurs potentiels de $|b-a|$ est celui dans l'intervalle initial $[2^{n-1}, 2^n - 1[$. Ce dernier correspond alors à l'ordre du groupe.

Enfin, même si nous n'avons aucune information sur l'ordre de grandeur de la cardinalité de G, il est possible de trouver celle-ci en procédant comme précédemment, par choix d'un entier b dans un intervalle large. Nous obtenons alors encore une fois un multiple de l'ordre du groupe. Répéter ce procédé pour trouver d'autres multiples puis calculer un pgcd des différents candidats nous donne au final l'ordre exact de G.

1.1.4 Difficulté du cas moyen et auto-réduction aléatoire

Lorsque l'on s'appuie sur des problèmes réputés difficiles pour construire des cryptosystèmes, il est crucial d'être certain que des instances prises aléatoirement seront, en pratique, réellement difficiles. Dès lors nous saisissons l'écart qui s'opère avec la définition classique en théorie de la complexité de la difficulté d'un problème. En effet, un problème qui admet seulement quelques instances difficiles en cryptographie peut se révéler indigne d'intérêt : nous avons besoin d'articuler nos protocoles autour de problèmes dont la difficulté n'est pas seulement relative aux pires cas, mais aussi aux cas moyens – l'idéal étant que cette difficulté se révèle même intrinsèque à la plupart des cas.

A ce sujet, le problème du logarithme discret détient une propriété amusante : l'auto-réduction aléatoire (ou *random self-reducibility* en anglais), évoquée pour la première fois en 1989 [AFK89]. Cette propriété traduit le fait que chaque instance du problème du logarithme discret peut être randomisée en une instance purement aléatoire, comme nous le détaillons quelques lignes plus bas. Par conséquent, si le problème est facile à résoudre dans le cas moyen, il l'est aussi dans le pire cas. La contraposée nous indique que, s'il existe des instances difficiles du problème du logarithme discret dans un certain groupe G, alors le problème du logarithme discret est lui-même difficile pour tout élément aléatoire de G.

La réduction en tant que telle fonctionne comme suit : supposons que nous soyons en communication avec un oracle qui sache résoudre le problème du logarithme discret dans G pour des instances aléatoires. Soit h une instance fixée au préalable. On souhaite trouver un entier x tel que $g^x = h$. Commençons par considérer un entier r modulo $|G|$ choisi de manière uniformément aléatoire, puis posons $z = hg^r$. L'élément z du groupe suit alors une distribution aléatoire uniforme dans G. Si l'oracle peut retrouver le logarithme discret $\log_g(z)$ de cet élément, alors nous pouvons calculer x à l'aide de la relation $x \equiv \log_g(z) - r$ modulo $|G|$.

1.2 Algorithme de Pohlig–Hellman

Les deux sections qui suivent comprennent la présentation d'algorithmes généraux pour la résolution du problème du logarithme discret. Soit G un groupe, g un générateur et h un élément arbitraire pour lequel nous souhaitons retrouver un logarithme x. Nous supposons par ailleurs que la factorisation de l'ordre du groupe est connue et donnée par :

$$|G| = \prod_{p_i \text{ premier tel que } p_i||G|} {p_i}^{e_i}.$$

L'algorithme de Pohlig-Hellman permet de reduire le problème du logarithme discret dans G aux problèmes du logarithme discret dans les groupes cycliques d'ordre premier p_i. La méthode s'articule autour des deux phases qui suivent.

1.2.1 Réduction aux ordres des puissances de nombres premiers

Tout d'abord nous réduisons le problème du logarithme discret dans G aux problèmes du logarithme discret dans des groupes d'ordre les puissances des nombres premiers p_i impliquées dans la factorisation de $|G|$. Pour chaque p_i nous posons :

$$|G_i| = \frac{n}{{p_i}^{e_i}}, \qquad g_i = g^{|G_i|} \quad \text{et} \quad h_i = h^{|G_i|}.$$

Ainsi, ${g_i}^{{p_i}^{e_i}} = g^n = 1$ et l'ordre de g_i est exactement ${p_i}^{e_i}$. Par ailleurs nous obtenons $g_i^x = g^{|G_i|x} = h^{|G_i|} = h_i$. Ainsi h_i appartient au sous-groupe d'ordre $p_i^{e_i}$ engendré par g_i. Plus précisément, h_i peut être considéré comme la projection de l'élément dont on souhaite trouver un logarithme discret sur le sous-groupe engendré par g_i. Sans originalité, appelons x_i le logarithme discret de h_i en base g_i. Ceci nous permet d'écrire la relation :

$$x \equiv x_i \mod p_i^{e_i}.$$

Puisque les $p_i^{e_i}$ sont deux à deux premiers entre eux, si nous connaissons tous les x_i, une application simple du théorème des restes chinois nous permet de retrouver l'entier x.

1.2.2 Réduction aux ordres premiers

Une réduction similaire autorise ensuite à réduire le problème du logarithme discret dans un groupe d'ordre une puissance d'un nombre premier

au problème du logarithme discret dans le sous-groupe d'ordre ce nombre premier précisément.

Soit G un groupe cyclique de cardinal p^e une puissance d'un nombre premier p. Considérons g un générateur de G et a un élément quelconque de G. Nous souhaitons retrouver le plus petit entier x tel que $a = g^x$. Comme $x < p^e$, nous pouvons noter son écriture en base p :

$$x = \sum_{j=0}^{e-1} x_j p^j$$

où $0 \leqslant x_j < p$ pour tout entier j dans l'intervalle de la somme. Montrons alors que chaque coefficient x_j peut être vu comme un logarithme discret dans un certain groupe d'ordre p. Pour commencer, multiplions par p^{e-1} l'écriture ci-dessus. Nous obtenons $p^{e-1}x = p^{e-1}x_0 + p^e \sum_{j=1}^{e-1} x_j p^{j-1}$. Or en élevant à la puissance p^{e-1} l'équation $a = g^x$, il vient $a^{p^{e-1}} = g^{p^{e-1}x}$ et par conséquent :

$$a^{p^{e-1}} = g^{p^{e-1}x_0},$$

car l'ordre du groupe est précisément p^e. Cette dernière équation permet d'interpréter x_0 comme le logarithme discret de $a^{p^{e-1}}$ dans le sous-groupe d'ordre p engendré par $g^{p^{e-1}}$. En effet, $g^{p^{e-1}}$ est bien d'ordre p, par définition de g.

Une récurrence sur les coefficients suivants nous permet de conclure. Autrement dit, supposons que $x_0, \cdots, x_{i-1}$ aient été déterminés comme des logarithmes discrets. Nous écrivons alors $ag^{-\sum_{j=0}^{i-1} x_j p^j} = g^{\sum_{j=i}^{e-1} x_j p^j}$ qui se traduit, une fois élevé à la puissance p^{e-i-1}, par :

$$(ag^{-\sum_{j=0}^{i-1} x_j p^j})^{p^{e-i-1}} = g^{p^{e-1}x_i}.$$

Nous pouvons interpréter x_i de nouveau comme le logarithme discret d'un élément – ici le membre de gauche – dans le même sous-groupe d'ordre p engendré par $g^{p^{e-1}}$. Ainsi, le calcul de tous les x_j pour j variant de 0 à $e-1$, qui ne sont autres que des calculs de logarithmes discrets dans le sous-groupe d'ordre p, permettent de reconstituer le logarithme x cherché.

En bref, l'idée essentielle qu'il est important de retenir est la suivante : calculer des logarithmes discrets dans un groupe général G n'est pas plus difficile que de calculer des logarithmes discrets dans chacun des sous-groupes d'ordre premier.

1.3 Calcul de logarithme discret dans un groupe général en la racine de l'ordre du groupe

Diverses méthodes permettent de calculer des logarithmes discrets dans un groupe général G en $O\left(\sqrt{|G|}\right)$ opérations. La plus connue d'entres-elles, qui est aussi probablement la plus simple à aborder, a été présentée par Shanks sous le nom de méthode des pas de bébés – pas de géants (*Baby Step – Giant Step technique* en anglais).

1.3.1 Pas de bébés – pas de géants

Notons encore G, g, h nos paramètres, et x le logarithme discret recherché. Eventuellement, g peut ne pas être un générateur du groupe entier, mais nous supposons alors que h est bel et bien dans le sous-groupe qu'il engendre. Considérons la division euclidienne de x par $s = \lceil\sqrt{|G|}\rceil$. Si q et r sont tels que $x = qs + r$ avec $0 \leqslant r$ et $q < s$ – ce qui est possible étant donnée la taille relative de s par rapport à $|G|$ – alors il est clair que retrouver un entier x revient exactement à exhiber une paire (q, r). Remarquons tout d'abord que nous avons l'égalité :

$$(g^s)^q = (g^{sq}g^r)g^{-r} = g^{sq+r}g^{-r} = hg^{-r}. \tag{1.1}$$

Nous créons dans un premier temps la liste :

$$\text{Bébé} = \{(hg^{-r}, r) | 0 \leqslant r < s\}$$

que l'on peut appeler la liste des pas de bébés, puisque l'on multiplie à chaque étape l'élément courant par l'inverse de g, qui peut être considéré comme petit. Si, par chance, il existe un couple $(1, r')$ dans cette liste, alors nous avons obtenu un entier pour lequel $hg^{-r'} = 1$. Par conséquent, $x = r'$. Dans le cas contraire nous créons une seconde liste :

$$\text{Géant} = \{((g^s)^q, q) | 0 \leqslant q < s\}$$

celle des pas de géants – cette fois-ci l'élément courant est multiplié par un gros élément, g^s. L'idée consiste ensuite à trier les deux listes pour mettre en évidence une collision sur les premiers éléments de chaque paire. Lorsque nous trouvons ainsi une paire de paire $(hg^{-r}, r) \in$ Bébé et $((g^s)^q, q) \in$ Géant telle que l'équation (1.1) soit satisfaite, nous en déduisons directement q et r, puis x.

La recherche d'une égalité une fois les deux listes de s entrées triées peut s'effectuer en temps linéaire, en supposant que la représentation des éléments

est suffisamment compacte. Aussi, le temps d'exécution de l'algorithme précédent est dominé par l'arithmétique nécessaire à la création des deux listes, auquel s'ajoute le temps passé à les trier. Cet algorithme déterministe résout le problème du logarithme discret en $\widetilde{O}\left(\sqrt{|G|}\right)$ opérations. Nous rappelons que la notation $\widetilde{O}(n)$ est un raccourci qui est similaire à l'écriture de $O(n\log^{\alpha} n)$ pour une valeur arbitraire de α.

1.3.2 L'algorithme Rho de Pollard

Cet algorithme s'exécute en un temps comparable à celui de Shanks, c'est-à-dire qu'il nécessite $O\left(\sqrt{|G|}\right)$ opérations, mais possède l'avantage considérable de n'utiliser en pratique que peu de mémoire. Contrairement à la méthode de Shanks en revanche, cet algorithme n'est pas déterministe, mais probabiliste. Proposé par John M. Pollard en 1978 [Pol78], il peut se résumer de la façon suivante.

Supposons que le groupe G possède une partition en trois sous-ensembles de tailles équivalentes que nous noterons A_1, A_2 et A_3. Soit f l'application définie par :

$$f(b) = \begin{cases} gb & \text{si } b \in A_1 \\ b^2 & \text{si } b \in A_2 \\ hb & \text{si } b \in A_3 \end{cases}$$

Choisissons maintenant un entier x_0 aléatoire dans $(\mathbb{Z}/|G|\mathbb{Z})^*$. A partir de $b_0 = g^{x_0}$ nous pouvons calculer la suite récurrente $b_{i+1} = f(b_i)$. L'algorithme repose alors sur deux constats. Le premier consiste à remarquer que pour chaque indice i nous pouvons réécrire b_i le i-ième élément de la suite comme :

$$b_i = g^{x_i} h^{y_i} \tag{1.2}$$

où $(x_i)_i$ et $(y_i)_i$ sont fixés par un choix initial de x_0, $y_0 = 0$ et :

$$x_{i+1} = \begin{cases} x_i + 1 \mod |G| & \text{si } b_i \in A_1 \\ 2x_i \mod |G| & \text{si } b_i \in A_2 \\ x_i & \text{si } b_i \in A_3 \end{cases} \quad \text{et} \quad y_{i+1} = \begin{cases} y_i & \text{si } b_i \in A_1 \\ 2y_i \mod |G| & \text{si } b_i \in A_2 \\ y_i + 1 \mod |G| & \text{si } b_i \in A_3 \end{cases}$$

En outre, puisque nous évaluons les termes d'une suite infinie dans un groupe fini, il existe deux indices $i \geqslant 0$ et $k \geqslant 1$ menant à une collision de la forme $b_i = b_{i+k}$. En pratique néanmoins, par souci de gain de mémoire, on cherchera plutôt une égalité de la forme plus particulière $b_i = b_{2i}$. Pour ces deux indices l'équation (1.2) nous donne alors :

$$g^{x_i} h^{y_i} = g^{x_{i+k}} h^{y_{i+k}}$$

qui mène à une équation linéaire en $\log_g(h)$:

$$x_i - x_{i+k} \equiv \log_g(h)(y_{i+k} - y_i) \mod |G|.$$

Dès lors que nous parvenons à inverser $y_{i+k} - y_i$ modulo $|G|$ nous sommes en mesure de retrouver un logarithme discret de h. En revanche, si $y_{i+k} - y_i$ n'est pas inversible, il nous faut alors nous remémorer le fait que l'algorithme Rho de Pollard que nous détaillons en ce moment même est utilisé le plus fréquemment comme un sous-algorithme de Pohlig–Hellman. Ceci implique en particulier que $|G|$ est un entier premier. Par conséquent, la seule option possible consiste à recommencer la recherche de collision avec une instance initiale différente, par exemple en choisissant un entier x_0 différent du premier testé. C'est précisément la possibilité de ce cas de figure, lorsque $y_{i+k} - y_i$ n'est pas inversible modulo $|G|$, qui donne à cet algorithme sa dimension probabiliste. Néanmoins, avec un $|G| = p$ premier, la probabilité d'échec au premier test est égale à $1/p$.

L'algorithme Rho de Pollard peut être implémenté de telle sorte qu'il requière en mémoire le stockage de $O(1)$ éléments seulement, pour $O\left(\sqrt{|G|}\right)$ opérations. Diverses améliorations pratiques de ce résultat ont été développées depuis sa présentation initiale en 1978 ; le lecteur intéressé pourra découvrir dans les articles [Tes00, BLS11, CHK12] certaines des plus récentes.

En pratique, comme nous l'avons suggéré, le calcul de logarithmes discrets dans un groupe général utilisera une combinaison des algorithmes de Pohlig-Hellman et Rho de Pollard. Selon l'architecture sous-jacente des machines à disposition, il existe des alternatives à l'algorithme de Pollard qui peuvent être plus appropriées – nous donnons plus de détails au paragraphe 1.4. Cependant, la complexité globale de tous ces algorithmes reste en $O\left(\sqrt{p}\right)$, où p désigne le plus grand nombre premier divisant la cardinalité du groupe cible. Mieux, nous détaillerons au paragraphe 1.5 le résultat qui stipule qu'un algorithme qui résout le problème du logarithme discret dans un groupe générique[1] ne peut abaisser cette complexité.

1.4 Passage à l'échelle des algorithmes de logarithmes discrets pour les groupes généraux

En informatique matérielle et logicielle, le passage à l'échelle désigne la capacité d'un programme à s'adapter à un changement d'ordre de grandeur du nombre d'instances à traiter, en particulier sa capacité à maintenir ses

1. Un algorithme qui fonctionne pour un groupe générique s'adapte au cas d'un groupe général.

fonctionnalités et ses performances en cas de forte demande. D'un point de vue théorique, un algorithme en $O(\sqrt{|G|})$ qui utilise une quantité de mémoire constante constitue déjà une solution très acceptable. Cependant, pour des objectifs pratiques, il est utile de savoir comment l'on peut procéder si l'on souhaite calculer plusieurs logarithmes discrets, et si de tels calculs peuvent être distribués, que ce soit sur une machine parallélisable, ou sur un réseau d'ordinateurs indépendants. En effet, cette notion de passage à l'échelle détermine souvent si le calcul est réaliste ou non.

Dans ce contexte, il est alors bénéfique de remplacer les algorithmes qui recherchent des cycles, comme Rho de Pollard que nous venons d'étudier, par des algorithmes basés sur la technique du point distingué. Cette idée a initialement été suggérée par Rivest, puis reprise en 1989 par Quisquater et Delescaille [QD89] pour exhiber des collisions dans le DES. L'étude détaillée de van Oorschot et Wiener [vOW99] dix ans plus tard montre comment cette technique peut être employée pour tirer partie du parallélisme dans la recherche de collisions.

Quelle est donc, alors, cette technique du point distingué ? L'idée principale repose sur la construction d'une chaine de calculs, qui commence à partir d'une valeur aléatoire x_0 et se poursuit par itération d'une application f préalablement définie. Plus précisément, nous calculons de manière itérative les $x_{i+1} = f(x_i)$, c'est-à-dire une longue chaine des successeurs du point initial par cette application, jusqu'à l'émergence d'un point x_N satisfaisant une propriété qui le distingue des autres. Un exemple parmi d'autre peut consister à affirmer que ce point est distingué si sa représentation commence avec un nombre prédéfini de bits à 0. De cette longue chaine découverte nous ne gardons pourtant en mémoire que la valeur du triplet (x_0, x_N, N). Dans la suite, l'idée est similaire à celle de l'algorithme Rho de Pollard : afin de retrouver le logarithme discret cherché, nous cherchons à exhiber une collision.

Avec cette technique de création de chaines itératives, n'importe quelle collision entre deux chaines assure que celles-ci se termineront sur le même point distingué. Réciproquement, étant donné deux chaines achevées par le même point distingué, si nous les calculons de nouveau l'une et l'autre à partir de leur point initial respectif, et si nous prenons correctement en compte leur différence de longueur [2], nous parvenons à retrouver une collision explicite. Enfin, puisque les pré-calculs des chaines sont indépendants les uns des autres, il est très aisé de les distribuer sur un grand nombre d'ordinateurs distincts.

2. De manière plus explicite : si Δ désigne l'écart positif de longueur entre les deux chaines, nous comparons chaque élément d'indice $i + \Delta$ de la chaine la plus longue, avec l'élément d'indice i de la chaine la plus courte, pour i variant de 0 à la longueur de la plus courte des deux chaines.

Récemment, l'utilisation d'une légère variante de cette technique [FJM14] a permis de montrer que l'on pouvait calculer L logarithmes discrets indépendants dans le même groupe G en $O(\sqrt{L|G|})$ opérations, plutôt qu'en $O(L\sqrt{|G|})$. Un résultat similaire était déjà connu sous la condition de ne pas vouloir retrouver trop de logarithmes discrets [KS01], c'est-à-dire sous l'hypothèse $L \leq O(|G|^{1/4})$.

1.5 Le modèle du groupe générique

1.5.1 Une démonstration d'optimalité ?

En 1997, Shoup [Sho97b] introduit un cadre théorique propice à l'étude de la complexité des algorithmes généraux : le modèle du groupe générique. Dans ce modèle, il montre que n'importe quel algorithme doit effectuer $\Omega(\sqrt{p})$ opérations de groupe, où p est le plus grand nombre premier qui divise la cardinalité du groupe. Puisque cette borne inférieure correspond, à des facteurs logarithmiques près, à la borne supérieure connue et traitée précédemment, le modèle du groupe générique souligne que les algorithmes généraux déjà créés sont quasi-optimaux pour résoudre le problème du logarithme discret.

Les travaux de Shoup s'inscrivent dans la continuité de ceux de Babai et Szemeredi [BS84], puis Nechaev [Nec94]. En 1984, Babai et Szemeredi se placent au sein d'un modèle de *boite noire* pour lequel ils exhibent pour la première fois des bornes inférieures pour le calcul de logarithmes discrets. Dans leur modèle, et pour des groupes cycliques, leur résultat apporte une borne inférieure en $\Omega(p)$ où p est le plus grand diviseur premier qui divise la cardinalité du groupe. Dix ans plus tard, Nechaev considère le modèle dans lequel un algorithme est autorisé à effectuer lui-même les opérations de groupe et les tests d'égalité, mais aucune autre opération ne lui est accessible (la notion d'encodage n'intervient pas dans ce modèle). Ici, les tests d'égalité sont gratuits, et seul le nombre d'opérations de groupe est comptabilisé. Nechaev prouve ainsi une borne inférieure en $\Omega(\sqrt{p})$ pour le nombre d'opérations de groupe.

Shoup étend ce dernier résultat à une classe plus large et plus naturelle d'algorithmes : tous les algorithmes couverts par le modèle de Nechaev le sont aussi par celui de Shoup. Mieux, il prend en compte des algorithmes qui ne l'était pas dans le modèle précédent, comme l'algorithme de Pollard. Dans le modèle du groupe générique de Shoup, les éléments du groupe sont identifiés de manière unique certes mais par des encodages arbitraires. Par conséquent, aucune propriété spécifique de l'encodage ne peut être exploitée, et les éléments du groupe ne peuvent être manipulés que par un oracle. C'est

ce dernier qui permet alors l'accès aux opérations du groupe. Notons aussi qu'un algorithme qui fonctionne dans le modèle générique – on parle alors d'algorithme générique, ou d'algorithme travaillant sur un *groupe générique* – fonctionne aussi dans le cas général de n'importe quel groupe efficacement représentable, que nous avons dénommé jusqu'ici *groupe général.*

1.5.2 Faiblesses du modèle

Néanmoins, une critique courante de ce modèle repose sur la faiblesse qu'il partage avec, par exemple, le modèle de l'oracle aléatoire, considéré avec suspicion par de nombreux cryptographes. Plus précisément, il est dérangeant de relever qu'il existe au sein de ces deux modèles des protocoles dont la sécurité est satisfaisante en théorie, mais qui ne peuvent être instanciés de façon correcte. Les articles [Den02] et [CGH00] apporteront au lecteur curieux des illustrations de systèmes sûrs dans le modèle générique, mais qui cessent de l'être lorsque l'on substitue le groupe générique par un groupe général.

Par ailleurs, le modèle du groupe générique entraine des difficultés techniques supplémentaires ; pour un groupe générique d'ordre composé, le calcul du symbole de Jacobi – et donc celui du symbole de Legendre – devient par exemple difficile.

Chapitre 2

La méthode du calcul d'indice

Sommaire

Les résultats d'optimalité obtenus dans le modèle du groupe générique ne tiennent plus lorsque nous avons accès à des informations supplémentaires sur la structure du groupe. En effet, ces connaissances nouvelles laissent la possibilité de voir émerger des algorithmes sensiblement plus rapides. La méthode du calcul d'indice, qui prend appui sur ces informations supplémentaires pour fournir des algorithmes sous-exponentiels, demeure la méthode incontournable lorsque l'on souhaite efficacement découvrir des logarithmes discrets.

2.1 Description générale

Bien que la méthode du calcul d'indice se montre aussi performante pour factoriser de grands entiers, nous nous intéresserons uniquement au cas des algorithmes de cette famille qui permettent de résoudre le problème du logarithme discret, encore une fois dans un groupe cyclique G engendré par g. Les algorithmes de cette famille s'articulent autour de plusieurs phases :

2.1.1 Phase préliminaire

Tout algorithme par calcul d'indice commence par fixer la description de G qui sera utilisée pour la suite, et qui peut différer de celle initialement fournie. Par exemple, il est possible de travailler sur $\mathbb{F}_{2^4}$ vu comme $\mathbb{F}_2[Y,Z]/(Y^2+Y+1, Z^2+Z+Y)$ même si ce corps fini[1] est initialement donné par $\mathbb{F}_2[X]/(X^4+X+1)$. Cette phase détermine aussi un sous-ensemble relativement petit d'éléments particuliers de G, que l'on nomme *la base de friabilité*, ou parfois *la base de facteurs*. Celle-ci est constituée d'éléments eux-même considérés comme petits, en un sens restant à définir.

2.1.2 Collect des relations, ou phase de crible

L'objectif de cette étape est de créer un grand nombre de relations multiplicatives entre éléments de la base de friabilité. Si la base utilisée est $\{g_i, i \in I\}$ nous nous intéresserons à des équations de la forme :

$$\prod_{i\in I} {g_i}^{m_i} = \prod_{i\in I} {g_i}^{n_i}, \tag{2.1}$$

1. Il s'agit ici d'un exemple didactique. Bien évidemment, ce corps est infiniment trop petit pour que la construction d'un algorithme de calcul d'indice sur celui-ci ait un intérêt.

les m_i et les n_i étant des entiers modulo l'ordre de G. En prenant le logarithme discret de chacun des deux membres nous en déduisons l'égalité modulaire :

$$\sum_{i \in I} m_i \log_g g_i \equiv \sum_{i \in I} n_i \log_g g_i \mod |G|.$$

Nous obtenons ainsi une équation linéaire entre les logarithmes des g_i, qui constituent nos inconnues. La phase de crible s'arrête lorsque l'on a récolté un nombre suffisant d'équations de cette forme, de sorte de pouvoir en extraire une solution unique (à constante multiplicative près). Autrement dit, le rang du système doit être égal au nombre d'inconnues moins un, modulo chacun des facteurs de $|G|$.

2.1.3 Algèbre linéaire

Cette étape cherche à résoudre le système linéaire issu des relations, afin d'obtenir les logarithmes discrets de tous les éléments de la base de friabilité – ou, tout du moins, une grande partie d'entre eux. En effet, certaines propriétés de la phase de crible peuvent mener à écarter quelques éléments de la base de friabilité, qui ne seront présents dans aucune équation. Auquel cas, le rang du système devra être égal au nombre d'inconnues moins un, moins ces inconnues artefacts. Notons qu'à l'issue de l'algèbre linéaire, puisque l'on obtient un élément arbitraire du noyau de la matrice des relations, les logarithmes retrouvés ne le sont qu'à constante multiplicative près. Autrement dit, au lieu d'expliciter l'ensemble $\{\log_g(g_i),\ i \in I\}$, on obtient $\{\lambda \log_g(g_i),\ i \in I\}$ pour un certain entier λ. Toutefois, il est facile de retrouver les bons logarithmes, en supposant que le générateur g appartienne à la base de friabilité, quitte à l'inclure artificiellement. Le logarithme de g, qui vaut 1, nous donne la valeur de ce λ artéfact.

Puisque peu d'éléments interviennent dans chacune des relations, le système produit est donc *creux*, c'est-à-dire que la matrice des coefficients m_i et n_i est essentiellement constituée de zéros. Cette propriété matricielle permet d'accélérer grandement l'algèbre linéaire car il existe des algorithmes de résolution spécifiques dont la complexité est alors quadratique, et non cubique comme dans le cas général. Cette particularité fait l'objet du paragraphe 2.3.

2.1.4 Calcul d'un logarithme individuel, ou phase de descente

Afin de résoudre réellement le problème du logarithme discret dans G, nous devons pouvoir retrouver le logarithme d'un élément arbitraire, et non seulement ceux des éléments de la base de friabilité. Soit $h \in G$ un tel élément arbitraire. Nous notons encore x logarithme en base g. A grands traits, l'idée

consiste alors à décomposer l'élément h en produits d'autres éléments qui peuvent être considérés comme plus petits que lui – toujours selon le même sens de petitesse que celui considéré lors de l'établissement de la base. En itérant ce procédé, h s'exprime finalement comme produit d'éléments de la base de friabilité, c'est à dire que l'on peut écrire :

$$g^x = h = \prod_{i \in I} {g_i}^{\alpha_i}.$$

Puisque l'on connait les logarithmes discrets de ces derniers, on retrouve alors facilement l'entier x en écrivant $x = \sum_{i \in I} \alpha_i \log g_i \mod |G|$.

2.2 Un souci de friabilité

2.2.1 Friabilité et heuristique

Les algorithmes par calcul d'indice reposent sur l'idée de décomposer des éléments [2] comme produits d'éléments considérés comme petits. Les éléments qui peuvent se factoriser de cette manière sont dits *friables*. Un problème essentiel pour l'analyse de ces algorithmes consiste donc à estimer la probabilité d'obtenir de tels éléments friables. Dans de nombreux cas, nous procèdons heuristiquement en supposant que les éléments créés se comportent comme des éléments aléatoires de même taille. Bien qu'inélégante, car non prouvée, cette heuristique a permis d'obtenir de nombreux progrès algorithmiques et a conduit à un grand nombre de factorisations explicites d'entiers et de calculs de logarithmes discrets.

Dépendre d'une telle heuristique est inconfortable, et nous aimerions nous en abstraire. Pourtant, les algorithmes rigoureux qui existent actuellement se montrent bien moins efficaces que leurs homologues heuristiques. De manière tout à fait surprenante, les dernières avancées spectaculaires concernant les corps finis de petite caractéristique reposent justement sur le fait que, lorsque certaines heuristiques deviennent fausses (et que les éléments engendrés par la phase de création de relations ne se conduisent pas comme des éléments aléatoires), il devient possible de retourner cette faille à notre avantage, en forçant les éléments que nous considérons à être bien plus souvent friable que leurs semblables arbitraires.

2.2.2 Friabilité et complexité

La complexité des algorithmes par calcul d'indice s'exprime généralement avec une notation particulière, qui provient précisément des estimations de

2. En pratique : des entiers, des idéaux ou des polynômes.

probabilité de friabilité d'éléments aléatoires. Soyons un peu plus rigoureux et éclairons notre propos de quelques définitions et théorèmes importants.

Définition 2.2.1 (friabilité). Un entier est dit *B-friable* si tous ses facteurs premiers sont inférieurs à B, tandis qu'un polynôme sur un corps fini est dit *d-friable* si tous ses facteurs irréductibles sont de degré au plus d.

Canfield, Erdös et Pomerance [CEP83] évaluent en 1983 les probabilités précises de friabilité des entiers. Une dizaine d'année plus tard, Panario, Gourdon et Flajolet [PGF98] généralisent cette idée à la friabilité d'un polynôme sur un corps fini. Ces deux résultats étonnamment proches l'un de l'autre peuvent se résumer ainsi :

Estimation 2.2.1. *Pour une large plage de paramètres, la probabilité qu'un entier aléatoire inférieur à A soit B-friable – respectivement qu'un polynôme aléatoire de degré inférieur à k soit d-friable – est :*

$$u^{-u+o(1)}$$

où u est donné par $u = \dfrac{\log A}{\log B}$ – respectivement $u = \dfrac{k}{d}$.

L'écriture formelle de ce théorème, où la plage de paramètres sera notamment explicitée, sera donnée lorsque nous l'appliquerons au sein de nos analyses de complexités. L'utilisation de ce résultat est simplifiée par la notation :

$$L_Q(\alpha, c) = \exp\Big((c + o(1))(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\Big)$$

où α et c sont des constantes telles que $0 \leqslant \alpha \leqslant 1$ et $c > 0$, et où $o(1)$ tend vers zéro pour $Q \to \infty$. L'abréviation $L_Q(\alpha)$ sera utilisée lorsque nous souhaiterons négliger c. Dans la première phase d'un algorithme par calcul d'indice, le nombre d'opérations à effectuer est grossièrement égal au nombre de relations que l'on souhaite obtenir multiplié par l'inverse de la probabilité ; il est donc normal que l'on retrouve la notation L_Q dans la complexité asymptotique finale de l'algorithme. Habituellement, l'entier Q désigne le cardinal du groupe cible dans lequel nous résolvons le problème du logarithme discret. Pour comparer un algorithme qui nécessite $L_Q(\alpha, c)$ opérations au total à d'autres algorithmes de même nature, il faut d'abord étudier le premier paramètre, puisqu'il gouverne le passage d'un algorithme exponentiel (en temps) à un algorithme polynomial. Plus précisément, si α tend vers 1, $L_Q(\alpha)$ devient exponentiel en la taille de Q, c'est à dire en $\log Q$. Soulignons que $\log Q$ est le nombre de bits nécessaires pour encoder les éléments du groupe en question ; il est donc naturel que ce soit cette valeur que l'on considère lorsque l'on

exprime la complexité des algorithmes. De même, lorsque α vaut 0, $L_Q(\alpha)$ est *polynomial* en $\log Q$. On qualifie de *sous-exponentiel* tout algorithme de calcul de logarithme discret sur un groupe de taille Q qui a une complexité en $L_Q(\alpha)$ avec $0 < \alpha < 1$. De même, on qualifie de *quasipolynomial* tout algorithme dont la complexité est en $L_Q(o(1))$. La Figure 2.1 donne une indication du type des complexités actuelles du problème du logarithme discret pour certains groupes usuels.

2.3 Algèbre linéaire creuse

2.3.1 L'algèbre linéaire, un goulot d'étranglement

Les algorithmes par calcul d'indice s'agencent toujours autour d'une phase d'algèbre linéaire pour retrouver les logarithmes des éléments de la base de friabilité. Puisque ces logarithmes sont uniquement déterminés modulo l'ordre du groupe cible, il est nécessaire de résoudre un large système d'équations linéaires sur un anneau résiduel $\mathbb{Z}/m\mathbb{Z}$. Longtemps, au cours des années 70 et au début des années 80, cette étape marque un frein majeur, aussi bien en pratique que dans les estimations données pour les temps de complexités des algorithmes de l'époque. Cette limitation provenait de la complexité cubique nécessaire à la résolution des systèmes linéaires par les méthodes classiques, comme l'élimination gaussienne.

Aujourd'hui encore, cette phase d'algèbre linéaire reste difficile et cristallise un point de complexité importante aussi bien pour le problème du logarithme discret que pour la factorisation. Malgré tout, soulignons que la factorisation nécessite simplement la résolution de systèmes définis modulo 2, alors que le calcul de logarithmes discrets demande des solutions modulo de grands entiers. Cette différence explique le décalage temporel qui persiste en pratique entre les records de factorisation et ceux de logarithmes discrets dans des corps premiers $\mathbb{F}_p$, p étant un nombre premier. A titre de comparaison, le DLP a été brisé sur $\mathbb{F}_p$ avec p premier de 768 bits en 2016 – nous renvoyons au tableau 3.1 qui référence les records en date – tandis que le plus grand entier de type RSA a avoir été factorisé est aussi un entier de 768 bits, mais le record date de 2009. Heureusement, les systèmes d'équations linéaires produits par les algorithmes de calcul d'indice sont très souvent creux.

2.3.2 Algorithmes de résolutions de systèmes linéaires creux

Une matrice creuse est une matrice qui ne contient qu'un petit nombre d'entrées non nulles. Très souvent, elles prennent la forme de matrices pour lesquelles chaque ligne (ou chaque colonne) ne présente qu'un petit nombre de

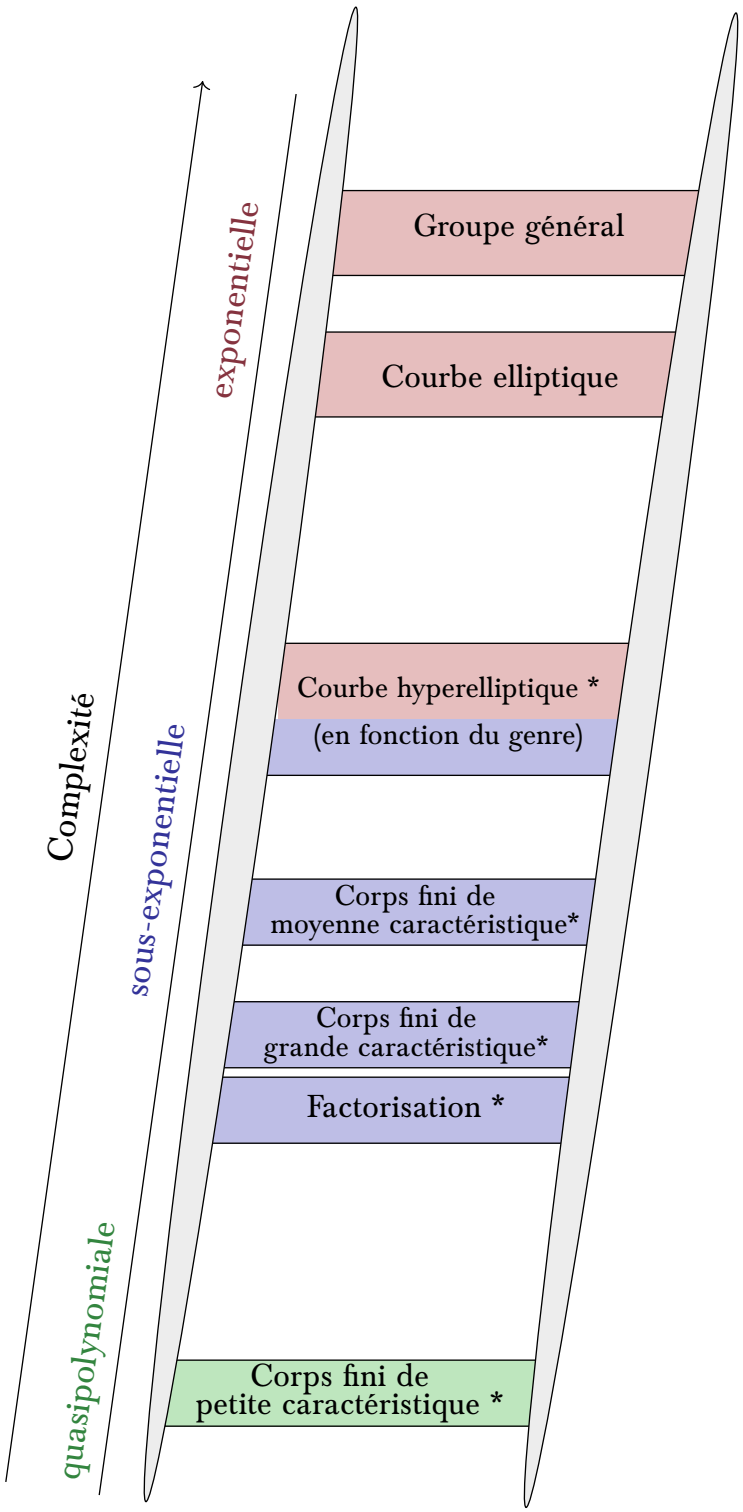


Figure 2.1 – Echelle schématique des complexités asymptotiques actuelles de la factorisation et du problème du logarithme discret dans différents groupes. L'astérisque indique l'appartenance à la famille des algorithmes par calcul d'indice.

telles entrées, comparé à la dimension de la matrice. A dimension égale, une matrice creuse occupera ainsi bien moins de place en mémoire : il suffit en effet de décrire chaque ligne (ou chaque colonne) par la liste des positions qui contiennent un coefficient non nul, associée à la liste des valeurs du coefficient correspondant. Lorsque nous traitons des systèmes d'algèbre linéaire creux, l'utilisation de l'élimination gaussienne est à proscrire. En effet, chaque étape de pivot introduit potentiellement de nouvelles entrées non nulles, suite à quoi, après un petit nombre d'étapes, la matrice ne peut plus être considérée comme creuse. Si la dimension de la matrice initiale est grande, l'élimination gaussienne sature donc très rapidement la mémoire allouée.

Trois familles d'algorithmes se dégagent lorsque l'on souhaite résoudre des problèmes d'algèbre linéaire sur des matrices creuses.

La première d'entre elles, l'élimination gaussienne structurée, a été proposée par Odlyzko [Odl85] dès 1985 pour être implémentée cinq ans plus tard dans [LO90]. Comme son nom le suggère, il s'agit d'une variante de l'élimination gaussienne dans laquelle l'algorithme sélectionne les pivots qui minimisent le remplissage de la matrice tout au long du procédé. Les méthodes qui s'appuient sur cette idée sont toujours utilisées comme préliminaire pour produire un système de dimension réduite qui reste pourtant raisonnablement creux. Les matrices qui en résultent sont ensuite traitées à l'aide d'un algorithme issu de l'une des deux familles suivantes.

Ces deux familles partagent la propriété commune de laisser inchangée la matrice d'entrée. Plus précisément, elle n'apparait dans ces différents algorithmes que sous la forme de produits matrice-vecteur : lorsque certains vecteurs sont multipliés soit par cette matrice, soit par sa transposée. La première de ces deux familles comporte les méthodes des sous-espaces de Krylov, qui nous proviennent de l'analyse numérique et ont été transformées en vue de la construction de suites de vecteurs mutuellement orthogonaux. En particulier cette famille contient l'algorithme de Lanczos, ainsi que l'algorithme du gradient conjugué. Le lecteur intéressé trouvera la description de ces méthodes, adaptées au contexte du logarithme discret, au sein de l'article [COS86].

La seconde famille s'articule autour de l'algorithme de Wiedemann [Wie86] publié en 1986, et de sa variante parallélisable, l'algorithme de Block Wiedemann [Cop94] proposé par Coppersmith en 1994. Succinctement, afin de résoudre un système linéaire, ces algorithmes reposent sur le calcul du polynôme minimal de la matrice associée. Lorsqu'il s'agit de Block Wiedemann l'objet manipulé est plus complexe mais l'idée reste similaire. Tous les détails de cette famille d'algorithmes seront traités au chapitre 6, puisqu'il s'agit du point de départ de la méthode que nous proposons pour résoudre des problèmes d'algèbre linéaire sur une classe plus large de matrices, incluant les

matrices creuses.

Enfin, notons que tous les algorithmes issus de ces deux familles – sous-espaces de Krylov et Wiedemann – coûtent un nombre de multiplications matrice-vecteur égal à un petit multiple de la dimension de la matrice d'entrée. Ainsi, pour une matrice $N \times N$ qui contient au plus ℓ coefficients non nuls par ligne, le coût global de ces algorithmes est en $O(\ell N^2)$. Donc bel et bien quadratique, sous réserve bien entendu que ℓ reste petit devant N.

2.4 Calcul d'indice dans les corps finis

2.4.1 La récolte des relations, une étape clef

Les algorithmes de calcul d'indice reposant sur la construction de relations multiplicatives, l'exposé succinct de différentes méthodes pour obtenir de telles relations permet dans un même temps de découvrir l'historique du logarithme discret dans les corps finis.

Test simple de friabilité, ou comment obtenir un algorithme en $L(1/2)$. Les premiers algorithmes de calcul d'indice pour les corps finis ont été proposés à la fin des années 70 par Adleman [Adl79] pour les corps de cardinalité première, que nous appellerons souvent par abus de langage : *corps premiers*. Pour calculer le logarithme discret de h modulo un nombre premier p, l'approche la plus simple consiste par exemple à prendre un entier aléatoire a, à calculer un représentant de h^a modulo p dans l'intervalle $[\![0, p-1]\!]$ et à vérifier si :

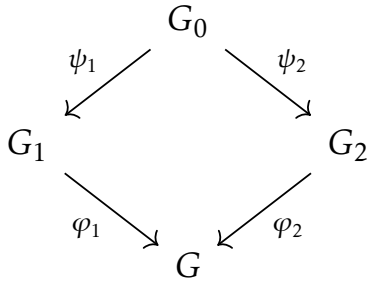
$$h^a \bmod p = \prod g_i$$

avec g_i des nombres premiers inférieurs à une certaine borne B. On obtient ainsi des relations dans une base de friabilité formée de h et de l'ensemble des nombres premiers plus petits que B. En supposant que g est bien dans la base de friabilité, on obtiendra directement le logarithme recherché sans qu'aucune phase de descente ne soit nécessaire [3]. Notons que pour la plupart des valeurs de a, h^a ne sera pas B-friable, et sera donc écarté. La complexité dépendra donc de la probabilité qu'un entier aléatoire inférieur à p soit B-friable. L'algorithme d'Adleman sous cette forme ne s'applique qu'aux corps premiers. Il a toutefois été généralisé aux corps finis de la forme $\mathbb{F}_{p^k}$, pour p fixé, par Hellman et Reyneri en 1982 [HR82]. Bien que primitive, cette approche permet déjà d'obtenir une complexité sous-exponentielle, de la forme $L_p(1/2)$. Elle se

3. Si g n'est pas dans la base, il est facile de l'ajouter en considérant des relations un peu plus générales de la forme $h^a g^b = \prod g_i$

généralise à de nombreux corps finis et présente l'intérêt de fournir un algorithme probabiliste rigoureusement prouvé [Pom87], contrairement aux autres algorithmes de la même famille, qui, bien que plus performants, reposent encore sur une ou plusieurs hypothèses heuristiques.

La méthode du diagramme commutatif. Une idée essentielle pour améliorer la recherche de relations multiplicatives consiste à préalablement décrire le groupe G de façon à pouvoir l'inclure dans un diagramme commutatif de la forme suivante :



Dans ce diagramme commutatif, G_0, G_1 et G_2 sont des groupes bien choisis – la plupart du temps, il s'agit même d'anneaux – et toutes les applications sont des morphismes. Ainsi, pour tout x dans G_0, nous obtenons une égalité dans G :

$$\varphi_1(\psi_1(x)) = \varphi_2(\psi_2(x)).$$

Comme dans la description générale, il n'est gardé qu'une petite partie de ces relations. A cette fin, il faut distinguer dans chaque groupe intermédiaire G_1 et G_2 un petit sous-ensemble d'éléments, disons $B_1 \subset G_1$ et $B_2 \subset G_2$. Le sous-ensemble $\varphi_1(B_1) \cup \varphi_2(B_2)$ de G est alors la base de friabilité de cet algorithme. Cela signifie que l'on ne conserve que les relations pour lesquelles $\psi_1(x)$ se décompose comme $\prod_{b_{1,i} \in B_1} b_{1,i}$ et $\psi_2(x)$ comme $\prod_{b_{2,i} \in B_2} b_{2,i}$. Ainsi, la relation obtenue est en vérité une égalité entre produits dans G de la forme :

$$\prod_{b_{1,i} \in B_1} \varphi_1(b_{1,i}) = \prod_{b_{2,i} \in B_2} \varphi_2(b_{2,i})$$

Si cette méthode se révèle générale, les choix de G_0, G_1 et G_2 sont en revanche spécifiques à la nature du groupe cible, et à quelques paramètres bien précis. Nous donnons un exemple concret de ces choix quelques lignes plus bas.

En 1984, Coppersmith [Cop84] obtient ainsi les premiers algorithmes de complexité $L_Q(1/3)$ où Q est la taille du corps considéré. Ce résultat, limité au cas des corps binaires, c'est-à-dire de caractéristique 2, est ensuite étendu progressivement à tous les corps finis, ainsi qu'à la factorisation. Néanmoins, il faut attendre 1986 et l'introduction de la méthode des entiers gaussiens

(*Gaussian Integer Method* en anglais) par Coppersmith, Odlyzko et Schroeppel [COS86] pour achever en pratique les premiers records significatifs. Bien qu'en $L_p(1/2, c)$, ce dernier présente pour la première fois une valeur de c égale à 1, c'est-à-dire bien plus basse que les précédentes valeurs courantes pour le cas des corps premiers $\mathbb{F}_p$. Les algorithmes en $L(1/3)$ se limitent pendant de longues années aux corps finis de petite caractéristique (en atteste le résultat de Coppersmith en 1984), aux corps premiers (concernés par les premiers résultats d'Adleman puis par ses variantes), voire, éventuellement, à ceux de petite extension [Sch00]. Le vent tourne pourtant en 2006 lorsque deux articles démontrent l'existence d'un algorithme de complexité $L(1/3)$ pour tous les corps finis, qu'importe la caractéristique.

Il s'agit préalablement de scinder les corps finis en trois sous-ensembles : ceux de petite caractéristique atteignent une complexité de $L(1/3, (32/9)^{1/3})$, ceux de caractéristique moyenne, une complexité de $L(1/3, (128/9)^{1/3})$ et ceux de grande caractéristique, une complexité de $L(1/3, (64/9)^{1/3})$ tous dans le courant de la même année, en 2006. La figure 2.2 reprend l'historique des algorithmes créés pour résoudre le problème du logarithme discrets dans les corps finis de 1970 à 2013, date du début de la thèse dont ce manuscrit est le fruit. Nous avertissons le lecteur que ce dessin n'est pas à l'échelle, puisque une variation du premier ou du second paramètre dans la notation en L_Q produit des effets de nature très différente sur les complexités. Cela étant, la couleur de fond de chacun des algorithmes illustre l'importance et la variation du premier paramètre : nous colorons ainsi en rouge les algorithmes en $L_Q(1/2)$, en bleu ceux en $L_Q(1/3)$, et en vert ceux en $L_Q(1/4)$ ou moins – quasipolynomiaux notamment. Ce dessin donne aussi une indication temporelle sur la création des différents algorithmes : à couleur fixée, plus le fond est sombre, plus l'algorithme est récent. Enfin, la notation $C(x) = L_Q(1/3, (x)^{1/3})$ permet simplement de faire rentrer toutes les complexités sur une seule et même page. Ainsi, le crible par corps de fonctions ou FFS [AH99, JL06] (de l'anglais *Function Field Sieve*) permet de calculer des logarithmes dans tous les corps finis de petite caractéristique, tandis que le crible par corps de nombres ou NFS [JLSV06] (de l'anglais *Number Field Sieve*) concerne les corps de caractéristiques moyenne à grande.

Petite, moyenne et grande caractéristiques. Examinons cette notion de taille : lorsque nous qualifions la caractéristique d'un corps de petite ou de grande, nous considérons en vérité implicitement la taille *relative* de cette caractéristique avec celle du degré de l'extension, pour une taille de corps fixée. Ainsi, si l'on s'intéresse au corps fini $\mathbb{F}_{p^n}$ et que l'on souhaite évaluer la taille de sa caractéristique p, on commencera par écrire p sous la forme $p =$

$L_{p^n}(l,c)$ avec $0 \leq l \leq 1$ et c une constante proche de 1. Si $l < 1/3$, on parlera de petite caractéristique, si $1/3 < l < 2/3$, de moyenne caractéristique, et de grande si $2/3 < l$. Avec cette notation, nous remarquons qu'un corps premiers $\mathbb{F}_p$ est donc un corps de grande caractéristique – puisque $p = L_p(1,1)$ – tandis qu'un corps binaire $\mathbb{F}_{2^n}$ est un corps de petite caractéristique – puisque $2 = L_{2^n}(0, \log 2/(\log(n \log 2)))$, ce qui correspond bien à l'intuition voulue.

Cette distinction en trois domaines s'éclaire à la lumière de la remarque suivante : en théorie de la complexité, le paramètre pertinent à prendre en compte pour évaluer la complexité d'un algorithme est essentiellement la taille en bits de l'objet qu'il prend en entrée. Lorsque l'on souhaite calculer des logarithmes dans $\mathbb{F}_{p^n}$, l'ordre de grandeur est donc donné par :

$$\log p^n = n \log p.$$

La caractéristique est grande lorsqu'un grand nombre de bits d'information est porté par la quantité qui lui est associée, c'est-à-dire $\log p$ – et petite, dans le cas contraire. Par souci de partage en domaines égaux, on peut fixer les limites en 1/3 et 2/3. Dit autrement : si 2/3 des bits d'information de la taille du corps est porté par $\log p$, alors la caractéristique est grande ; si 1/3 l'est seulement, c'est qu'elle est petite. Cette définition intuitive rejoint agréablement celle précédemment donnée, dont la notation en L_{p^n} rappelle que la création des trois domaines provient de la comparaison des complexités asymptotiques, mais dissimule l'idée naturelle.

2.4.2 Corps de nombres ou corps de fonctions ?

Dans le diagramme commutatif, les groupes G_0, G_1 et G_2 donnent une représentation implicite du corps fini $G = \mathbb{F}_{p^n}$. Les deux groupes intermédiaires G_1 et G_2 ne peuvent donc pas être tous les deux très petits. Cette restriction limite le choix de la base de friabilité, tout comme la forme des relations multiplicatives créées. Ce sont de ces restrictions que découle enfin une barrière naturelle de complexité en $L_Q(1/3)$. Etudions de plus près la nature de ces groupes.

Moyenne et grande caractéristique : crible par corps de nombres. Les différentes variantes de cribles par corps de nombres se basent sur un diagramme qui part de l'anneau de polynômes à coefficients entiers $G_0 = \mathbb{Z}[X]$ et passe par deux corps de nombres $G_1 = \mathbb{Q}(X)/(f_1(X))$ et $G_2 = \mathbb{Q}(X)/(f_2(X))$. Les deux polynômes de définition f_1 et f_2 sont choisis pour avoir une racine commune dans $\mathbb{F}_{p^n}$. La connaissance de G_1 et G_2 permet donc de retrouver G, c'est la représentation implicite que nous évoquions précédemment.

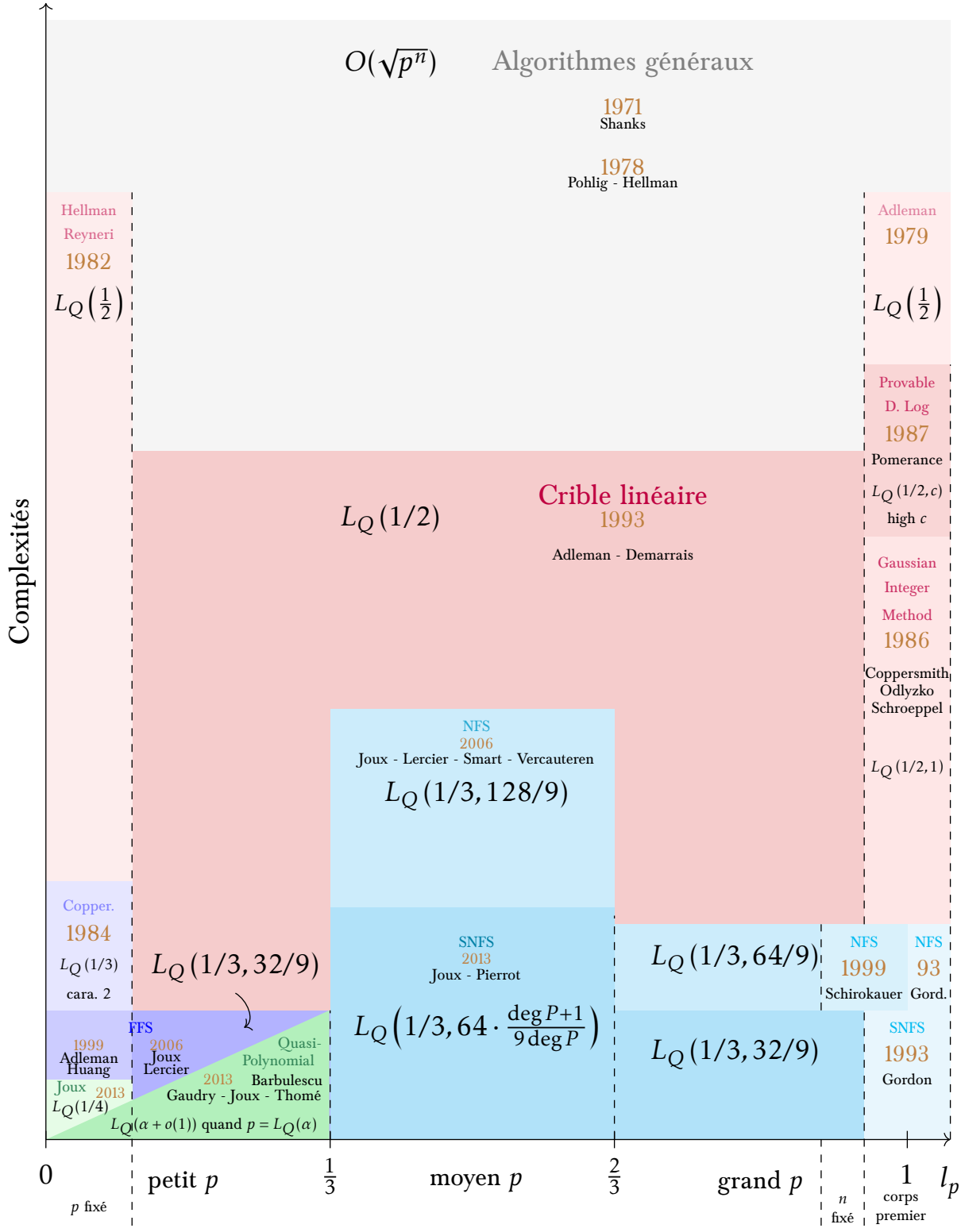


Figure 2.2 – *Représentation globale des différents algorithmes de calcul de logarithmes discrets dans les corps finis, et de leurs complexités, entre 1970 et 2013 – au début de cette thèse.*

A l'intérieur de chaque corps de nombres, les petits éléments qui nous servent à construire la base de friabilité sont les idéaux dont la norme est plus petite qu'une certaine borne de friabilité. L'utilisation des idéaux provient du besoin d'obtenir une factorisation unique de chaque côté du diagramme. La manière de descendre explicitement ces idéaux vers le corps fini dans le diagramme est complexe, et nous passons les détails sous silence dans cette partie.

Nous étudierons au chapitre 5 comment l'utilisation d'un diagramme à plusieurs branches permet de tirer parti d'un grand nombre de corps de nombres simultanément. En moyenne caractéristique, le crible par corps de nombres multiples [Pie15] atteint une complexité asymptotique de :

$$L_{p^n}(1/3, (8(9+4\sqrt{6})/15)^{1/3}))$$

tandis qu'en grande caractéristique le crible présenté dans [BP14] est de complexité :

$$L_{p^n}(1/3, (2(46+13\sqrt{13})/27)^{1/3})),$$

cette seconde constante étant plus faible. Sur ces deux plages, ces deux algorithmes sont actuellement les plus performants asymptotiquement pour résoudre le problème sur des corps finis qui ne présentent aucune particularité, ni extension composée étrange, ni caractéristique creuse. Certains corps d'extension particulière ont un algorithme dédié qui permet de descendre encore la complexité du problème du logarithme discret [BGK15, SS16c, KB16, SS16d, JK16].

De telles expressions peuvent sembler un peu barbares. Le second résultat est pourtant particulièrement intéressant puisque l'on retrouve exactement la complexité la plus basse connue pour factoriser un entier de même taille ! Et c'est plutôt heureux, car le crible par corps de nombres multiples est l'adaptation d'un algorithme de factorisation similaire.

Petite caractéristique : crible par corps de fonctions. En petite caractéristique, les groupes utilisés ont une structure plus simple puisqu'il s'agit de corps de fonctions. Depuis 2006, la simplification est telle que l'on ne considère plus que de simples anneaux de polynômes, bien que l'on conserve, par tradition, le terme de *crible par corps de fonctions*. Pour cette dernière construction, nous prenons $G_0 = \mathbb{F}_p[X, Y]$, $G_1 = \mathbb{F}_p[X]$ et $G_2 = \mathbb{F}_p[Y]$, liés par des relations de la forme $Y = f_1(X)$ et $X = f_2(Y)$ permettant de définir les fonctions φ et ψ (par exemple $\psi_1(P(X, Y)) = P(X, f_1(X))$). La contrainte d'avoir une racine commune dans $\mathbb{F}_{p^n}$ s'exprime simplement en exigeant que $f_2(f_1(X)) - X$ ait un facteur irréductible de degré n dans $\mathbb{F}_p[X]$.

Au lieu de chercher à factoriser des entiers (les normes des idéaux dans les corps de nombres), nous travaillons cette fois sur des polynômes à coefficients dans un petit sous-corps. Si la mécanique générale reste la même, la structure des polynômes sur les corps finis est toutefois bien mieux comprise ! Le chapitre 3 explique comment dans un tel contexte l'un des ingrédients majeurs pour briser la barrière en $L_Q(1/3)$ consiste à invalider l'heuristique usuelle en exhibant des polynômes de grands degrés – donc des éléments qui ne sont plus si petits que cela – qui ont la propriété agréable de se factoriser systématiquement en termes de petits degrés.

2.5 Calcul d'indice dans les courbes elliptiques

Derrière ce sous-titre alléchant se cache une déception inévitable. Les algorithmes sous-exponentiels de calcul d'indice s'appliquent à une large variété de problèmes de logarithmes discrets. Pourtant, et l'exception est notable, il n'est pas connu d'algorithme pour les courbes elliptiques plus performants que les algorithmes généraux. Il existe bien des méthodes qui fonctionnent pour certaines classes de courbes particulières, ou d'autres qui déplacent le problème vers un corps fini [MOV93, FR94] ou une courbe de genre plus grand [GHS02], mais en toute généralité, le problème reste de complexité exponentielle.

Néanmoins, afin de rétablir la confiance du lecteur vexé de s'être fait prendre par ce titre aguicheur, et pour satisfaire sa curiosité que nous imaginons galopante, nous pouvons brosser un portrait très rapide de ces cas particuliers pour lesquels la difficulté du problème du logarithme discret est réduite. Nous renvoyons à l'article de Galbraith et Gaudry [GG16] pour un survol complet du problème de logarithme discret sur les courbes. Il existe ainsi des algorithmes de calcul d'indice pour les courbes de grands genres. Par ailleurs, lorsque l'on considère des courbes elliptiques définies sur des extensions, deux familles d'attaques se distinguent : les attaques par recouvrement – ou par descente de Weil – et les attaques par décomposition. Certains configurations permettent aussi de combiner ces deux types d'attaques pour obtenir des algorithmes plus efficaces [JV12].

2.5.1 Courbes de grand genre

Un résultat primordial concernant les courbes de genre plus grand que 3 a été découvert en 2007 [GTTD07]. Cet article présente en effet un algorithme de calcul d'indice qui s'applique aux courbes hyperelliptiques de genre $g \geq 3$ définie sur un corps fini $\mathbb{F}_q$, et qui calcule des logarithmes discrets sur cette

courbe en temps :

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

Cette complexité asymptotique est à comparer avec celle des algorithmes génériques qui sont en $\tilde{O}(q^{g/2})$. D'autres algorithmes plus récents pour les courbes de grand genre permettent de généraliser aux cas des courbes non hyperelliptiques, comme proposé dans les articles [EGT11] et [DK13].

2.5.2 Courbes elliptiques particulières

Descente de Weil. Les attaques par recouvrement introduites dans [GHS02] en 2002 visent à transporter le problème du logarithme discret d'une courbe elliptique sur une extension de corps vers une courbe de plus grand genre définie quant à elle sur un corps plus petit. Si le genre de la seconde courbe n'est pas trop grand, ceci mène à un algorithme de calcul de logarithme discret plus efficace que dans le cas général.

Décomposition. Initialement proposée en 2004, la méthode de décomposition [Sem04] recherche des relations entre les éléments de la base de friabilité qui sont exhibées en s'aidant du n-ième polynôme de sommation de Semaev. Ce dernier autorise en effet une traduction polynomiale, donc plus facilement utilisable, du fait que l'addition de n points sur une courbe puisse donner le point à l'infini. Rappelons que, puisque nous travaillons ici sur le groupe des points rationnels d'une courbe, qui est un groupe additif, le pendant des relations de la forme $\prod g_i^{m_i} = 1$ données par l'équation (2.1) se rencontre sous l'égalité :

$$\sum m_i \cdot P_i = \mathcal{O},$$

où les P_i sont des points dans la base de friabilité, les m_i des entiers, et où $\mathcal{O}$ représente l'élément neutre par l'addition, c'est-à-dire précisément le point à l'infini. Grâce à la symétrie du polynôme de Semaev, il est possible de réduire son degré en exprimant tout en termes de polynômes symétriques élémentaires en l'abscisse des points solutions. Ainsi, sur une extension de corps de degré proche de n au-dessus du corps de base, choisir comme base de friabilité des points d'abscisse dans ce corps de base permet de réécrire l'équation donnée par le polynôme de Semaev comme un système polynomial dans le corps de base.

Lorsque le degré d'extension est trop large et ne peut se décomposer pour donner lieu à une tour d'extensions favorable, la situation est plus complexe. En revanche, les courbes elliptiques définies sur $\mathbb{F}_{2^p}$, où p est un nombre premier, peuvent se révéler sensibles à certaines attaques. Ces dernières sont

étudiées au sein des articles [FPPR12] et [PQ12]. La plus grande difficulté réside dans le fait suivant : contrairement au cas précédent, aucun des choix naturels de base de friabilité n'est préservé par les polynômes symétriques (en les abscisses). La réduction du degré du polynôme de Semaev n'est donc plus si aisée, ce qui rend le comportement asymptotique de cette méthode d'autant plus difficile à prédire.

Deuxième partie

Le problème du logarithme discret dans les corps finis de petite caractéristique

Chapitre 3

Des algorithmes sous-exponentiels aux complexités quasipolynomiales

Sommaire

Ce chapitre sert de longue introduction au chapitre suivant, qui détaille quant à lui un algorithme quasipolynomial simplifié permettant d'abaisser la complexité des phases polynomiales. Néanmoins, afin de mieux appréhender comment ce résultat se place dans un contexte compétitif, nous retraçons ici l'historique des évolutions techniques qui ont affecté le problème du logarithme discret pour les corps finis de petite caractéristique ces quarante dernières années. Cette histoire commence par la croyance répandue consistant à poser le problème comme suffisamment difficile pour pouvoir prétendre être à la base de fondations solides pour des protocoles cryptographiques, et s'achève à l'état d'abandon actuel, où les algorithmes d'attaques sont si efficaces que le problème dans ce cas de figure ne peut plus être utilisé à des fins cryptographiques.

La plupart des algorithmes de calcul d'indice ont été créés pour satisfaire des objectifs pratiques, comme l'illustre le tableau 3.1 des records et calculs de grande envergure effectués depuis 1992 aussi bien en petite qu'en grande caractéristiques. Cette exigence de practicité explique aussi pourquoi la communauté privilégie souvent des algorithmes encore heuristiques, lorsqu'ils se montrent plus performants que leurs homologues rigoureux. Aussi, sauf mention contraire explicite, tous les algorithmes mentionnés dans ce chapitre sont heuristiques.

Dans cette partie nous brossons donc les évolutions techniques clefs qui ont permis de modifier le paysage du problème du logarithme discret en petite caractéristique, afin de mettre en évidence la provenance des idées majeures qui charpentent l'état de l'art actuel [BGJT14, AMORH13, AMOR14, GKZ14a, GKZ15]. Pour simplifier notre propos, nous abandonnons dès le départ les corps de fonctions au profit des variantes qui manipulent uniquement des polynômes dans des corps finis. En particulier, nous contournons volontairement les descriptions relatives au crible par corps de fonctions (FFS, comme *function field sieve* en anglais) proposé initialement par Adleman et Huang [AH99]. Nous préférons donner les détails de la variante du même algorithme basée sur des polynômes bivariés [JL06], par souci de cohérence avec les méthodes qui ont suivi.

Table 3.1 – Historique des records de calcul de logarithmes discret. L'étoile ★ dans la colonne de la taille des corps considérés indique qu'il s'agit d'extension de Kummer (éventuellement tordues).

Date	Corps	Taille en bits	Coût (en heures CPU)	Algorithme	Auteurs
1992	2^{401}	401	114000	[COS86]	Gordon, McCurley
1996	p	281	?	[COS86]	Weber, Denny, Zayer
1998/02	Special p	427	12 500	[Gor93]	Weber
1998/05	p	298	2900	[COS86]	Joux, Lercier
2001/01	p	364	290	[JL03]	Joux, Lercier
2001/04	p	397	960	[JL03]	Joux, Lercier
2001/09	2^{521}	521	2 000	[JL02]	Joux, Lercier
2002	2^{607}	607	> 200 000	[Cop84]	Thomé
2005/06	p	431	350	[JL03]	Joux, Lercier
2005/09	2^{613}	613	26 000	[JL02]	Joux, Lercier
2005/10	65537^{25}	400	50	[JL06]	Joux, Lercier
2005/11	370801^{30}	556 ★	200	[JL06]	Joux, Lercier
2007	p	530	29 000	[JL03]	Kleinjung
2012/06	$3^{6\cdot 97}$	923	895 000	[JL06]	Hayashi, Shimoyama, Shinohara, Takagi
2012/12	p^{47}	1175 ★	32 000	[Jou13a]	Joux
2013/01	p^{57}	1425 ★	32 000	[Jou13a]	Joux
2013/02	2^{1778}	1778 ★	220	[Jou13b]	Joux
2013/02	2^{1991}	1991 ★	2200	[GGMZ13]	Gologlu, Granger, McGuire, Zumbragel
2013/03	2^{4080}	4080 ★	14 100	[Jou13b]	Joux
2013/04	2^{809}	809	19 300	[AH99, JL06]	The Caramel Group
2013/04	2^{6120}	6120 ★	750	[GGMZ13, Jou13b]	Gologlu, Granger, McGuire, Zumbragel
2013/05	2^{6168}	6168 ★	550	[Jou13b]	Joux
2014/01	$3^{6\cdot 137}$	1303	920	[Jou13b]	Adj, Menezes, Oliveira, Rodriguez-Henriquez
2014/01	2^{9234}	9234 ★	398 000	[Jou13b]	Granger, Kleinjung, Zumbragel
2014/01	2^{4404}	sous-groupe de 698 bits	52 000	[Jou13b]	Granger, Kleinjung, Zumbragel
2014/09	3^{5*479}	3796, mais sous-groupe de 760 bits	8 600	[JP14]	Joux, Pierrot
2016/04	p^3	508	42 600	[JLSV06, Gui15]	Guillevic, Morain, Gaudry, Thomé
2016/06	p	768	45 600 000	[JL06]	Kleinjung, Diem, A. Lenstra, Priplata, Stahlke
2016/07	3^{6*509}	4841, mais sous-groupe de 806 bits	1 752 000	[JP14]	Adj, Canales-Martinez, Cruz-Cortés, Menezes, Oliveira, Rodriguez-Henriquez, Rivera-Zamarripa

3.1 Préliminaires algébriques pour les corps finis

A travers ce chapitre et le suivant, q représente une puissance d'un nombre premier. Nous supposons que q est fixé et que nous souhaitons résoudre le problème du logarithme discret dans $\mathbb{F}_{q^k}$, une extension de haut degré k du corps de base $\mathbb{F}_q$. Aussi, toutes les complexités sont exprimées en fonctions de k, et les opérations dans le corps de base sont supposées avoir un coût unitaire.

Par ailleurs, nous noterons $\mathbb{F}_q[X]$ l'ensemble des polynômes univariés en X dont les coefficients appartiennent au corps $\mathbb{F}_q$. Nous rappelons que le corps fini $\mathbb{F}^*_{q^k}$ peut être représenté comme $\mathbb{F}_q[X]/(I_k(X))$ où I_k est un polynôme irréductible de degré k, c'est-à-dire que les éléments du corps finis peuvent être représentés comme des polynômes réduits modulo $I_k(X)$. Puisqu'il n'existe qu'un unique corps fini à q^k éléments, toutes ces représentations sont isomorphes. Cependant, certaines d'entre elles se révèlent plus agréables à manipuler lorsqu'il s'agit de calculer des logarithmes discrets dans ce corps ; aussi, trouver une bonne manière de choisir ce polynôme irréductible est une considération importante dans notre contexte.

3.1.1 Factoriser des polynômes à coefficients dans un corps fini

Il est connu que tout polynôme unitaire $F(X)$ dans $\mathbb{F}_q[X]$ peut s'écrire comme le produit :

$$F(X) = \prod_{i=1}^{t} F_i(X)^{e_i},$$

où chaque F_i représente un polynôme irréductible unitaire et $e_i \geq 1$ correspond à sa multiplicité. Cette factorisation en produit d'irréductibles est unique, à l'ordre près des facteurs. Pour des polynômes qui ne sont pas unitaires nous obtenons la décomposition :

$$F(X) = \beta \cdot \prod_{i=1}^{t} F_i(X)^{e_i}$$

en factorisant au préalable simplement par le coefficient de tête β de F.

La décomposition de polynômes en produits d'irréductibles constitue un outil essentiel aux calculs des logarithmes discrets. Une telle décomposition se montre particulièrement utile lorsque tous les facteurs qui interviennent présentent un petit degré. Afin de quantifier cette notion, nous rappelons que nous disons d'un polynôme $F(X)$ qu'il est d-friable si tous ses facteurs irréductibles sont de degré au plus d. Cette propriété est heureusement facile à tester

dans notre cas en pratique, grâce à l'existence d'algorithmes rapides pour la factorisation de polynômes univariés sur les corps finis. Diverses méthodes sont répertoriées dans l'article de Von Zur Gathen et Panario [vzGP01].

3.1.2 Polynômes friables et polynômes irréductibles

Nous pouvons donc tester facilement l'irréductibilité ou la friabilité d'un polynôme. Parallèlement à ces tests, il est utile d'avoir une notion de l'aspect combinatoire de ce même problème, c'est-à-dire de savoir compter le nombre de polynômes irréductibles d'un corps fini, et donner une estimation de la proportion de polynômes friables. En ce qui concerne la friabilité, nous avons à notre disposition le théorème de Panario, Gourdon et Flajolet. Celui-ci donne une estimation de la probabilité pour un polynôme aléatoire d'être friable – comme nous l'avons déjà dit, il s'agit d'une généralisation des probabilités de friabilité des entiers donnée dans [CEP83].

Estimation 3.1.1 (Panario, Gourdon et Flajolet [PGF98]). *La probabilité qu'un polynôme aléatoire de degré inférieur à D soit d-friable est :*

$$(D/d)^{-(D/d)+o(1)}.$$

Cette estimation n'est valable que pour une certaine plage de valeurs relatives de D et de d, que nous n'explicitons pas ici, mais qui couvre bel et bien tous nos cas de figures. Par ailleurs, le nombre $N_q(n)$ de polynômes irréductibles de degré précisément n dans un corps fini de cardinalité q est donné par :

$$N_q(n) = n^{-1} \sum_{d|n} \mu(d) q^{n/d}$$

où μ désigne la fonction de Möbius [LN97]. Nous pouvons donc en déduire le résultat suivant :

Lemme 3.1.1. *Si $N_q(n)$ désigne le nombre de polynômes irréductibles de degré n dans un corps fini de cardinalité q alors $q^n/n - \alpha \leq N_q(n) \leq q^n/n$, où $\alpha = q^{\lfloor n/2 \rfloor + 1}/n$.*

Démonstration. D'après [LN97] nous pouvons écrire :

$$n N_q(n) + \sum_{d|n, d \neq n} d N_q(d) = q^n. \tag{3.1}$$

Ce qui entraine alors la majoration $N_q(n) \leq q^n/n$.

En remplaçant de nouveau à l'intérieur de l'équation (3.1), il en découle :

$$q^n \leq n N_q(n) + \sum_{d|n, d\neq n} q^d \leq n N_q(n) + \sum_{i=1}^{\lfloor n/2 \rfloor} q^d.$$

Nous avons donc extrait la minoration $N_q(n) \geq n^{-1}(q^n - q^{\lfloor n/2 \rfloor + 1}/(q-1))$, ce qui nous permet de conclure le lemme. □

3.1.3 Probabilité de factorisation des polynômes

Les algorithmes par représentation de Frobenius, et en particulier la méthode que nous présentons au chapitre suivant, nécessitent d'estimer les probabilités qu'un polynôme de degré D se factorise en termes de degrés au plus d. Cette estimation passe classiquement par l'heuristique selon laquelle les polynômes que nous traitons se conduisent comme des polynômes aléatoires.

Dans ce paragraphe, nous analysons donc les probabilités de friabilité de polynômes aléatoires au delà de l'estimation 3.1.1. Commençons par traiter un exemple simple : nous considérons la probabilité qu'un polynôme unitaire aléatoire de degré D se factorise en termes linéaires. Sur un corps fini $\mathbb{F}_q$, il y a précisément q^D polynômes unitaires de degré D distincts. Parmi ceux-ci, il est facile de compter le nombre de polynômes sans facteur carré qui se factorisent en termes linéaires, puisqu'ils sont en correspondance avec leurs D racines distinctes dans $\mathbb{F}_q$. Il y a donc précisément :

$$\binom{q}{D} = \frac{q \cdot (q-1) \cdots (q-(D-1))}{D!}$$

polynômes de cette nature. Aussi, la proportion de polynômes qui se factorisent en termes linéaires est minorée par :

$$\binom{q}{D} \cdot q^{-D},$$

qui tend rapidement vers $1/D!$ quand q tend vers l'infini.

Pour obtenir une majoration adéquate, nous devons comptabiliser les polynômes qui se factorisent en termes linéaires et qui possèdent potentiellement des racines multiples. La formule est plus complexe, puisqu'il est nécessaire de calculer une somme sur les différentes partitions de D qui présentent des multiplicités. Cependant, le nombre de termes dans cette somme est indépendant de q, et chacun des termes est un multinôme qui régit le bon nombre de multiplicité pour chacune des racines potentielles. Puisque chaque terme contient

au plus $D-1$ racines, nous pouvons majorer la contribution par $C(D)q^{D-1}$ où $C(D)$ ne dépend pas de q. Par conséquent, lorsque q tend vers l'infini, la majoration de la proportion totale de polynômes qui se factorisent en termes linéaires avec ou sans multiplicité tend vers $1/D!$ elle aussi.

Pour des décompositions plus complexes, et notamment lorsque le degré de friabilité que l'on souhaite est supérieur à 1, une analyse similaire reste faisable, quoique pénible pour des valeurs arbitraires de D et d. Heureusement, dans le chapitre suivant nous n'utiliserons que des valeurs telles que :

$$d+1 > D/2.$$

Sous cette contrainte, l'analyse devient maintenant plus aisée. En effet, si un polynôme P de degré D ne peut se factoriser en termes de degré au plus d, alors il existe au moins un facteur F_κ de gros degré $\kappa \geq d+1$. Puisque $\kappa > D/2$, nous savons que ce facteur est unique. La probabilité que le polynôme P s'écrive comme $F_\kappa \cdot Q$, avec F_κ un polynôme irréductible de degré κ et Q un polynôme arbitraire de degré $D-\kappa$ est précisément $(N_q(\kappa)\cdot q^{D-\kappa})/q^D = N_q(\kappa)/q^\kappa$, où $N_q(\kappa)$ désigne comme précédemment le nombre de polynômes irréductibles de degré κ sur $\mathbb{F}_q$. Par conséquent, cette probabilité est exactement la proportion de polynômes irréductibles de degré κ parmi les polynômes du même degré, qui tend vers $1/\kappa$ lorsque q tend vers l'infini, comme nous l'avons vu. Nous en déduisons le résultat suivant :

Lemme 3.1.2. *Soit d et D deux entiers naturels tels que $d+1 > D/2$. Alors, lorsque q tend vers l'infini, la probabilité qu'un polynôme de degré D se factorise en termes de degré au plus d tend vers :*

$$1 - \sum_{\kappa=d+1}^{D} \frac{1}{\kappa}.$$

En guise d'exemples et puisqu'elles seront directement utiles au chapitre suivant, nous donnons les estimations des probabilités suivantes :

— Pour $D=3$ et $d=2$ la probabilité est $1-\frac{1}{3}=\frac{2}{3}$.
— Pour $D=4$ et $d=2$ la probabilité est $1-\frac{1}{3}-\frac{1}{4}=\frac{5}{12}\approx 0.4167$.
— Pour $D=6$ et $d=3$ la probabilité est $1-\frac{1}{4}-\frac{1}{5}-\frac{1}{6}=\frac{23}{60}\approx 0.3833$
— Pour $D=7$ et $d=3$ la probabilité est $1-\frac{1}{4}-\frac{1}{5}-\frac{1}{6}-\frac{1}{7}=\frac{101}{420}\approx 0.2405$.

3.1.4 Passage d'une représentation à une autre d'un même corps

Concrètement, les défis de calcul de logarithmes discrets dans les corps finis que l'on peut chercher à résoudre commencent toujours par donner une

description explicite de $\mathbb{F}_{q^k}$ comme une extension du corps de base $\mathbb{F}_q$ à l'aide d'un polynôme irréductible I de degré k dans $\mathbb{F}_q[X]$. Ensuite, le problème consiste à trouver le logarithme discret d'un élément h, donné lui-même comme un polynôme modulo I, dans la base g, un second polynôme modulo I. A l'opposé, la plupart des algorithmes de calcul de logarithmes discrets débutent par la recherche de leur représentation propre, souvent différente, du même corps $\mathbb{F}_{q^k}$. A cette fin, ils sélectionnent souvent un second polynôme irréductible J, toujours de degré k dans $\mathbb{F}_q[X]$.

Pour transférer le problème du logarithme discret initial de la représentation originelle vers la représentation de travail dont nous avons besoin, nous procédons de la façon suivante. Soit β une racine du polynôme J et considérons I comme un polynôme à coefficients dans $\mathbb{F}_{q^k} = \mathbb{F}_q(\beta)$ que l'on factorise comme un polynôme dans ce corps plus gros. Nous savons que I peut se décomposer en termes linéaires dans ce corps plus gros, on peut donc exprimer toutes ses racines comme des polynômes en β. Choisissons l'une d'entre elles au hasard, que l'on nommera α. Nous pouvons donc écrire $\alpha = f(\beta)$ où f est une expression polynomiale à coefficients dans $\mathbb{F}_q$. Dans le problème initial, h et g sont donnés comme des polynômes en α, disons que l'on a $H(\alpha)$ et $G(\alpha)$. Dans la nouvelle représentation, il suffit donc de trouver le logarithme discret de $H(f(\beta))$ en base $G(f(\beta))$.

3.1.5 A propos du générateur multiplicatif

L'un des obstacles premiers du calcul de logarithme discret s'élève lorsqu'il s'agit de trouver un élément primitif du corps fini $\mathbb{F}_{q^k}$, c'est-à-dire un générateur multiplicatif du groupe $\mathbb{F}^*_{q^k}$. C'est obstacle peut sembler inexistant si l'on suppose que l'on connait déjà un générateur du groupe lorsque que l'on cherche à calculer un logarithme discret pour une instance donnée. Néanmoins, cette difficulté peut apparaitre en amont, lors de la sélection du groupe à utiliser au sein d'un protocole cryptographique, quelqu'il soit. Une méthode couramment utilisée pour contourner cet obstacle consiste à prendre des éléments aléatoires et à tester si leur ordre est un diviseur strict de $q^k - 1$ ou s'il est égal à ce dernier. Ce test s'avère facile si la factorisation de $q^k - 1$ est connue. Cependant, si l'entier q^k est grand, cette factorisation peut se révéler elle-même difficile. De manière étonnante, les techniques que nous présentons dans ce chapitre pour résoudre des problèmes de logarithmes discrets ont aussi été utiles [HN13] pour retrouver un générateur multiplicatif de $\mathbb{F}_{q^k}$ sans nécessiter la factorisation préalable de $q^k - 1$.

3.2 L'algorithme de Hellman–Reyneri en $L_{q^k}(1/2)$

L'algorithme de Hellman et Reyneri [HR82] est une adaptation pour le corps $\mathbb{F}_{q^k}$ de celui proposé par Adleman [Adl79] et que nous avons déjà évoqué pour le calcul des logarithmes discrets dans les corps premiers. Sa forme originelle est heuristique, comme tous les algorithmes que nous présentons par la suite, mais il est remarquable de noter, pour celui-ci, l'existence d'une variante prouvée que nous devons à Pomerance [Pom87]. Celle-ci permet d'obtenir une complexité asymptotique similaire.

L'algorithme commence par sélectionner I_k, un polynôme unitaire irréductible arbitraire de degré k, afin de représenter $\mathbb{F}_{q^k}$ comme $\mathbb{F}_q[\alpha]$ où α désigne une racine (fixée) de I_k. Nous choisissons ensuite un générateur g du groupe multiplicatif $\mathbb{F}_{q^k}^*$. Par ailleurs, nous fixons un paramètre entier d tel que $1 < d < k$, qui régira une fois de plus le degré maximum des polynômes que nous laisserons apparaitre dans nos relations.

3.2.1 Création des relations

Pour un entier r choisi uniformément aléatoirement dans $[\![0, q^k - 1]\!]$, nous observons que g^r est aussi un élément uniformément aléatoire de $\mathbb{F}_{q^k}^*$. Cet élément peut en particulier être représenté comme un polynôme $G_r(X)$ de degré au plus $k-1$, et calculé efficacement par une méthode d'exponentiation rapide à base d'élévations au carré et de multiplications (***square-and-multiply method*** en anglais). Lorsque ce polynôme G_r est d-friable, nous écrivons :

$$g^r = \beta_r \cdot \prod_{i=1}^{t_r} F_i^{(r)}(\alpha)^{e_i^{(r)}}, \tag{3.2}$$

où les $F_i^{(r)}$ sont les polynômes irréductibles unitaires de degré au plus d qui apparaissent dans la factorisation de G_r et β_r est le coefficient de tête de G_r. En prenant les logarithmes discrets dans $\mathbb{F}_{q^k}$ de ces éléments, nous obtenons une équation affine :

$$r = \log_g(\beta_r) + \sum_{i=1}^{t_r} e_i^{(r)} \log_g\left(F_i^{(r)}(\alpha)\right) \pmod{q^k - 1},$$

où les inconnues sont les logarithmes des éléments de l'ensemble :

$$\mathcal{F} = \left\{\beta | \beta \in \mathbb{F}_q\right\} \cup \left\{F(\alpha) | F \text{ irred. unitaire de degré au plus } d \text{ dans } \mathbb{F}_q[X]\right\}.$$

Cet ensemble constitue la *base de friabilité*. Puisque $\mathcal{F}$ est de cardinalité plus petite que q^{d+1} et comme les équations sont générées de manière aléatoire, nous espérons que la création de q^{d+1} équations de la forme (3.2) nous permette d'écrire un système n'admettant qu'une unique solution formée des logarithmes en base g des éléments de $\mathcal{F}$. Cette hypothèse formalise précisément la raison pour laquelle l'algorithme de Hellman–Reyneri est heuristique.

3.2.2 Logarithme individuel

Une fois que les logarithmes des éléments de $\mathcal{F}$ sont connus, et puisque nous cherchons le logarithme d'un élément arbitraire h dans $\mathbb{F}_{q^k}^*$, il suffit d'exhiber une relation supplémentaire de la même forme que (3.2) mais faisant cette fois-ci intervenir h. Par exemple, nous pouvons chercher un élément $h \cdot g^r$ qui se factorise uniquement en termes appartenant à $\mathcal{F}$ et déduire ensuite le logarithme de h en soustrayant r à la somme des logarithmes des éléments de la factorisation. Ainsi, le calcul d'un logarithme discret du corps fini, une fois le précalcul des logarithmes de la base de friabilité effectué, coûté bien moins cher que de ce dernier. En pratique, il s'agit d'une remarque de grande importance : c'est pour cette raison que le plupart des algorithmes de logarithmes discrets présentent une phase de logarithme individuel séparée. Les algorithmes rigoureux sont une exception à cette règle. En effet, les considérations pratiques ne rentrent pas en compte dans leur construction. On préfère alors ne pas s'encombrer d'une étape supplémentaire séparée dont l'analyse rigoureuse complexifierait encore une démonstration qui sait l'être déjà.

Mise à l'écart des petits facteurs. Afin de simplifier l'exposition de notre propos, la présentation ci-dessus diffère de la manière classique d'aborder les algorithmes par calcul d'indice. Normalement, les éléments de $\mathbb{F}_q$ ne sont pas inclus dans la base de friabilité $\mathcal{F}$. Au lieu de cela, ils sont simplement écartés des équations. Cette remarque provient du fait que ces éléments qui sont donc constants possèdent un ordre divisant $q-1$, ce qui implique en particulier que leurs logarithmes sont des multiples de $(q^k-1)/(q-1)$. Nous les traitons donc à part en nous restreignant simplement au calcul de logarithmes modulo $(q^k-1)/(q-1)$.

De manière plus générale, tous les petits facteurs de q^k-1, et en particulier l'ordre $q-1$, sont habituellement écartés des considérations pratiques lorsque nous effectuons les phases d'algèbre linéaire. Deux raisons sont à mentionner. La première est un souci de diminution du nombre d'obstructions possibles : en effet, ces petits facteurs peuvent contrarier l'exécution de l'algèbre linéaire puisqu'ils augmentent la possibilité d'apparition d'éléments non inversibles

lors de cette phase. La seconde réside dans l'opportunité de recourir à une technique qui rend justement obsolète ce souci potentiel. Grâce à l'utilisation des algorithmes génériques comme ceux de Pohlig–Hellman et Rho de Pollard, nous savons en réalité facilement retrouver la part manquante des logarithmes.

3.2.3 Analyse de complexité

Puisque la phase de calcul de logarithme discret individuel est négligeable comparée aux précalculs, il suffit, pour analyser la complexité asymptotique d'un tel algorithme, d'exprimer le coût de la collecte des relations et de l'algèbre linéaire en fonction du paramètre d. En majorant par q^{d+1} la cardinalité de $\mathcal{F}$ et en désignant par $\mathcal{P}$ la probabilité qu'un polynôme aléatoire G_r soit d-friable, le coût de l'exécution de la phase de collecte des relations est alors majoré par $q^{d+1}/\mathcal{P}$. En ce qui concerne l'algèbre linéaire, nous remarquons dans un premier temps que le nombre d'éléments de la base de friabilité dans n'importe laquelle des relations formées est majorée par k. Ceci provient du fait que le nombre de polynômes irréductibles qui apparaissent lorsque l'on factorise un polynôme est toujours inférieur à son degré. Aussi, l'algèbre linéaire que l'on exécute s'attache toujours à une matrice très creuse qui ne contient au plus que k termes non nuls par ligne. Le coût d'une telle méthode est alors quadratique en la taille de $\mathcal{F}$, si l'on utilise des techniques creuses comme présentées plus haut. Plus précisément, le temps de calcul de cette étape peut être majoré par $k\,q^{2d+2}$.

De l'estimation 3.1.1 nous pouvons écrire :

$$-\log_q \mathcal{P} \leq \frac{k}{d} \cdot \log_q(k/d).$$

Par conséquent, en choisissant notre paramètre de la façon suivante :

$$d = \left\lceil \sqrt{\frac{k \log_q k}{2}} \right\rceil,$$

nous obtenons une complexité totale de :

$$q^{\sqrt{(2+o(1))\,k \log_q k}} = L_{q^k}(1/2, \sqrt{2}).$$

3.3 L'algorithme de Coppersmith en $L_{q^k}(1/3)$

En 1984, l'algorithme de Coppersmith [Cop84] est le premier à atteindre une complexité inférieure à $L_{q^k}(1/2)$ pour le calcul de logarithme discret dans

un corps fini. Cette nouvelle méthode est alors illustrée par un record de calcul sur le corps binaire $\mathbb{F}_{2^{127}}$. Sous sa forme initiale, il s'applique uniquement aux corps de caractéristique 2, mais il est rapidement généralisé à n'importe quelle caractéristique [BMV84] dans le courant de la même année. L'idée clef qui permet de dépasser l'algorithme de Hellman–Reyneri consiste à utiliser la liberté que nous avons dans la représentation de $\mathbb{F}_{q^k}$ pour choisir l'une de celles qui facilitent la création des relations.

3.3.1 Représentation du corps cible

Coppersmith propose de choisir le polynôme irréductible qui définit son corps fini $\mathbb{F}_{q^k}$ de la forme :

$$X^k - S(X),$$

où S est un polynôme de degré d_S aussi petit que possible. Dans la suite de cette section nous remplaçons le choix de ce polynôme par une construction plus récente proposée dans [Jou13b], qui simplifie la présentation de la construction des relations multiplicatives sans dénaturer l'idée initiale de Coppersmith. Supposons maintenant que n soit l'unique entier naturel vérifiant la double inégalité $q^{n-1} < k \leq q^n$ et choisissons un polynôme $S(X)$ de petit degré tel que $X^{q^n} - S(X)$ n'ait qu'un unique facteur irréductible de degré k. Nous appelons $I_k(X)$ ce facteur, et, sans surprise, c'est lui que nous choisissons pour définir le corps fini $\mathbb{F}_{q^k}$.

Nous décomposons maintenant n en $n = n_1 + n_2$ pour des valeurs n_1 et n_2 qui seront déterminées lors de l'analyse de complexité. La construction des relations tourne autour de l'idée clef qu'à chaque paire de polynômes A et B à coefficients dans $\mathbb{F}_q$, nous avons :

$$\left(A(X) + X^{q^{n_1}} B(X)\right)^{q^{n_2}} = A(X^{q^{n_2}}) + S(X)B(X^{q^{n_2}}) \mod I_k(X), \tag{3.3}$$

grâce à la linéarité de l'élévation à la puissance q des termes, ainsi qu'à l'égalité $X^{q^n} = S(X) \mod I_k(X)$.

3.3.2 Création des relations

En définissant la base de friabilité $\mathcal{F}$ comme dans la section 3.2, nous remarquons que l'équation (3.3) entraine une relation multiplicative satisfaisante dès lors que les polynômes $A(X) + X^{q^{n_1}} B(X)$ et $A(X^{q^{n_2}}) + S(X)B(X^{q^{n_2}})$ sont d-friables. A ce point, il est classique de prétendre que cet évènement a lieu lorsque le produit :

$$\left(A(X) + X^{q^{n_1}} B(X)\right) \cdot \left(A(X^{q^{n_2}}) + S(X)B(X^{q^{n_2}})\right)$$

est d-friable, et que la probabilité d'un tel évènement est comparable à la friabilité d'un polynôme aléatoire du même degré. Encore une fois, l'heuristique sous-jacente à l'algorithme se dissimule précisément dans cette dernière hypothèse.

Continuons. Nous étudions maintenant les paires de polynômes non nuls A et B de degré au plus d, telles que A soit unitaire. Cette dernière restriction provient du fait que la multiplication des deux polynômes A et B par une même constante du corps de base $\mathbb{F}_q$ produira une équation identique. Nous ôtons ainsi à A son coefficient de tête afin d'éviter les redondances. Nous cherchons maintenant une paire (n_1, n_2) d'entiers naturels qui minimise le degré du produit ci-dessus donné. Puisque le degré total est :

$$d + q^{n_1} + d_S + \frac{d \cdot q^n}{q^{n_1}},$$

il est donc minimisé lorsque q^{n_1} est aussi proche que possible de $\sqrt{d\,q^n}$. Un tel choix entraine un degré asymptotique de l'ordre de $(2 + o(1))\sqrt{d\,q^n}$. Exprimé en fonction de k, celui-ci varie alors entre $(2 + o(1))\sqrt{d\,k}$ et $(2\sqrt{q} + o(1))\sqrt{d\,k}$, selon la manière dont k est proche ou non de la prochaine puissance de q.

Finalement, d est choisi de la même manière qu'à la section 3.2 en équilibrant l'opposé du logarithme de la probabilité de succès et la valeur de d. Dit autrement nous prenons :

$$d = (4/3)^{1/3}\, q^{n/3}\, n^{2/3}.$$

La complexité devient donc : $L_{q^{q^n}}(1/3, (32/9)^{1/3})$. Exprimé en fonction de q^k, celle-ci varie alors entre $L_{q^k}(1/3, (32/9)^{1/3})$ et $L_{q^k}(1/3, (32\,q/9)^{1/3})$, de nouveau selon la manière dont k est proche ou non de la prochaine puissance de q. Nous soulignons cependant que la version simplifiée que nous venons de présenter aboutit sur un pire cas lui-même un peu dégradé par rapport à la version initiale de l'algorithme de Coppersmith.

3.3.3 A propos de l'algèbre linéaire

Comparé à l'algorithme de Hellman et Reyneri, celui de Coppersmith se singularise lors de la phase d'algèbre linéaire par deux différences importantes. La première concerne le type des équations qui apparaissent : dans la version de Coppersmith, c'est-à-dire lors de la création de relations de la forme (3.3), les termes constants ne sont pas présents.

Ces relations incluent en revanche de gros coefficients q^{n_2}. En effet, chacune des relations multiplicatives de la forme (3.3) se traduit en terme de

logarithme discret par une égalité linéaire du type :

$$q^{n_2} \sum_{g_i \in \mathcal{F}} e_i \log_g(g_i) = \sum_{g_i \in \mathcal{F}} e'_i \log_g(g_i) \mod q^k - 1.$$

Par conséquent, le système admet plusieurs solutions. Fort heureusement, n'importe laquelle de ces solutions (à l'exception du vecteur nul) nous permettra de retrouver les logarithmes, en multipliant simplement par la bonne constante. Par exemple, si le générateur g lui-même apparait dans la base de friabilité $\mathcal{F}$, il est facile de renormaliser le vecteur solution pour obtenir les logarithmes en base g. Si g est en dehors de la base en revanche, nous procédons alors à une phase de logarithme individuel sur g pour déceler la valeur correcte de la constante de normalisation.

Par ailleurs, les gros coefficients q^{n_2} ont un impact sur la performance des multiplications matrice-vecteur et dégradent ainsi le processus d'algèbre linéaire. On pourrait s'attendre, pour chaque entrée d'un produit matrice-vecteur en cours, à devoir effectuer autant de multiplications de gros entiers qu'il y a de gros coefficients à l'intérieur de la ligne correspondante à l'entrée du vecteur que l'on calcule. Cependant, puisque ce coefficient de grande taille est unique, nous pouvons regrouper les termes à multiplier en amont, de sorte qu'une seule telle grosse multiplication par ligne suffise au produit matrice-vecteur en cours. Notons que la moitié environ des coefficients non nuls de la ligne sont des multiples de q^{n_2}. Enfin, dans le cas fréquent des corps binaires – donc de caractéristique 2 – cette multiplication correspond simplement à une décalage vers la gauche de n_2 bits. Aussi, elle ne ralentit pas réellement le procédé.

3.3.4 Logarithme individuel et naissance de la descente

L'une des difficultés de l'algorithme de Coppersmith se joue lors de la construction d'une relation supplémentaire finale faisant intervenir un élément arbitraire du corps fini. Contrairement à l'algorithme de Hellman et Reyneri, la génération d'une telle relation n'est plus aussi aisée. Aussi, une nouvelle notion essentielle apparait : il s'agit de la notion de descente. Ce procédé commence par écrire une relation entre notre élément arbitraire vu comme un polynôme en α, l'élément primitif du corps cible, et un petit nombre d'autres polynômes auxiliaires de degré plus petit. Par itérations successives, il devient alors possible d'exprimer le polynôme initial en produit d'éléments de la base de friabilité $\mathcal{F}$. Par conséquent, le calcul d'un logarithme discret est maintenant représenté par un arbre dont les fils d'un noeud qui correspond à un polynôme sont les noeuds des polynômes de plus bas degré apparaissant dans

la relation associée. Toutes les feuilles de l'arbre doivent être dans $\mathcal{F}$. Enfin, le coût d'un tel calcul est déterminé en prenant soin de borner le nombre de noeuds de l'arbre, ainsi que le coût de la recherche d'une relation entre un noeud et ses fils. Une telle analyse dépasse le cadre de ce chapitre mais le résultat important à retenir se concrétise autour du fait que la totalité du processus de descente reste négligeable comparé au coût de calcul initial des logarithmes des éléments de $\mathcal{F}$.

Pour trouver une relation impliquant un polynôme $h(\alpha)$, qui peut être supposé irréductible sans perte de généralité, nous procédons alors de la façon suivante :

— Nous choisissons n_1 et n_2 deux entiers adaptés au degré de h, et tel que $n = n_1 + n_2$.

— Nous remarquons que l'ensemble des polynômes (A, B) tel que h divise $A(X) + X^{q^{n_1}} B(X)$ forme un réseau. Aussi nous calculons une base (A_1, B_1) et (A_2, B_2) de ce réseau.

— Pour une paire de polynômes de petit degré (λ_1, λ_2), posons $A = \lambda_1 A_1 + \lambda_2 A_2$ et $B = \lambda_1 B_1 + \lambda_2 B_2$ et considérons la relation potentielle :

$$\left(A(X) + X^{q^{n_1}} B(X)\right)^{q^{n_2}} = A(X^{q^{n_2}}) + S(X)B(X^{q^{n_2}}) \mod I_k(X).$$

Par linéarité, h divise alors le polynôme $A(X) + X^{q^{n_1}} B(X)$.

— Parmi toutes ces relations candidates qui contiennent toutes h, nous ne gardons que celles dont les degrés de tous les polynômes irréductibles impliqués sont plus petits que $\kappa \deg h$ pour une certaine constante $\kappa < 1$ (un choix classique consisterait à prendre $\kappa = 3/4$).

La descente procède ainsi récursivement jusqu'à atteindre uniquement des éléments de la base de friabilité, c'est-à-dire des polynômes de degré inférieur ou égal à d.

3.4 Des polynômes univariés aux polynômes bivariés

L'algorithme qui fait suite à celui de Coppersmith est le crible par corps de fonction (FFS) proposé en 1999 par Adleman et Huang [AH99]. La figure 3.1 illustre l'intérêt pratique de cette méthode, utilisée plusieurs fois comme levier pour atteindre de nouveaux records de calculs de logarithmes. Nous présentons ici une variation de l'algorithme originel, parue dans [JL06] et qui, tout comme la variante de Coppersmith que nous avons proposée, n'emploie que des polynômes et restitue l'idée initiale des auteurs. Poussée à son maximum, cette variante est celle utilisée à l'été 2012 pour calculer des logarithmes discrets dans un corps de 923-bit de caractéristique 3, en 900 000 heures-CPU

environ [SSHT14].

3.4.1 Egalités entre polynômes univariés d'inconnues différentes

Le point de départ de l'algorithme réside une nouvelle fois dans la manière de définir le corps cible. L'idée consiste implicitement à construire l'extension à l'aide de deux relations polynômiales bivariées sur $\mathbb{F}_q$:

$$Y = f(X) \quad \text{et} \quad X = g(Y).$$

En assemblant ces deux relations, nous obtenons $X = g(f(X))$. Ainsi, si le polynôme $g(f(X)) - X$ possède un facteur irréductible $I_k(X)$ de degré k, nous pouvons représenter $\mathbb{F}_{q^k}$ à l'aide de celui-ci. Sous l'hypothèse bien sûr que le produit des degrés de f et g soit plus grand que k, nous espérons heuristiquement trouver une bonne paire de tels polynômes f et g. Ceci étant fixé, nous désignons par α l'un des racines dans $\mathbb{F}_{q^k}$ du polynôme $I_k(X)$ et nous posons $\beta = f(\alpha)$. Par construction nous avons donc $\alpha = g(\beta)$ dans $\mathbb{F}_{q^k}$.

A partir de n'importe quelle paire de polynômes A et B univariés nous pouvons alors écrire :

$$\begin{aligned} A(\alpha) + \beta\, B(\alpha) &= A(\alpha) + f(\alpha)\, B(\alpha) \quad \text{et} \\ A(\alpha) + \beta\, B(\alpha) &= A(g(\beta)) + \beta\, B(g(\beta)). \end{aligned}$$

Ceci constituera la base de nos relations.

En assemblant cette fois-ci les parties droites entre-elles, nous obtenons une égalité dans $\mathbb{F}_{q^k}$ entre un premier polynôme en α et un second en β. De plus, chacun de ces deux polynômes est de degré relativement petit, si les paramètres sont bien choisis. Par conséquent, si chacun d'entre eux se factorise en produits de polynômes irréductibles de degré au plus d, nous pouvons exploiter cette relation multiplicative. En comparaison avec les algorithmes précédents, nous soulignons que nous avons ici doublé la taille de la base de friabilité, puisqu'elle contient maintenant l'évaluation de chaque polynôme irréductible de petit degré en α et en β.

3.4.2 Complexité asymptotique

L'analyse de la complexité s'articule autour de l'optimisation simple du paramètre d qui borne les degrés des polynômes irréductibles de la base de friabilité $\mathcal{F}$. Sans surprise, les polynômes A et B sont aussi choisis de sorte que leurs degrés soient au plus d, tandis que les degrés de f et g sont sélectionnés

pour satisfaire $\deg f \cdot \deg g \approx k$ et $\deg f / \deg g \approx d$. L'optimisation que nous ne détaillons pas ici entraine une complexité asymptotique heuristique de :

$$L_{q^k}\left(1/3, (32/9)^{1/3}\right).$$

Cette représentation bivariée fait elle aussi intervenir un processus de descente similaire à celui des algorithmes de la section 3.3, dont le coût est lui aussi négligeable comparé au précalcul des logarithmes discrets des éléments de la base de friabilité.

3.5 Changement de paradigme

3.5.1 Construction et non plus recherche de polynômes friables

Bien que différents, les algorithmes des sections 3.3 et 3.4 partagent un principe commun : chacun d'entre eux crée des polynômes de degré moyen qui apparaissent respectivement des deux côtés d'une relation multiplicative, tout en espérant que ces deux polynômes se factorisent simultanément en polynômes irréductibles de petit degré. L'obstruction fondamentale qui s'élève ici se tient dans l'impossibilité qu'ont ces deux polynômes de taille moyenne à présenter des petits degrés. En effet, la relation qui lie ces deux polynômes encode d'une certaine manière la représentation du corps cible. A cause de cette remarque, la baisse du degré d'un des deux polynômes entrainerait la hausse du degré du second. Cependant, puisque la probabilité d'obtenir de bonnes relations dépend directement de la chance que l'on a d'avoir des polynômes friables, donc de degré pas trop élevé, la meilleure stratégie consiste à équilibrer ces degrés. Descendre l'un artificiellement ne mène donc nulle part. Les degrés des deux polynômes sont ainsi bloqués, de sorte que la variante bivariée que nous avons présentée semble être, dans ce contexte, la meilleure qui soit.

Pour échapper à cette obstruction il faut tenter une nouvelle approche, et abandonner cette idée de tester au hasard la friabilité de deux polynômes de taille moyenne. Une possibilité récemment découverte [Jou13b] a permis de briser la barrière en $L(1/3)$ en s'appuyant sur des relations qui font intervenir d'un côté un polynôme de haut degré, de l'autre un polynôme de petit degré, mais telle que le polynôme de haut degré soit construit pour que sa friabilité soit avérée. Cette idée a d'abord mené à un algorithme en $L(1/4)$ amélioré par la suite en une méthode de complexité quasipolynomiale pour les corps de caractéristique fixée [BGJT14]. L'idée séminale de la construction de polynômes friables – et non plus de leur recherche – se trouve dans une version moins efficace de l'algorithme [Jou13a]. Indépendamment, de son amélioration

en $L(1/4)$ publié dans [Jou13b] et qui construit donc des polynômes de hauts degrés friables, [GGMZ13] considère une famille de polynômes de hauts degrés contenant une grande proportion de polynômes avec cette même propriété. Cette seconde proposition ne mène qu'à un algorithme en $L(1/3)$ mais ces deux variantes ont permis toutefois l'établissement de records de même envergure. Actuellement, le plus grand corps fini de petite caractéristique pour lequel le problème du logarithme discret a été brisé grâce à cette nouvelle génération d'algorithmes, que l'on appelle les algorithmes par Représentation de Frobenius, est une extension de Kummer exposant une cardinalité de 9234-bits.

3.5.2 $\mathbb{F}_q$ est le corps de décomposition du polynôme $X^q - X$

L'identité phare de la construction de polynômes friables de haut degré est la suivante :

$$X^q - X = \prod_{\gamma \in \mathbb{F}_q} (X - \gamma). \tag{3.4}$$

Il s'agit de généraliser maintenant l'idée d'une relation dans laquelle un polynôme de haut degré, ici $X^q - X$, se factorise en produits de polynômes de petit degré, ici, l'intégralité des polynômes unitaires linéaires à coefficients dans le corps de base. Si l'on considère deux polynômes A et B non nuls à coefficients dans $\mathbb{F}_q$ nous pouvons factoriser :

$$\begin{aligned} A(X)^q B(X) - A(X) B(X)^q &= B(X)^{q+1} \left(\left(\frac{A(X)}{B(X)} \right)^q - \frac{A(X)}{B(X)} \right) \\ &= B(X)^{q+1} \prod_{\gamma \in \mathbb{F}_q} \left(\frac{A(X)}{B(X)} - \gamma \right) \\ &= B(X) \prod_{\gamma \in \mathbb{F}_q} (A(X) - \gamma B(X)). \end{aligned}$$

Ansi, $A(X)^q B(X) - A(X) B(X)^q$ est construit comme un produit de polynômes de degrés au plus égal au maximum des degrés de A et B. Pour utiliser ce polynôme dans une relation multiplicative, nous cherchons à le relier à un polynôme de petit degré dans le corps fini cible. Avec cette optique en tête nous utilisons donc dans un premier temps la linéarité du Frobenius pour écrire :

$$A(X)^q B(X) - A(X) B(X)^q = A(X^q) B(X) - A(X) B(X^q).$$

C'est alors qu'interviennent les souvenirs que nous avons de l'algorithme de Coppersmith, dans lequel il était naturel de construire le corps cible $\mathbb{F}_{q^k}$

via la recherche d'un polynôme S de petit degré tel que $X^q - S(X)$ possède un facteur irréductible $I_k(X)$ du bon degré, c'est-à-dire de degré k. De nouveau, cette hypothèse requiert $q \geq k$. Appelons α une racine de I_k. Il est maintenant possible de faire intervenir un polynôme friable construit (à gauche) dans la même équation qu'un polynôme de petit degré (à droite) en s'appuyant sur une relation de la forme :

$$B(\alpha) \prod_{\gamma \in \mathbb{F}_q} (A(\alpha) - \gamma\, B(\alpha)) = A(S(\alpha))\, B(\alpha) - A(\alpha)\, B(S(\alpha)), \tag{3.5}$$

Plus précisément, si A et B sont de degré au plus d, alors le degré du polynôme de droite est majoré par $d\,(\deg S + 1)$.

3.5.3 Lorsque la descente domine

Comme dans les algorithmes précédents, le paramètre d régit la taille de la base de friabilité. La différence fondamentale tient au fait que ce nouveau contexte permet de fixer d égal à une petite constante. Par conséquent, le précalcul des logarithmes des éléments de la base devient polynomial en temps, et c'est alors la phase de logarithme discret individuel qui domine asymptotiquement ! Différentes méthodes ont permis de rendre la descente quasipolynomiale, citons [BGJT14] puis [GKZ14b], qui achèvent tous deux des complexités de la forme :

$$q^{O(\log(\max(q,k)))}$$

pour n'importe quel corps fini $\mathbb{F}_{q^k}$. Cette complexité est moins bonne que polynomiale mais incroyablement meilleure qu'une complexité sous-exponentielle en $L_{q^k}(1/3)$. Pour finir, quand bien même le terme de *quasipolynomial* à fait grand bruit et sert encore à désigner parfois l'intégralité de ces algorithmes, il ne serait pas honnête de laisser croire que *ces algorithmes sont quasipolynomiaux pour tous les corps finis de petite caractéristique.* En effet, si la définition de petite caractéristique est claire, elle correspond à une plage bien plus large que celle, étroite, où les algorithmes par Représentation de Frobenius se montrent quasipolynomiaux. Par définition, un algorithme qui calcule des logarithmes discrets sur $\mathbb{F}_Q$ est quasipolynomial si sa complexité est en $L_Q(o(1))$. Or les variantes que nous avons évoquées atteignent une complexité asymptotique que l'on peut réécrire en :

$$L_Q(c + o(1))$$

lorsque la caractéristique est $L_Q(c)$. Sous cette forme, nous voyons clairement que cette complexité est meilleure que les complexité en $L_Q(1/3)$ des

générations précédentes pour tous les corps finis de petite caractéristique ; cependant, elle n'est *quasipolynomiale* que pour les corps dont la caractéristique est en $L_Q(o(1))$, c'est-à-dire presque fixée.

Remarque 3.5.1. Afin d'augmenter la flexibilité pratique de ces algorithmes, la plupart des articles relatifs remplacent le choix du polynôme $I_k(X)$ comme un des facteurs de $X^q - S(X)$, par le choix d'un polynôme divisant $h_1(X)X^q - h_0(X)$. Ainsi, la partie droite des équations de la forme (3.5) devient :

$$A\left(\frac{h_0(\alpha)}{h_1(\alpha)}\right) B(\alpha) - A(\alpha) B\left(\frac{h_0(\alpha)}{h_1(\alpha)}\right),$$

c'est-à-dire qu'il s'agit d'une fraction rationnelle dont le dénominateur est fixé et pour laquelle le numérateur est un polynôme de petit degré. Nous étudions les détails de ces relations dans le chapitre suivant.

Chapitre 4

Le Simply : un algorithme par représentation de Frobenius simplifié

Sommaire

Dans ce chapitre, nous proposons une description simplifiée des algorithmes par représentation de Frobenius qui, adjointe à quelques idées nouvelles, permet d'améliorer la complexité des phases polynomiales de ces algorithmes. Notre méthode réduit ainsi le coût de ces étapes à $O(q^6)$, où q est l'ordre du corps de base en question. Cette complexité asymptotique est à comparer avec la meilleure complexité publiée jusqu'à lors pour ces mêmes précalculs, qui étaient en $O(q^7)$. Nos paramètres simplifiés autorisent ainsi un corps fini de petite caractéristique général à présenter une complexité aussi basse que celle connue jusqu'ici pour les extensions de Kummer, ou de Kummer tordues, pour ces mêmes étapes.

Cette nouvelle méthode vient ainsi s'inscrire dans un des trois axes d'améliorations possibles de cette famille d'algorithmes. Si plusieurs questions restent ouvertes, d'un point de vue théorique il serait agréable de pouvoir faire abstraction des hypothèses heuristiques qui persistent. Un premier pas en ce sens a été effectué dans l'article [GKZ14b], qui se propose d'effectuer uniquement l'étape de descente et dont l'idée principale repose sur la possibilité

que nous avons de descendre les éléments du corps cible définis comme des polynômes de degré pair $2D$ vers des polynômes de degré D. Bien entendu, il serait aussi remarquable d'un point de vue théorique de pouvoir achever l'abaissement des complexités et descendre d'un algorithme quasipolynomial vers un algorithme réellement polynomial. D'un point de vue pratique en revanche, les étapes de l'algorithme qui consomment trop et limitent les records actuels dans le cas de corps cibles généraux – par opposition aux extensions particulières de Kummer par exemple – sont celles sous-jacentes aux calculs des logarithmes discrets des éléments de la base de friabilité. Imaginons que nous travaillions avec le corps de base $\mathbb{F}_q$. Alors, si la meilleure complexité pour ces phases est en $O(q^7)$ jusqu'à lors – nous renvoyons par exemple à [AMOR14] – certains auteurs mentionnent toutefois une complexité plus élevée. Ainsi, le record reporté dans [GKZ14a] pour un corps de base de cardinalité $q = 2^6$ demande une étape d'algèbre linéaire sur une matrice dont la dimension est réduite de q^4 to $q^4/24 = q^4/\log_2(q^4)$. Une telle approche mène à une complexité asymptotique en $O(q^9/\log(q)^2)$ qui, seulement dans le cas très spécifique des extensions de Kummer, peut être abaissée à $O(q^6)$.

Comme annoncé, nous cherchons dans ce chapitre à atteindre une complexité de :

$$O(q^6)$$

pour le cas général. L'idée autour de laquelle nous travaillons a été évoquée pour la première fois oralement à la conférence DLP 2014. Elle n'y était cependant présentée que sous les traits d'une version simplifiée permettant de mieux comprendre les mécanismes sous-jacents aux FRA (de l'anglais *Frobenius Representation Algorithms*) mais qui en dégradait les performances. La raison principale de cette dégradation apparente reposait sur le fait que, si l'utilisation de polynômes de degré inférieur à δ sur $\mathbb{F}_q$ semble équivalente à celle de polynômes linéaires sur $\mathbb{F}_{q^\delta}$, cette méthode ne permet le calcul de logarithmes discrets que pour $\mathbb{F}_{q^k}$ – avec k de l'ordre de grandeur de q – tandis que les précédentes variantes pouvaient atteindre le corps cible $\mathbb{F}_{q^{\delta k}}$. Nous perdons donc cet exposant δ qui apparaissait essentiellement sans surcoût dans les approches classiques, δ étant un tout petit entier habituellement compris entre 2 et 4. Une correspondance de même nature entre polynômes de petit degré sur une grande extension et polynômes de haut degré sur une plus petite extension régit en particulier les travaux présentés par [GKZ14b].

Pour rendre l'algorithme efficace il faut rendre le degré d le plus bas possible. Au premier regard, il semble qu'une valeur de $d = 3$ au minimum soit requise pour pouvoir achever les calculs. En réalité, notre approche simplifiée permet d'expliquer pourquoi il est possible, sous des hypothèses heuristiques

raisonnables, de réduire le degré des polynômes de la base de friabilité qui sont à coefficients dans $\mathbb{F}_q$, et, plus précisément, de leur imposer de ne pas dépasser $d = 2$. Une fois cette première base de friabilité calculée – c'est-à-dire une fois que nous connaissons les logarithmes discrets de ses éléments, ce que nous pouvons achever pour un coût de $O(q^5)$ – nous utilisons celle-ci comme levier pour étendre la base au degré $d = 3$ puis $d = 4$, pour un coût total de $O(q^6)$. L'utilisation de la descente quasipolynomiale heuristique de [BGJT14] ou de l'alternative proposée dans [GKZ14b], permet de descendre les éléments arbitraires de $\mathbb{F}_{q^k}$ vers la base de friabilité étendue formée des polynômes irréductibles de degré inférieur à 4.

Organisation du chapitre. Nous introduisons la représentation de notre corps cible en section 4.1 pour présenter ensuite en section 4.2 le calcul des logarithmes discrets de la base de friabilité simple et étendue. La section 4.3 récapitule l'analyse de la complexité asymptotique totale que nous parvenons à abaisser. Finalement, la section 4.4 illustre l'efficacité de l'algorithme par un calcul pratique de logarithme dans le cas général d'une extension de degré première qui ne divise pas[1] $q(q+1)(q-1)$. Plus précisément, nous attaquons le corps cible $\mathbb{F}_{q^k}$ où le corps de base est de cardinalité $q = 3^5$ et le degré d'extension est $k = 479$, qui n'est rien d'autre que le plus grand nombre premier inférieur à $2q$.

4.1 Petits polynômes de représentation du corps cible, petits polynômes de base de friabilité

Dans toute la suite $\mathbb{F}_{q^k}$ désigne le corps fini pour lequel nous souhaitons résoudre le problème du logarithme discret. Nous rappelons au paragraphe 4.1.1 comment les algorithmes par représentation de Frobenius représentent le corps cible à l'aide de deux polynômes h_0 et h_1, pour perfectionner au pragraphe 4.1.2 le choix de ces polynômes essentiels. La description de la base de friabilité constitue le sujet du dernier paragraphe 4.1.3 de cette section.

4.1.1 Les algorithmes par représentation de Frobenius

Comme tous les algorithmes de cette famille, notre méthode repose sur deux notions clefs. La première tient dans l'identité polynomiale bien connue

1. Les cas spéciaux connus qui présentent des facilités pour les records étant les extensions de Kummer de degré divisant $q-1$, celles de Kummer tordues de degré divisant $q+1$ et les extensions d'Artin-Schreier.

sur $\mathbb{F}_q[X]$:

$$\prod_{\alpha \in \mathbb{F}_q} (X - \alpha) = X^q - X. \tag{4.1}$$

La seconde consiste à définir le corps fini $\mathbb{F}_{q^k}$ via deux polynômes h_0 et h_1 de degré au plus H, c'est-à-dire de sorte qu'il existe un polynôme irréductible unitaire $I(X)$ de degré k sur $\mathbb{F}_q[X]$ tel que :

$$I(X) \text{ divise } h_1(X)X^q - h_0(X). \tag{4.2}$$

Si θ désigne une racine du polynôme I dans $\bar{\mathbb{F}}_q$, poser $\mathbb{F}_{q^k} = \mathbb{F}_q[X]/(I(X)) = \mathbb{F}_q(\theta)$ nous mène à une représentation du corps qui satisfait :

$$\theta^q = h_0(\theta)/h_1(\theta).$$

Puisque l'application qui élève un élément de la clôture algébrique $\bar{\mathbb{F}}_q$ à la puissance q s'appelle le Frobenius, ceci explique par ailleurs le nom donné à cette famille d'algorithmes.

La variante par représentation de Frobenius duale

Une alternative pour la construction du corps qui est notamment celle proposée dans [GKZ14a] impose une contrainte voisine sur le polynôme I. Ici :

$$I(X) \text{ divise } h_1(X^q)X - h_0(X^q).$$

L'avantage de cette option consiste à autoriser une plus grande plage de degré d'extention k pour un corps de base donné $\mathbb{F}_q$. Cependant, en ce qui nous concerne, l'utilisation d'un tel type de représentation complique quelque peu notre propos, comme nous le verrons. Dans ce contexte, la représentation du corps fini satisfait dorénavant :

$$\theta = h_0(\theta^q)/h_1(\theta^q).$$

Lorsque nous souhaiterons nous référer à cette variante, nous dirons que nous considérons la *représentation de Frobenius duale*, ou, de manière équivalente, la *représentation par Verschiebung.*

4.1.2 Choix simplifié des polynômes h_0 et h_1

Une construction *réellement simple*

Nous rappelons que le choix classique consiste à prendre deux polynômes quadratiques afin de pouvoir représenter, au moins de manière heuristique,

une grande plage de corps finis. Puisque l'utilisation de polynômes affines en lieu et place de h_0 et h_1 ne permet pas d'atteindre suffisamment d'extensions, mais qu'un petit sous-ensemble bien particulier d'entre eux, nous proposons de nous tenir précisément dans l'entre deux. Aussi, **nous sélectionnons un polynôme affine h_0 au côté d'un polynôme quadratique h_1.** Nous supposons par ailleurs que le terme constant de h_1 est nul. En factorisant par son coefficient de tête dans l'équation (4.2), nous pouvons donc supposer sans perte de généralité que ce polynôme h_1 est unitaire. Pour simplifier notre présentation nous écrirons dans la suite :

$$h_0(X) = rX + s \quad \text{et} \quad h_1(X) = X(X + t) \tag{4.3}$$

Une variante utile

Une seconde option naturelle consiste à choisir cette fois-ci h_0 comme un polynôme quadratique, de terme constant nul, et h_1 un polynôme affine. Cette construction qui est le symétrique de la précédente peut aussi être notée de manière plus explicite en posant :

$$h_0(X) = X(X + w) \quad \text{et} \quad h_1(X) = uX + v. \tag{4.4}$$

Rien n'indique à première vue que l'un des deux choix soit plus efficace ou plus simple à manipuler que le second. Nous verrons plus loin qu'ils sont en réalité équivalents en terme de complexité asymptotique. Cependant, les détails de la section 4.2 soulignent que la première des deux constructions données mène à une description plus simple de l'algorithme, aussi s'agira-t-il de celle que nous privilégierons. En guise de moyen mnémotechnique concernant le choix du nom des constantes, nous invitons le lecteur à remarquer que le triplet de coefficients $(\mathbf{r}, \mathbf{s}, t)$ correspond à la construction **r**éellement **s**imple, tandis que le triplet $(\mathbf{u}, \mathbf{v}, w)$ réfère à la **v**ariante **u**tile.

4.1.3 A la recherche d'une base de friabilité naturelle

La représentation du corps cible étant fixée, nous cherchons maintenant à déterminer notre base de friabilité. Dans un esprit de simplification de la description de notre méthode, nous choisissons de nous affranchir des polynômes à coefficients dans une extension.

Polynômes irréductibles à coefficients dans le corps de base

Nous fixons un paramètre d à déterminer plus tard et considérons la base la plus naturelle possible, à savoir l'ensemble des polynômes irréductibles

de degré inférieur à d sur $\mathbb{F}_q[X]$. Ceci diffère des algorithmes de la même génération qui choisissent plutôt une base de friabilité égale à un ensemble de polynômes irréductibles à coefficients dans une extension de $\mathbb{F}_q$ et non dans le corps de base. Afin d'engendrer une relation multiplicative, nous partons d'une paire de polynômes A et B de degrés inférieurs ou égaux à d, puis, en assemblant les équations (4.1) et (4.2) nous écrivons :

$$\begin{aligned} B(\theta)\prod_{\alpha\in\mathbb{F}_q}(A(\theta)-\alpha B(\theta)) &= B(\theta)A(\theta)^q - A(\theta)B(\theta)^q \\ &= B(\theta)A(\theta^q) - A(\theta)B(\theta^q) \\ &= B(\theta)A\left(\frac{h_0(\theta)}{h_1(\theta)}\right) - A(\theta)B\left(\frac{h_0(\theta)}{h_1(\theta)}\right). \end{aligned}$$

Par souci de clarté et de compacité des notations, nous faisons correspondre $B(\theta)$ avec le point à l'infini α de la droite projective $\mathbb{P}_1(\mathbb{F}_q)$. Cette correspondance permet alors de réécrire dans la suite le premier produit comme étant égal à :

$$\prod_{\alpha\in\mathbb{P}_1(\mathbb{F}_q)}(A(\theta)-\alpha B(\theta)).$$

Nous introduisons par ailleurs la notion de crochet suivante :

Définition 4.1.1. Soit D un entier naturel, et h_0, h_1, A, B quatre polynômes tels que A et B Soit de degré inférieur ou égal à D. On note alors $[A,B]_D$ et l'on appelle le D-crochet de A et B le polynôme défini tel que :

$$[A,B]_D(X) = h_1(X)^D\left(B(X)A\left(\frac{h_0(X)}{h_1(X)}\right) - A(X)B\left(\frac{h_0(X)}{h_1(X)}\right)\right).$$

Proposition 4.1.2. *Si h_0 et h_1 sont deux polynômes de degré au plus H et si A et B sont deux polynômes de degré au plus D alors :*

- *$[A,B]_D$ est un polynôme de degré au plus $(H+1)\cdot D$.*
- *L'application $[.,.]_D$ est bilinéaire et antisymétrique. En particulier, nous avons l'égalité $[A,A]_D = 0$.*

La preuve de chacun des deux points de la proposition est directe. Grâce à ces notations, nous pouvons maintenant alléger l'égalité et la réécrire comme :

$$\prod_{\alpha\in\mathbb{P}_1(\mathbb{F}_q)}(A(\theta)-\alpha B(\theta)) = \frac{[A,B]_d(\theta)}{h_1(\theta)^d}. \tag{4.5}$$

Puisque le numérateur $[A,B]_d$ du membre droit de l'équation (4.5) est de degré borné, sous des heuristiques classiques de comportement de ce polynôme,

la probabilité qu'il se factorise en un produit de polynômes irréductibles de degrés au plus d peut être minoré par une certaine constante que nous appellerons p_H. Si nous souhaitons plutôt employer la représentation de Frobenius duale nous obtenons de manière similaire la relation :

$$\prod_{\alpha \in \mathbb{P}_1(\mathbb{F}_q)} (A(\theta) - \alpha\, B(\theta)) = \left(\frac{[A,B]_d(\theta)}{h_1(\theta)^d} \right)^q. \tag{4.6}$$

Degré des polynômes de la base de friabilité

Le choix du paramètre d, essentiel à notre algorithme, nécessite au préalable de prendre en compte l'équilibre délicat des trois idées suivantes : en vue d'abaisser la complexité de l'algèbre linéaire, nous cherchons à définir une petite base de friabilité, c'est-à-dire que la valeur du paramètre d qui donnera le degré des polynômes de la base est souhaité le plus petit possible. Néanmoins, nous devons être en mesure dans un même temps d'engendrer suffisamment de bonnes équations pour construire notre système, puis, une fois ceci achevé, de descendre de grands polynômes vers ces polynômes particuliers de notre base. Ces deux dernières idées demandent donc une valeur de d plutôt élevée, afin d'augmenter la probabilité de friabilité des éléments testés, et de nécéssiter une descente vers un étage moins bas. Pour rappel, nous dirons d'une équation qu'elle est satisfaisante – ou tout simplement bonne – si elle se présente sous la forme de l'équation (4.5) tout en présentant deux membres pouvant s'écrire comme produits de polynômes de la base de friabilité uniquement.

L'obstacle de degré 3. Lorsque nous nous plaçons dans le cas très général de deux polynômes h_0 et h_1 de degré borné par H, l'analyse est la suivante. Le nombre d'équations qui peuvent être engendrées s'obtient par comptage du nombre de paires de polynômes (A, B) qui subsistent lorsque l'on prend en compte le fait que les paires Soit invariantes sous l'action de $\mathrm{PGL}_2(\mathbb{F}_q)$. En d'autres termes, en ignorant les cas pour lesquels le degré peut être réduit, comme expliqué à la remarque 4.1.3, nous pouvons supposer que :

$$A(X) = X^d + a(X) \quad \text{et} \quad B(X) = X^{d-1} + b(X),$$

où $a(X)$ et $b(X)$ sont de degré au plus $d-2$. Par conséquent, puisque les polynômes de degré $d-2$ ont $d-1$ coefficients, le nombre d'équations satisfaisantes qui peuvent être obtenues de cette manière est de l'ordre de $p_H \cdot q^{2d-2}$. De plus, le nombre d'éléments de la base de friabilité, c'est-à-dire le nombre de polynômes irréductible de degré au plus d, est proche de q^d/d. Or nous

souhaitons être en mesure d'écrire plus d'équations que d'inconnues, ce qui, traduit avec nos paramètres courants donne l'inégalité :

$$d \cdot p_H \cdot q^{d-2} \geqslant 1.$$

A moins de savoir accroitre considérablement la probabilité p_H, il est donc nécessaire de minorer le degré des polynômes de la base par $d \geqslant 3$, comme le souligne [GKZ14b].

Par conséquent, le meilleur espoir que nous ayons à propos de la complexité asymptotique du calcul des logarithmes des éléments de la base est de l'ordre de $(q^d)^2 \cdot q \geq q^7$. Pire, un rapide coup d'oeil aux records existants nous indique que cette complexité asymptotique en q^7 n'est pas toujours atteinte. Certains calculs nécessitent en réalité d'étendre la base de friabilité jusqu'au degré $d = 4$, ce qui élève la complexité de la phase associée à $O(q^9)$. Un tel élargissement est référencé par exemple dans [GKZ14a], qui parviennent malgré tout, grâce à une utilisation judicieuse de l'invariant de Galois, à réduire le coût de cette base étendue en regroupant les objets de degré 4 – c'est-à-dire dans leur contexte des polynômes quadratiques sur une extension de degré 2 – en des groupes constitués de 24 conjugués.

La raison d'être de cet élargissement tient en la connaissance actuelle que nous avons des techniques de descente du degré 4 vers le degré 4 : celles-ci ne permettent pas complètement de descendre du degré 4 au degré 3, puisque dans la plupart des cas seule une fraction des logarithmes des polynômes irréductibles de degré 4 peut s'obtenir de cette manière. Cette situation est similaire à celle reportée dans [AMOR14], où la moitié des polynômes quadratiques sur une extension cubique peuvent être dérivés d'une technique de descente vers les polynômes linéaires.

Remarque 4.1.3 (Action de $\mathrm{PGL}_2(\mathbb{F}_q)$ sur les polynômes). Nous détaillons ici la raison pour laquelle nous pouvons nous restreindre au cas des paires de polynômes de la forme $A(X) = X^d + a(X)$ et $B(X) = X^{d-1} + b(X)$, avec a et b des polynômes de degré $d-2$. Supposons que nous ayons à disposition une égalité entre deux polynômes A_0 et B_0 de degrés d. Nous pouvons tout d'abord supposer que ces deux polynômes sont unitaires en multipliant simplement l'équation (4.5) par l'inverse du produit de leurs termes de tête. De plus, grâce à la proposition 4.1.2 nous avons :

$$[A_0, B_0]_d = [A_0, B_0 - A_0]_d.$$

Aussi, nous pouvons remplacer B_0 par $B_1 = B_0 - A_0$. En l'absence de chute de degré inattendue, B_1 est de degré $d-1$. Nous considérons donc de nouveau

que ce polynôme est unitaire. Si le coefficient associé à X^{d-1} dans A_0 est noté a_{d-1}, nous pouvons alors écrire :

$$[A_0, B_1]_d = [A_0 - a_{d-1}B_1, B_1]_d.$$

Enfin, nous pouvons remplacer A_0 par un polynôme A_1 dont le coefficient devant le monôme X^{d-1} est nul. La nouvelle paire (A_1, B_1) engendre donc la même équation que la paire (A_0, B_0) ; elle est bien de la forme restreinte annoncée.

Remarque 4.1.4 (Relations manquantes). Notons qu'il manque certaines des relations pouvant être obtenues avec A et B quelconques, de degré plus petit par exemple. Néanmoins, la fraction manquante reste suffisamment petite pour ne par causer de difficultés avérées.

Saut d'obstacle. En suivant l'argumentation précédente, poser $d = 2$ donne environ $q^2/2$ polynômes irréductibles. En posant de plus $H = 2$, nous obtenons donc une valeur de p_H très en dessous de $1/2$. Aussi, en l'absence d'amélioration sur cette probabilité, le nombre d'équations que nous pouvons espérer écrire est trop petit comparé à celui des inconnues, et il est donc impossible de retrouver les logarithmes discrets des éléments de la base de cette manière.... Pourtant, contre toute attente, nous définissons bel et bien la base de friabilité comme étant égale à **l'ensemble des polynômes irréductibles de degré** 2 **dont les coefficients résident dans le corps de base** $\mathbb{F}_q$. Le paragraphe 4.2.1 qui suit détaille comment surmonter cet obstacle et retrouver malgré tout les logarithmes de la base de friabilité.

4.2 Calculs accélérés de la base de friabilité (étendue)

Dans cette section nous présentons deux contributions qui permettent de réduire le coût total des phases polynomiales des algorithmes de logarithme discret. La première, qui se trouve au paragraphe 4.2.1, décrit comment nous pouvons adapter l'utilisation de l'équation (4.5) pour exécuter un premier calcul de la base de friabilité initiale correspondant au degré $d = 2$ pour un coût de $O(q^5)$. Nous montrons ensuite au paragraphe 4.2.2 qu'une fois ce premier calcul effectué, l'élargissement de la base au degré $d = 3$ peut s'effectuer pour un coût réduit de $O(q^6)$, au lieu du $O(q^7)$ attendu.

La seconde contribution apparait au paragraphe 4.2.3. Il s'agit d'une nouvelle technique de descente qui ne requiert qu'un petit sous-ensemble des polynômes irréductibles de degré 4 pour calculer à la volée les logarithmes d'une partie impressionnante des autres polynômes du même degré. S'il reste

suffisamment de mémoire libre, il est aussi possible d'adapter cette technique pour obtenir les logarithmes correspondant à une base étendue à l'intégralité des polynômes de degré 4. Chacune des deux options peut être exécutée avec une complexité asymptotique temporelle en $O(q^6)$.

4.2.1 Une base réduite au degré 2

Comme évoqué précédemment, le choix d'une base de friabilité constituée des polynômes linéaires et quadratiques semble n'apporter qu'un nombre insuffisant d'équations satisfaisantes comparé au nombre des inconnues. Nous proposons deux outils pour contourner cet obstacle virtuel. Dans un premier temps, nous montrons comment la sélection de nos polynômes h_0 et h_1 plus petits qu'à l'habitude permet d'améliorer la valeur de p_H qui borne la probabilité d'obtenir une telle équation satisfaisante, en exhibant des facteurs systématiques. Par ailleurs, nous proposons une seconde manière d'obtenir de bonnes équations, afin de compléter le système. En guise d'effet secondaire agréable, cette seconde source de relations amène des équations encore plus creuses que celles de la forme (4.5).

Minoration plus fine de p_H via des facteurs systématiques

Une fois que nous avons fixé :

$$\begin{aligned} A(X) &= X^d + a(X) \\ \text{et} \quad B(X) &= X^{d-1} + b(X), \end{aligned}$$

nous constatons que le membre de gauche de l'équation (4.5) (ou (4.6)), tout comme le dénominateur du membre de droite, peut s'écrire comme produit d'éléments de la base de friabilité. Aussi, nous devons analyser la probabilité que le numérateur du membre de droite, c'est-à-dire le D-crochet de A et B, se factorise lui aussi comme produits de polynômes de degré au plus 2.

Le cas de la construction simple : h_0 affine et h_1 quadratique. La proposition 4.1.2 nous permet de majorer le degré de $[A, B]_d$ par $(H+1)\cdot d$. Par conséquent, pour des valeurs particulières de $H = 2$ et $d = 2$, ce degré est inférieur à 6. La probabilité qu'un polynôme aléatoire de degré 6 se factorise en terme de degré inférieur à 2 est bien trop basse pour que l'on puisse obtenir suffisamment d'équations. Aussi, comme mentionné dans [GKZ14b], nous remarquons qu'un terme systématique apparait toujours dans la factorisation de $[A, B]_d(X)$. Plus précisément, nous avons le résultat suivant :

Lemme 4.2.1 (Facteur systématique du D-crochet). *Soit A et B deux polynômes de degré au plus D. Alors $[A,B]_D(X)$ est un multiple de $Xh_1(X) - h_0(X)$.*

Démonstration. Par bilinéarité, si $A(X) = \sum_{i=0}^{D} a_i X^i$ et $B(X) = \sum_{i=0}^{D} b_i X^i$, nous pouvons écrire :

$$[A,B]_D = \sum_{i=0}^{D}\sum_{j=0}^{D} a_i b_j \left[X^i, X^j\right]_D.$$

Par ailleurs, puisque $[.,.]_D$ est bilinéaire et antisymétrique, il est clair que $\left[X^i,X^j\right]_D = -\left[X^j,X^i\right]_D$ et $\left[X^i,X^i\right]_D = 0$. Il suffit donc de considérer le D-crochet de X^i et X^j où $i < j$. Calculons maintenant :

$$\begin{aligned}
\left[X^i,X^j\right]_D &= h_1^{D-j}(X)\left(X^j h_0(X)^i h_1(X)^{j-i} - X^i h_0(X)^j\right) \\
&= h_1^{D-j}(X) X^i h_0(X)^i \left((Xh_1(X))^{j-i} - h_0(X)^{j-i}\right) \\
&= h_1^{D-j}(X) X^i h_0(X)^i (Xh_1(X) - h_0(X)) \sum_{k=1}^{j-i} h_0(X)^{k-1} (Xh_1(X))^{j-i-k}.
\end{aligned}$$

Nous en déduisons que le polynôme $Xh_1(X) - h_0(X)$ divise bien $\left[X^i,X^j\right]_D$, donc divise aussi $[A,B]_D$. □

Après avoir divisé le crochet $[A,B]_d$ par son facteur systématique de degré 3, la question qui subsiste consiste donc à déterminer si le polynôme de degré 3 restant se factorise en produits de termes de degré au plus 2 ou non. En supposant qu'il se comporte comme un polynôme aléatoire du même degré, nous pouvons maintenant minorer la probabilité par 2/3 – il s'agit simplement d'une application du lemme 3.1.2 démontré au chapitre précédent avec $D = 3$ et $d = 2$. Enfin, puisque cette probabilité est maintenant plus grande que 1/2 de manière certaine, nous savons que nous pouvons obtenir suffisamment d'équations pour calculer les logarithmes des éléments de la base initiale.

La variante utile : h_0 quadratique et h_1 affine. Il est facile de vérifier à nouveau que le numérateur du membre de droite de l'équation (4.5) ou (4.6) est systématiquement divisible par $\theta h_1(\theta) - h_0(\theta)$. Cependant, dans cette variante le degré de ce même polynôme est moins élevé que précédemment : il n'est que de degré 2. Si la valeur de p_H est partiellement améliorée, elle reste néanmoins trop faible pour obtenir un nombre suffisant d'équations.

Pour réduire le degré du polynôme restant d'un cran supplémentaire, nous devons remarquer que la borne sur le degré du crochet $[A,B]_d$ donné dans la

proposition 4.1.2, c'est-à-dire $(H+1)\cdot d$, peut en réalité être amélioré dans ce contexte spécifique, puisque le polynôme h_1 est ici affine. En vérité, le degré est maintenant majoré par $(H+1)\cdot d - 1$. De nouveau, pour $H = 2$ et $d = 2$, ceci réduit gratuitement le degré du polynôme résiduel de 6 à 5. En divisant ensuite par le facteur systématique de degré 2 qui provient du lemme 4.2.1, il reste comme précédemment un polynôme de degré 3. La probabilité p_H est encore une fois minorée par $2/3 > 1/2$.

Une seconde source de relations

Bien que les équations recueillies grâce à la nouvelle manière de choisir h_0 et h_1, que ce soit dans le cas de la construction simple ou de la variante utile, suffisent déjà à produire un système linéaire de taille adéquate pour une base de paramètre $d = 2$, proposer une autre source de relations peut s'avérer utile. Dans ce paragraphe, pour engendrer ces équations supplémentaires nous considérons simplement une variation des relations systématiques qui ont été introduites dans [BMV84] et souvent utilisées en pratique dans le cadre du crible par corps de fonctions.

Plus précisément, nous considérons $f(X) = X^2 + f_1 X + f_0$ un polynôme irréductible de degré 2 dans $\mathbb{F}_q[X]$. Nous pouvons écrire :

$$f(\theta)^q = f\left(\frac{h_0(\theta)}{h_1(\theta)}\right) = \frac{h_0(\theta)^2 + f_1\, h_0(\theta)h_1(\theta) + f_0\, h_1(\theta)^2}{h_1(\theta)^2}.$$

Le numérateur du membre de droite est un polynôme de degré 4, puisque l'un des deux polynômes h_0 ou h_1 est quadratique et que le second est affine. Nous remarquons que la moitié environ de ses numérateurs sont irréductibles tandis que le restant se factorisent en produits de deux polynômes irréductibles de degré 2 chacun. Dans le contexte de la représentation de Frobenius duale, les équations systématiques que nous pouvons écrire différent légèrement. Elles sont de la forme :

$$f(\theta) = f\left(\frac{h_0(\theta)}{h_1(\theta)}\right)^q = \left(\frac{h_0(\theta)^2 + f_1\, h_0(\theta)h_1(\theta) + f_0\, h_1(\theta)^2}{h_1(\theta)^2}\right)^q.$$

Le principe reste cependant identique. L'avantage de ces relations que nous pouvons systématiquement obtenir réside par ailleurs dans la facilité que nous trouvons à les généraliser à des polynômes irréductibles de degré supérieur, tout en conservant de nouveau cette répartition moitié irréductible - moitié produit d'irréductibles.

Lemme 4.2.2. *Soit h_0 et h_1 deux polynômes tels que l'un soit affine et l'autre quadratique. Si f est un polynôme irréductible unitaire et de degré D dans l'anneau $\mathbb{F}_q[X]$, alors $h_1(X)^{2D} f(h_0(X)/h_1(X))$ est un polynôme de degré $2D$ qui présente une probabilité égale à $1-p$ d'être irréductible et une probabilité p de se factoriser en deux polynômes irréductibles de degré moitié, c'est-à-dire D, avec :*

$$\frac{1}{q^D}\left(\frac{q^D-1}{2} - \frac{q^{\lfloor D/2\rfloor+1}-q}{q-1}\right) \leq p \leq \frac{q^D+3}{2\,q^D}.$$

Démonstration. Soit h_0 et h_1 deux polynômes tels que l'un soit affine et l'autre quadratique. Nous définissons le polynôme P via l'égalité :

$$P(X) = h_1(X)^{2D} f(h_0(X)/h_1(X)).$$

Notons qu'un élément α est une racine de P dans $\bar{\mathbb{F}}_q$ si et seulement si $h_0(\alpha)/h_1(\alpha)$ est une racine de f dans $\bar{\mathbb{F}}_q$.

Supposons alors que α soit une racine de P dans $\bar{\mathbb{F}}_q$ et que $\sigma, \sigma^q, \cdots, \sigma^{q(D-1)}$ désigne les racines conjuguées de f dans $\mathbb{F}_{q^D}$. Sans perte de généralité, nous pouvons de plus faire l'hypothèse que $h_0(\alpha)/h_1(\alpha) = \sigma$. Avec cette notation, nous déduisons que α est une racine de $h_0 - \sigma h_1 \in \mathbb{F}_{q^D}[X]$. Le polynôme $h_0 - \sigma h_1$ étant quadratique, il est soit irréductible dans $\mathbb{F}_{q^D}[X]$, soit produit de deux polynômes linéaires. Dans le premier des deux cas, α n'appartient pas à $\mathbb{F}_{q^D}$ et P est irréductible sur $\mathbb{F}_q[X]$. Dans le second cas, il possède deux racines α et α' possiblement égales et appartenant à $\mathbb{F}_{q^D}$. Le polynôme P se factorise alors en deux polynômes irréductibles de degré D.

Afin d'étudier la probabilité dans chacun des deux cas, nous reformulons le problème et remarquons que P se factorise en polynômes irréductibles de degré D lorsque σ est l'abscisse du point à coordonnées dans $\mathbb{F}_{q^D}$ défini sur la courbe :

$$\mathcal{C} : \quad h_0(Y) - X\,h_1(Y).$$

Ainsi définie, $\mathcal{C}$ est une courbe de genre 0 qui contient q^D+1 points définis sur $\mathbb{F}_{q^D}$, en comptant les (au plus 2) points à l'infini. Si (σ, α) est un point de $\mathcal{C}$, alors il existe un autre point (σ, α') de même abscisse, différent du premier, à moins que $h_0(Y) - \sigma\, h_1(Y)$ soit un carré. Nous soulignons qu'il y a au plus de 2 valeurs particulières de σ. Par conséquent, le nombre d'abscisses possible pour des points de $\mathcal{C}$ sur $\mathbb{F}_{q^D}$ appartient à l'intervalle $[\![(q^D-1)/2, (q^D+3)/2]\!]$.

Nous devons prendre en compte maintenant la restriction supplémentaire suivante : σ doit être une racine d'un polynôme irréductible de degré D et ne doit donc appartenir à aucun des sous-corps de $\mathbb{F}_{q^D}$. Pour comptabiliser aux mieux les bonnes valeurs de σ, nous bornons le nombre d'éléments appartenant à un sous-corps et leurs contributions possibles pour les abscisses

de $\mathcal{C}$. Tout d'abord, nous affirmons que si $\mathbb{F}_{q^i}$ est un sous-corps strict de $\mathbb{F}_{q^D}$ alors $i \leq D/2$. Puisqu'il existe au moins un sous-corps pour chaque degré, le nombre total d'éléments dans les sous-corps est majoré grossièrement par :

$$\sum_{i=1}^{\lfloor D/2 \rfloor} q^i = \frac{q^{\lfloor D/2 \rfloor + 1} - q}{q - 1}.$$

Nous en déduisons que le nombre d'abscisses possibles σ pour des points de $\mathcal{C}$ à coefficients dans $\mathbb{F}_{q^D}$ qui n'appartiennent pas à un sous-corps strict appartient à l'intervalle :

$$[\![\frac{q^D - 1}{2} - \frac{q^{\lfloor D/2 \rfloor + 1} - q}{q - 1}, \frac{q^D + 3}{2}]\!].$$

Ceci achève la preuve. □

En particulier, nous constatons que pour les polynômes irréductibles de degré 1, qui appartiennent donc à la base de friabilité initiale, nous obtenons toujours une équation systématique reliant d'un côté ce polynôme, et de l'autre, soit un polynôme quadratique, soit le produit de deux polynômes affines. Dans tous les cas, l'intégralité des termes qui apparaissent dans l'égalité appartiennent à la base de friabilité. Nous pouvons aussi utiliser ces équations systématiques pour des polynômes de plus haut degré, comme il sera proposé au paragraphe 4.2.3 pour faciliter le calcul des logarithmes des polynômes de degré 4.

4.2.2 Extension de la base de friabilité au degré 3

Afin de pouvoir élargir la base de friabilité à tous les polynômes irréductibles de degré inférieur à 3, sans dégrader l'algèbre linéaire par une exécution sur une matrice de dimension q^3, nous proposons une méthode qui présente quelques similarités avec l'approche de [Jou13b]. Nous commençons ainsi par scinder l'ensembles des polynômes irréductibles cubiques en plusieurs groupes, comme illustré par la figure 4.1, puis nous cherchons un moyen d'engendrer des relations qui ne font intervenir que des polynômes dans un même groupe, avec, éventuellement, des polynômes linéaires ou quadratiques dont nous connaissons déjà les logarithmes discrets.

Groupes de degré 3 pour la construction simple

Pour définir un groupe de polynômes de degré 3 nous partons d'un élément g dans le corps de base $\mathbb{F}_q$, et nous appelons $\mathcal{P}_g$ le groupe associé, qui

correspond à l'ensemble des polynômes de degré 3 tels que :

$$\mathcal{P}_g = \{(X^3 + g) + \alpha X^2 + \beta X | (\alpha, \beta) \in {\mathbb{F}_q}^2\}.$$

Aussi, lorsque que nous engendrons une relation de la forme (4.5), ou (4.6), en partant de $A(X) = (X^3 + g) + \alpha X^2$ et $B(X) = (X^3 + g) + \beta X$, avec a et b dans $\mathbb{F}_q$, alors tous les polynômes de degré 3 qui interviennent dans le membre de gauche appartiennent bien au groupe $\mathcal{P}_g$. Les éléments de $\mathcal{P}_g$ peuvent en réalité être divisés naturellement en deux sous-groupes :

— les polynômes réductibles dont les logarithmes peuvent être retrouvés via la somme des logarithmes de leurs facteurs (que nous connaissons grâce à l'étape précédente).
— et les polynômes irréductibles qui incarnent nos nouvelles inconnues. Remarquons que les polynômes irréductibles dans un groupe $\mathcal{P}_g$ sont environ au nombre de $q^2/3$.

Pour un élément g préalablement fixé, en balayant toutes les possibilités pour les valeurs de α et β, nous trouvons q^2 relations candidates. Parmi elles, nous ne conservons cependant que celles dont le membre de droite se factorise en produits de termes de degré au plus 2. Ces équations sont satisfaisantes ; mais sommes-nous parvenu à en obtenir suffisamment pour résoudre le système linéaire associé ?

Pour répondre à cette interrogation légitime, nous devons étudier plus attentivement le membre de droite. Avec les choix de h_0 et h_1 relatifs à cette construction, il s'agit d'un polynôme de degré 9, comme décrit dans la proposition 4.1.2. En outre, il découle du lemme 4.2.1 qu'il est toujours multiple du polynôme de degré 3 donné par $\theta h_1(\theta) - h_0(\theta)$. Nous sommes donc en présence d'un polynôme résiduel de degré 6, dont nous aimerions évaluer la probabilité de friabilité en termes de degré au plus 2. En suivant les heuristiques habituelles, celle-ci n'est malheureusement pas plus grande que 1/3. Pour augmenter cette probabilité, nous remarquons tout d'abord que nos choix de polynômes A et B donnent en réalité un membre de droite dont le numérateur est de degré 8 (et non 9 si l'analyse n'est pas assez finement poussée). Aussi nous obtenons un polynôme résiduel de degré 5, dont nous cherchons encore une fois à évaluer la probabilité de friabilité en termes de degré au plus 2. Pour améliorer de nouveau cette probabilité, nous révélons alors un facteur systématique des plus simples.

Lemme 4.2.3 (Facteur systématiques d'un 3-crochet particulier dans le cadre de la construction simple). *Soit h_0, h_1, A et B quatre polynômes tels que h_0 soit affine, $h_1(X) = X(X + t)$, $A(X) = (X^3 + g) + \alpha X^2$ et $B(X) = (X^3 + g) + \beta X$, avec t, g, α et β dans $\mathbb{F}_q$. Alors le crochet $[A, B]_3$ est un polynôme de degré au plus 8 divisible par X.*

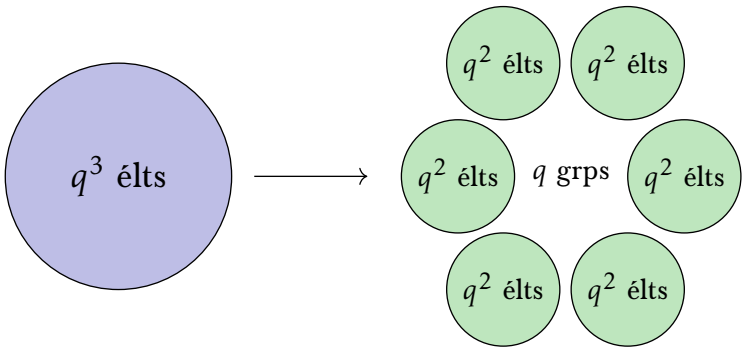


Figure 4.1 – Décomposition des polynômes cubiques unitaires

Démonstration. Par bilinéarité et antisymétrie nous avons :

$$[A,B]_3 = \alpha\left[X^2, X^3+g\right]_3 + \beta\left[X^3+g, X\right]_3 + \alpha\beta\left[X^2, X\right]_3.$$

Calculons alors les 3-crochets :

$$\begin{aligned}
\left[X, X^2\right]_3 &= X^2 h_0 h_1^2 - X h_0^2 h_1 \\
\left[X^3+g, X\right]_3 &= X(h_0^3 + g h_1^3) - (X^3+g) h_0 h_1^2 \\
&= X[h_0^3 + g h_1^3 - (X^3+g) h_0 X(X+t)^2] \\
\left[X^3+g, X^2\right]_3 &= X^2(h_0^3 + g h_1^3) - (X^3+g) h_0^2 h_1 \\
&= X[X(h_0^3 + g h_1^3) - (X^3+g) h_0^2 (X+t)]
\end{aligned}$$

Le résultat du lemme provient du fait que tous les 3-crochets impliqués dans le calcul de $[A,B]_3$ sont divisibles par X. De plus, une étude attentionnée des degrés des polynômes de ces éléments nous indique que $\left[X, X^2\right]_3$ est de degré 6 tandis que $\left[X^3+g, X\right]_3$ est de degré 7 et $\left[X^3+g, X^2\right]_3$ est enfin de degré 8. □

Il découle de ce qui précède un facteur résiduel de degré 4. Appliquer le lemme 3.1.2 pour $D = 4$ et $d = 2$ nous permet maintenant d'écrire que la probabilité heuristique que celui-ci se factorise en termes de degré au plus 2 est proche de 41%. Puisque celle-ci est maintenant plus haute que 1/3, nous espérons trouver suffisamment d'équations pour calculer les logarithmes discrets de tous les polynômes irréductibles appartenant à $\mathcal{P}_g$.

Par ailleurs, il est clair que n'importe lequel des polynômes unitaires irréductibles de degré 3 appartient bien à l'un des groupes $\mathcal{P}_g$.

Groupes de degré 3 pour la variante utile

Ce contexte demande un découpage des polynômes de degré 3 plus complexe. Pour définir un groupe dans cette configuration, nous partons cette fois-ci d'un triplet (g_1, g_2, g_3) d'éléments de $\mathbb{F}_q$. Le groupe associé est alors défini comme l'ensemble des polynômes de degré 3 tels que :

$$\mathcal{P}_{g_1,g_2,g_3} = \{X^2(X-g_1) + \alpha X(X-g_2) + \beta(X-g_3) | (\alpha,\beta) \in {\mathbb{F}_q}^2\}.$$

Soit $(g_1, g_2, g_3) \in {\mathbb{F}_q}^3$. Si nous engendrons une relation de la forme (4.5) avec les polynômes $A(X) = X^2(X-g_1) + \alpha(X-g_3)$ et $B(X) = X(X-g_2) + \beta(X-g_3)$, avec α et β dans $\mathbb{F}_q$, alors tous les polynômes de degré 3 qui apparaissent dans le membre de gauche appartiennent au groupe associé $\mathcal{P}_{g_1,g_2,g_3}$. Nous ne conservons maintenant que les q^2 relations candidates dont le membre droit se factorise en termes de degrés au plus 2. La question de nouveau consiste à déterminer si nous obtenons suffisamment d'équations satisfaisantes pour résoudre le système d'algèbre linéaire ou non. Nous rappelons que les inconnues sont les logarithmes des $q^2/3$ polynômes irréductibles de $\mathcal{P}_{g_1,g_2,g_3}$.

Lorsque h_0 est quadratique et h_1 affine, le membre de droite est encore une fois un polynôme de degré 8 divisible par le polynôme $\theta h_1(\theta) - h_0(\theta)$. Il s'agit donc de savoir si le polynôme résiduel, qui est de degré 6, se factorise en termes de degrés au plus 2. Sans aucune amélioration ou idée nouvelle, la probabilité que celui-ci se factorise en polynômes suffisamment petit reste cependant bien trop basse pour obtenir un nombre satisfaisant d'équations.

Pour surmonter cet obstacle, nous ne considérons plus des groupes généraux de cette forme. Le but consiste maintenant à exhiber des groupes particuliers pour lesquels le membre de droite démontrerait des facteurs systématiques. Un second argument qui explique notre intérêt pour des groupes spéciaux, quand bien même ils seraient moins nombreux, provient du fait que le nombre de polynômes de degré 3 produits avec tous les groupes généraux est bien trop grand. En effet, considérer les q^3 groupes de la forme $\mathcal{P}_{g_1,g_2,g_3}$ est un excès clair, puisque chacun d'entre eux contient déjà q^2 éléments tandis qu'il n'existe que q^3 polynôme unitaires de degré 3. En fait, nous pouvons même espérer que l'intégralité de ces polynômes Soit recouverts par q groupes seulement. Nous nous restreignons alors aux q choix des éléments g_1, g_2 et g_3 tels que g_1 soit une valeur arbitraire de $\mathbb{F}_q$ puis nous calculons :

$$g_2 = G(g_1) \quad \text{et} \quad g_3 = G(g_2)$$

où $G : \mathbb{F}_q \mapsto \mathbb{F}_q$ est une application à définir. Nous proposons par exemple de poser :

$$G : g \mapsto \frac{v(v+w)}{(1+u)(v+w-g)}. \tag{4.7}$$

Nous rappelons que u, v et w désignent les coefficients des polynômes h_0 et h_1, comme donnés dans l'équation (4.4). En supposant que chacun des deux éléments g_1 et g_2 ainsi construits n'est pas égaux à $v+w$, alors les trois valeurs du triplet (g_1, g_2, g_3) sont bien définies. Ce choix particulier d'éléments nous permet de constater maintenant que le membre de droite dans une équation de la forme (4.5) ou (4.6) démontre un nouveau facteur systématique de degré 2 qui n'est autre que :

$$\theta h_1 + h_0 + (v+w)h_1 = (1+u)\theta^2 + (1+u)(v+w)\theta + vw + v^2.$$

Ce résultat provient du lemme 4.2.4. De nouveau, le polynôme résiduel du membre de droite est de degré 4, dans ce cas précis. Puisque la probabilité qu'un polynôme aléatoire de degré $D = 4$ se factorise en termes de degré au plus $d = 2$ est d'environ 41%, donc plus grande que 1/3, nous pouvons retrouver les logarithmes discrets des polynômes irréductibles du groupe particulier $\mathcal{P}_{g_1, G(g_1), G(G(g_1))}$ pour tout élément g_1 du corps de base.

Lemme 4.2.4 (Facteur systématique d'un 3-crochet particulier dans le contexte de la variante utile). *Soit G l'application désignée par* (4.7) *et notons h_0, h_1, A et B quatre polynômes tels que :*

- $h_0(X) = X(X+w)$,
- $h_1(X) = uX + v$,
- $A(X) = X^2(X-g) + a(X - G(G(g)))$
- *et* $B(X) = X(X - G(g)) + b(X - G(G(g)))$,

avec u, v, w, a, b et g des éléments arbitraires de $\mathbb{F}_q$.

Alors $[A, B]_3$ est un multiple du polynôme $(1+u)X^2 + (1+u)(v+w)X + vw + v^2$.

Démonstration. Par bilinéarité et antisymétrie nous savons comme précédemment que $[A, B]_3$ est égal à la somme des polynômes donnée par :

$$\begin{aligned}&\left[X^2(X-g), X(X-G(g))\right]_3\\ &+b\left[X^2(X-g), X-G(G(g))\right]_3\\ &+a\left[X-G(G(g)), X(X-G(g))\right]_3.\end{aligned}$$

Le résultat du lemme provient des calculs des 3-crochets effectués sur les trois paires possibles constituées de deux des éléments parmi $X^2(X-g)$, $X(X-G(g))$ et $X - G(G(g))$. Ce calcul pédestre mais sans difficulté est épargné au lecteur. □

Fraction des polynômes de degré 3 recouverts par nos groupes

Nous pouvons donc retrouver tous les logarithmes discrets des éléments qui apparaissent dans chacun de nos groupes particuliers. La question consiste

maintenant à déterminer si chacun des polynômes dont nous cherchons le logarithme se manifeste bien dans l'un de ces groupes au moins.

Groupes valides. Dans la suite nous nous restreignons au cas où $v+w \neq 0$. En effet, si $v+w=0$ alors G correspond à l'application nulle, dont l'étude trop particulière ne sera pas menée dans ce manuscrit. Puisque G est une homographie, nous pouvons la transformer en une permutation de la droite projective $\mathbb{P}_1(\mathbb{F}_q)$. Suivant les conventions classiques nous adjoignons donc aux images de G les deux valeurs suivantes :

$$\begin{aligned} G(\infty) &= 0 \\ \text{et } G(v+w) &= \infty. \end{aligned}$$

Cette double définition nous permet de constater que les groupes qui forment l'objet de notre attention sont indicés par les triplets $(g, G(g), G(G(g)))$ qui ne contiennent pas la valeur ∞. Par ailleurs, puisque $v+w \neq 0$ alors ∞ appartient à un cycle de longueur au moins égale à 3. Il y a donc $q-2$ groupes valides associés aux valeurs de g dans $\mathbb{F}_q - \{G^{-1}(\infty), G^{-1}(G^{-1}(\infty))\}$. Cette description nous permet alors d'atteindre au mieux q^3-2q^2 polynômes de degré 3.

Groupes à l'infini. Afin d'accéder à davantage de polynômes nous définissons trois groupes additionnels un peu particulier, à savoir $\mathcal{P}_{g,G(g),G(G(g))}$ lorsque $g, G(g)$ ou $G(G(g))$ est égal à ∞. Ces trois groupes sont décrits de la façon suivante :

$$\begin{aligned} \mathcal{P}_{\infty,0,G(0)} &= \left\{ X\left(X^2 + \frac{vw+v^2}{1+u}\right) + \alpha X^2 + \beta (X - G(0)) | (\alpha,\beta) \in \mathbb{F}_q^2 \right\}. \\ \mathcal{P}_{\infty^{-1},\infty,0} &= \left\{ X^2(X-\infty^{-1}) + \alpha X + \beta \left(X^2 + \frac{vw+v^2}{1+u}\right) | (\alpha,\beta) \in \mathbb{F}_q^2 \right\}. \\ \text{et } \mathcal{P}_{\infty^{-2},\infty^{-1},\infty} &= \{X^2(X-\infty^{-2}) + \alpha X(X-\infty^{-1}) + \beta | (\alpha,\beta) \in \mathbb{F}_q^2\}. \end{aligned}$$

où ∞^{-1} désigne $G^{-1}(\infty)$ et ∞^{-2} correspond à $G^{-1}(G^{-1}(\infty))$. Une première remarque consiste à relever que chacun de ces trois groupes supplémentaires satisfait aussi la propriété de divisibilité des groupes précédemment exposée. De plus, nous venons d'élargir le nombre de polynômes accessibles de q^3-2q^2 à q^3+q^2, ce qui est donc suffisant dorénavant pour prétendre essayer de recouvrir tous les polynômes unitaires de degré 3.

Recouvrement de l'intégralité des polynômes de degré 3. Soit $P(X) = X^3 + a_2X^2 + a_1X + a_0$ un polynôme arbitraire unitaire de degré 3. Si P appartient à l'un des groupes valides $\mathcal{P}_{g,G(g),G(G(g))}$, alors il existe α et β tel

que :

$$\begin{aligned} \alpha - g &= a_2, \\ \beta - \alpha\, G(g) &= a_1, \\ \text{et } -\beta\, G(G(g)) &= a_0. \end{aligned}$$

En substituant les équations les unes dans les autres, ceci implique alors :

$$a_0 = -(a_1 + (a_2 + g)\, G(g)) \cdot G(G(g)). \tag{4.8}$$

Nous pouvons simplifier cette expression et la réécrire $H_{a_1,a_2}(g) = a_0$, où H_{a_1,a_2} est une homographie dont les coefficients dépendent de a_1 et a_2. Si ceux-ci ne dégénèrent pas, il n'y a qu'une seule possibilité pour la valeur de g. De manière plus explicite, notons cette homographie $H_{a_1,a_2}(g) = \frac{\lambda+\mu g}{\lambda'+\mu' g}$ avec $\lambda = -v(w+v)((1+u)a_1 + va_2)$, $\mu = v((1+u)a_1 - v(v+w))$, $\lambda' = (1+u)(u(v+w)+w)$ et $\mu' = -(1+u)^2$. Différents cas de figures se manifestent alors :

- Si $a_0 \neq \mu/\mu'$, alors cette homographie est inversible.
  - Par conséquent, dès lors que $g \neq \infty^{-1}$ et $g \neq \infty^{-2}$, le polynôme P appartient au groupe valide engendré par $g = H^{-1}_{a_1,a_2}(a_0), G(g)$ et $G(G(g))$, et uniquement à celui-ci. Notons qu'il existe précisément $q^3 - 3q^2$ polynômes de la sorte.
  - Si $g = \infty^{-1}$ alors $H_{a_1,a_2}(\infty^{-1}) = a_0$ devient $a_0(\lambda' + \mu'(v+w)) = \lambda + \mu(v+w)$ et finalement $a_0 = \infty^{-1} v(a_2 + \infty^{-1})/(1+u)$. De plus, P appartient au groupe à l'infini $\mathcal{P}_{\infty^{-1},\infty,0}$ sous réserve de l'existence de α et β tels que $\beta - \infty^{-1} = a_2$, $\alpha = a_1$, et $\beta v(v+w)/(1+u) = a_0$. En substituant les équations en β précédentes les unes dans les autres, nous en déduisons que $a_0 = \infty^{-1} v(a_2 + \infty^{-1})/(1+u)$. Ceci conclut l'appartenance du polynôme P au groupe $\mathcal{P}_{\infty^{-1},\infty,0}$. Cette fois-ci nous trouvons $q^2 - q$ tels polynômes.
  - De manière similaire, si $g = \infty^{-2}$ alors P appartient au groupe à l'infini $\mathcal{P}_{\infty^{-2},\infty^{-1},\infty}$ et, de nouveau, il existe $q^2 - q$ polynômes de la même nature.
- En revanche si $a_0 = \mu/\mu'$ alors l'équation (4.8) est équivalente à $0 = g(a_0\mu' - \mu) = \lambda - \lambda'$. De plus, exiger $\lambda = \lambda'$ mène à $a_2 = \kappa(\kappa' a_1 + \kappa'')$ où $\kappa = (1+u)/(v^2(v+w))$, $\kappa' = v(v+w)$ et $\kappa'' = -u(v+w) - w$.
  - Si $a_2 = \kappa(\kappa' a_1 + \kappa'')$ alors P appartient à tous les groupes valides. Il y a q tels polynômes.
  - Si $a_2 \neq \kappa(\kappa' a_1 + \kappa'')$ la question est de savoir si les $q^2 - q$ polynômes restant appartiennent bien à l'un des groupes à l'infini. Heureusement, si α désigne a_2 et β correspond à $a_1 - v(v+w)/(1+u)$ alors

nous avons l'égalité polynomiale qui suit : $X(X^2 + v(w+v)/(1+u)) + \alpha X^2 + \beta(X - G(0)) = X^3 + a_2X^2 + a_1X + v(v(v+w) - a_1(1+u))/(1+u)^2 = P(X)$. Ainsi, le polynôme P appartient au groupe à l'infini $\mathcal{P}_{\infty,0,G(0)}$.

Remarque 4.2.5. La preuve précédente n'est pas affectée par la contrainte que nous avons sur le coefficient a_2. Aussi, les q polynômes qui sont tels que $a_0 = \mu/\mu'$ et $a_2 = \kappa(\kappa' a_1 + \kappa'')$ font aussi partie du groupe à l'infini $\mathcal{P}_{\infty,0,G(0)}$. Parallèlement, nous signalons que chacune des intersections entre deux groupes à l'infini consiste exactement en q polynômes.

4.2.3 Logarithmes discrets des polynômes de degré 4

Les obstructions antérieures

L'approche classique lorsqu'il s'agit de calculer le logarithme de $I_4(\theta)$, où I_4 est un polynôme irréductible de degré 4, commence par construire deux polynômes cubiques $A(X) = X^3 + a_1X + a_0$ et $B(X) = X^2 + b_1X + b_0$. Elle forme avec ceux-ci une relation depuis l'équation (4.5) et teste si I_4 divise $[A,B]_3$, jusqu'à trouver une bonne paire de polynômes A et B. Pour se faire, nous réécrivons la condition de divisibilité comme :

$$[A,B]_3 = 0 \mod I_4.$$

Nous obtenons ainsi 4 équations bilinéaires en les 4 inconnues (a_0, a_1, b_0, b_1) qui ne sont autre que les coefficients de A et B. De manière expérimentale et comme il est expliqué dans [Joul3b], ce système se résout facilement grâce à l'utilisation d'algorithmes classiques de bases de Gröbner. Cependant, le système n'admet de solution qu'en moyenne pour la moitié des polynômes de degré 4 seulement. Ceux-ci sont donc accessibles mais l'autre moitié reste hors d'atteinte, et l'on ne peut connaitre les logarithmes de ces éléments sans utiliser d'autres techniques.

Une seconde idée déjà présente dans [AMOR14] s'articule autour des relations additionnelles du paragraphe 4.2.1, afin d'améliorer les probabilités de succès. Prenons un polynôme irréductible de degré 4 qui échoue à être exprimé uniquement avec des polynômes de degrés inférieurs. Alors il y a une chance sur deux que son image par Frobenius, qui est alors de degré 8, se factorise en deux polynômes quartiques. Chacun d'entre eux à de nouveau une chance sur deux de pouvoir être exprimé comme produit de termes de degré 3 au plus. Aussi, pour un polynôme qui échoue, nous avons une probabilité de 1/8 de calculer son logarithme grâce à ce procédé. Ceci augmente déjà la probabilité de trouver le logarithme d'un polynôme de degré 4 à 9/16. En

répétant cette méthode, la probabilité augmente progressivement. Aussi, nous espérons heuristiquement atteindre une probabilité $p_0 = (4 - \sqrt{8})/2 \approx 0.586$. Ce gain de probabilité ne suffit malheureusement pas à atteindre tous les polynômes de degré 4. Pour contourner ce problème, de nombreuses techniques ont été proposées mais aucune d'entre elles ne suffit à régler le souci du cas général. Nous détaillons dans la suite une approche qui s'adapte au cas de la construction réellement simple, l'analogue pour la variante utile étant similaire.

Nouvelle approche des polynômes de degré 4 pour la construction simple

L'idée que nous souhaitons mettre en place consiste à diviser les polynômes quartiques en groupes de taille q^3 chacun puis à suivre une approche similaire à celle du traitement des polynômes cubiques présentée au paragraphe 4.2.2. Nous calculons d'abord tous les logarithmes discrets d'un groupe $\mathcal{Q}_g$ de polynômes de degré 4 de la forme :

$$\mathcal{Q}_g = \{(X^4 + g) + \alpha X^3 + \beta X^2 + \gamma X | (\alpha, \beta, \gamma) \in \mathbb{F}_q^3\}. \tag{4.9}$$

D'une certaine manière nous allons étendre la méthode classique de [Jou13b] qui s'appuie sur les polynômes cubiques pour calculer les polynômes quartiques de $\mathcal{Q}_g$. Plus précisément, nous considérons une partition du groupe $\mathcal{Q}_g = \cup_{g' \in \mathbb{F}_q} \mathcal{Q}_{g,g'}$ avec :

$$\mathcal{Q}_{g,g'} = \{(X^4 + g) + \alpha X^3 + \beta X^2 + g' X | (\alpha, \beta) \in \mathbb{F}_q^2\}. \tag{4.10}$$

Cette idée est schématisée dans la figure 4.2. Pour construire des relations ne faisant intervenir que des polynômes de $\mathcal{Q}_{g,g'}$ nous réécrivons l'équation (4.5) avec les polynômes de la forme $A(X) = (X^4 + g) + aX^2 + g'X$ et $B(X) = X^3 + bX^2$. Dans le contexte de la construction simple, le lemme 4.2.6 indique que le polynôme $[A, B]_4$ est de degré 11 et qu'il possède un facteur systématique de degré 1. En combinant ce résultat avec le facteur systématique général de degré 3 que nous connaissons grâce au lemme 4.2.1, notre polynôme résiduel est alors de degré $D = 7$. Le lemme 3.1.2 nous conduit à évaluer la probabilité qu'il se factorise en termes de degré au plus $d = 3$ comme étant approximativement égale à 24%.

Par ailleurs, le nombre de polynômes irréductibles du groupe $\mathcal{Q}_{g,g'}$ est proche de $q^2/4$. Combien d'inconnues allons-nous traiter ? A l'aide des techniques précédentes que nous pouvons combiner, c'est-à-dire après avoir écarté les polynômes réductibles puis les polynômes irréductibles dont nous savions déjà calculer le logarithme, nous sommes en présence de $(1 - 0.586) \cdot q^2/4 \approx$

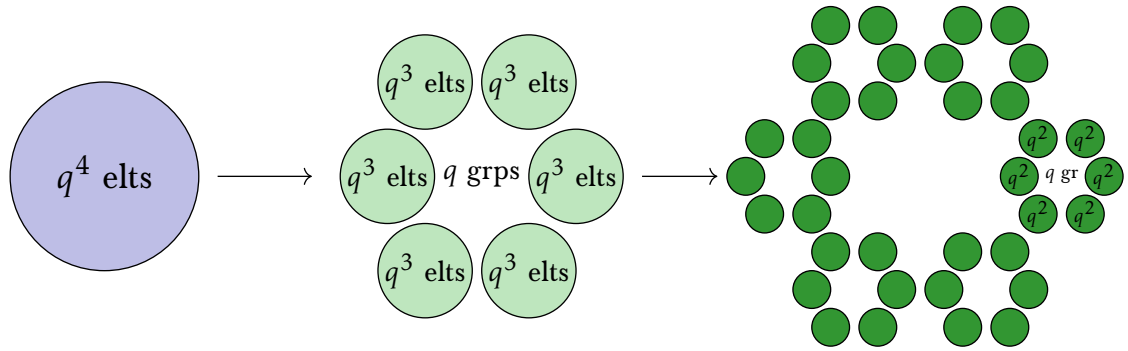


Figure 4.2 – Décomposition des polynômes quartiques unitaires

$0.10\,q^2$ inconnues environ. Aussi nous constatons que nous obtenons heuristiquement un nombre suffisant d'équations linéaires pour résoudre le système. Nous retrouvons enfin les logarithmes discrets des éléments de $\mathcal{Q}_g$ en passant par le calcul de ceux de ses q-sous-groupes l'un après l'autre.

Lemme 4.2.6 (Facteur systématique d'un 4-crochet particulier dans le cadre de la construction simple). *Soit h_0, h_1, A et B quatre polynômes de $\mathbb{F}_q[X]$ tels que h_0 soit affine, $h_1(X) = X(X+t)$, $A(X) = (X^4+g) + \alpha X^2 + \alpha' X$ et $B(X) = X^3 + \beta X^2 + \beta' X$. Alors $[A,B]_4$ est un polynôme de degré au plus 11 qui est un multiple de X.*

La preuve de ce lemme est similaire à celle donnée pour le lemme 4.2.3, aussi nous ne la détaillons pas ici.

Retrouver les logarithmes discrets qui sont encore secrets. Soit $I_4 \notin \mathcal{Q}_g$ un polynôme de degré 4. Nous partons de nouveau de la paire de polynômes $A(X) = (X^4+g) + aX^2 + a'X$ et $B(X) = X^3 + bX^2 + b'X$, à laquelle nous appliquons l'équation (4.5) pour construire une relation, dans l'espoir que I_4 divise le crochet $[A,B]_4$. Comme dans [Jou13b], et en l'absence d'autres manipulations, la probabilité heuristique de trouver une solution au système bilinéaire qui en découle est 1/2. Encore une fois, comptabilisons les facteurs systématiques que nous pouvons extraire de ce crochet. En additionnant les contributions du facteur de degré 1 donné par le lemme 4.2.6, celui général de degré 3 qui apparait dans le lemme 4.2.1, et enfin en divisant $[A,B]_4$ qui est de degré 11 par notre polynôme cible I_4 de degré 4, nous obtenons directement

un polynôme de degré 3, dont le logarithme discret est donc connu. Nous pouvons ainsi écrire une relation de la forme :

$$I_4(\theta)P_3(\theta)S_3(\theta)S_1(\theta) = [A,B]_4(\theta),$$

où S_3 et S_1 sont les facteurs systématiques, et P_3 un polynôme de logarithme connu. Puisque $[A,B]_4$ peut se réécrire lui-même comme produit de h_1 et d'éléments de $\mathcal{Q}_g$, avec un seul groupe de la forme décrite dans (4.9) nous retrouvons le logarithme discret d'environ la moitié des polynômes irréductibles de degré 4 manquants.

Remarque 4.2.7. La probabilité de retrouver le logarithme encore inconnu d'un polynôme est en réalité plus grande que 1/2. En effet, nous pouvons utiliser les équations additionnelles présentées au paragraphe 4.2.1. Bien qu'elle soit très utile en pratique, cette approximation à 1/2 suffit pour notre analyse, aussi nous ne cherchons pas de minoration plus fine.

Pour combler progressivement l'ignorance que nous avons des logarithmes résiduels, nous appliquons récursivement cette même méthode aux autres groupes de la forme (4.9). La figure 4.3 récapitule l'idée directrice que nous venons de détailler. Nous montrons au paragraphe 4.3.3 qui suit que $O(\log(q))$ groupes suffisent en réalité pour retrouver les logarithmes, et que le coût de calcul des logarithmes des éléments associés est asymptotiquement dominé par le coût de calcul du premier groupe, qui est en $O(q^6)$, comme annoncé.

4.3 Complexités asymptotiques

4.3.1 Logarithmes discrets des polynômes quadratiques irréductibles

Nous souhaitons rassembler environ q^2 équations à l'issue de la récolte des relations. Puisque la probabilité d'obtenir une relation satisfaisante est minorée par 2/3, cette phase coûte de l'ordre de $O(q^2)$ opérations. Nous exécutons ensuite une étape d'algèbre linéaire sur une matrice de taille $O(q^2)$, dont le nombre d'entrées non nulles pour chaque ligne est en $O(q)$, grâce à la forme des relations créées. Le coût total pour retrouver le logarithme discret des polynômes de degré inférieur à 2 est donc $O((q^2)^2 \cdot q) = O(q^5)$.

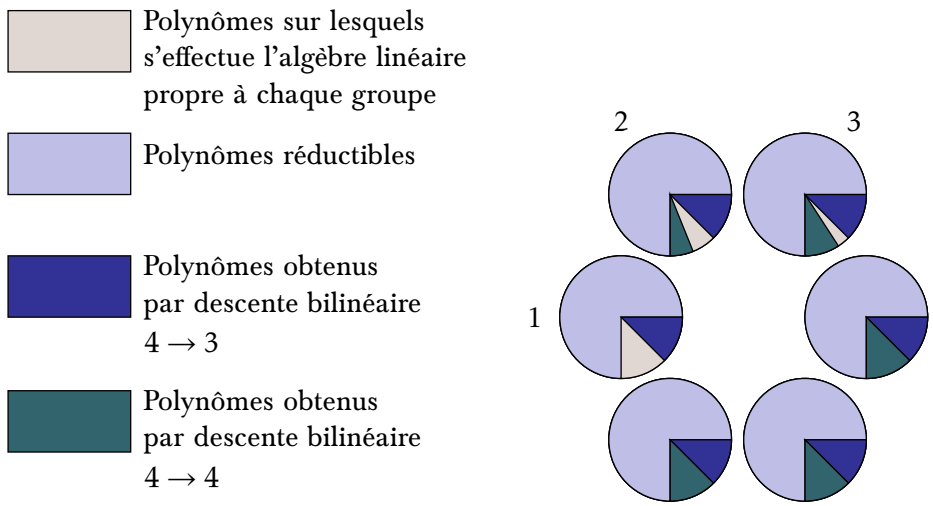


Figure 4.3 – *Représentation des différentes méthodes utilisées pour calculer les logarithmes des polynômes quartiques.* Chaque disque correspond à l'un des moyens groupes de polynômes à traiter ; nous remarquons que les 3/4 des polynômes à l'intérieur de chacun sont déjà réductibles, donc de logarithmes connus. La moitié des polynômes restant descendent du degré 4 vers le degré 3. Parmi l'autre moitié (donc un huitième des polynômes d'un groupe), la moitié de nouveau d'entre eux peuvent être descendus vers les polynômes quartiques déjà calculés. Cette idée n'affecte pas le traitement du premier groupe, celui situé à 9 heures sur la figure, mais intervient dès le second, à 11 heures, et ainsi de suite, divisant par 2 à chaque étape la dimension de la matrice sur laquelle s'effectue chaque algèbre linéaire.

4.3.2 Logarithmes discrets des polynômes cubiques irréductibles

Pour la construction simple

Puisque chacun des groupes $\mathcal{P}_g$ contient $O(q^2)$ inconnues, et comme l'algèbre linéaire effectuée pour chacun d'eux manipule une matrice contenant encore une fois $O(q)$ entrées non nulles par ligne, le coût de calcul des logarithmes discrets de tous les éléments d'un seul groupe s'élève à $O(q^5)$. Nous traitons q groupes de cette forme, donc le coût total est simplement en $O(q^6)$.

Pour la variante utile

L'analyse est identique, puisque nous considérons ici 3 groupes à l'infini au côté de $q-2$ groupes valides, à $O(q^2)$ inconnues chacun. Le coût total asymptotique de cette étape s'élève donc encore une fois à $O(q^6)$.

4.3.3 Logarithmes discrets des polynômes quartiques irréductibles

Pour la construction simple. Nous calculons en premier lieu les logarithmes discrets d'un seul groupe de la forme (4.10). Puisque nous obtenons un système de dimension $O(q^2)$ avec $O(q)$ coefficients non nuls par ligne, nous pouvons le résoudre pour un coût de $O(q^5)$. Pour retrouver l'intégralité des logarithmes d'un seul groupe de la forme (4.9), nous effectuons donc déjà $O(q^6)$ opérations.

Parallèlement, la probabilité de retrouver le logarithme d'un polynôme irréductible de degré 4 depuis le premier groupe de la forme (4.9) déjà traité, est heuristiquement de $1/2$. En considérant que les probabilités de succès sont indépendantes, et qu'il y a k tels groupes, la proportion des logarithmes discrets qui restent inconnus après cette étape est $1/2^k$. Lorsque le nombre de groupes disponibles augmente, la proportion des logarithmes qui restent non déterminés tend ainsi rapidement vers 0. Avec $O(\log(q))$ groupes de cette forme nous espérons obtenir le logarithme de tous les polynômes quartiques. Aussi, en effectuant le calcul sur $O(\log(q))$ tels groupes avec cette méthode directe, nous obtiendrions une complexité de l'ordre de $O(q^6 \log q)$. Cependant, cette analyse naïve ne tient pas compte du fait qu'à chaque nouveau groupe considéré, la taille du système linéaire correspondant décroit. Le coefficient de réduction suit d'ailleurs une loi géométrique. Par conséquent, le coût de calcul pour les $O(\log(q))$ groupes nécessaires est dominé par le calcul du premier de ceux-ci.

Une autre possibilité consiste à poursuivre le calcul pour tous les groupes. Grâce à cette progression géométrique, la complexité de cette phase sera iden-

tique. Cependant, il faut souligner qu'une telle méthode, si elle mène à un temps de calcul plus bas que l'option de trouver à la volée les logarithmes des polynômes de degré 4 manquants au moment où ils sont utiles, requiert en guise d'effet secondaire désagréable l'utilisation d'une quantité de mémoire significativement plus grande.

Par conséquent, la complexité totale du précalcul des phases de collecte de relations, d'algèbre linéaire sur la base de friabilité initiale, et d'extension de cette base est maintenant en :

$$O(q^6).$$

Celle-ci est à comparer avec l'ancienne complexité en $O(q^7)$ donnée antérieurement pour le même précalcul. Néanmoins, nous soulignons que l'étape qui domine encore asymptotiquement les algorithmes par représentation de Frobenius est précisément la dernière, c'est-à-dire celle de descente que nous ne considérons pas dans cet article.

Remarque 4.3.1. Un petit mot sur la complexité asymptotique totale. Nous n'étudions ici que les phases d'algèbre linéaire depuis le point de vue des algorithmes des familles de Lanczos ou Wiedemann, qui ont une complexité en $O(n^2)$ pour toute matrice de dimension n avec un nombre constant d'entrées non nulles. Il s'agit en effet de poursuivre une analyse au plus près de la réalité. Pourtant, l'utilisation de méthodes d'algèbre linéaire dense à base de multiplications matricielle rapide au lieu de ces méthodes creuses entraineraient une chute de complexité de $O(q^6)$ à :

$$O(q^{5.746}).$$

Néanmoins, cette complexité asymptotique n'a que peu de sens, puisqu'en pratique ces algorithmes ne sont absolument pas compétitifs.

4.4 Application en caractéristique 3

4.4.1 Représentation du corps cible

En guise d'illustration nous avons implanté notre nouvel algorithme sur un exemple de taille compétitive en caractéristique 3. Plus précisément nous nous sommes proposés de résoudre le problème du logarithme discret sur le corps cible :

$$\mathbb{F}_{3^{5\cdot 479}}.$$

Nous posons en premier lieu $q = 3^5$ puis définissons notre corps de base de cardinalité q comme $\mathbb{F}_q = \mathbb{F}_3[\alpha]$, où α satisfait $\alpha^5 - \alpha + 1 = 0$. En choisissant les deux polynômes $h_0 = X^2 + \alpha^{111} X$ et $h_1 = \alpha X + 1$ nous remarquons que le polynôme de degré $2q = 486$ donné par $X h_1(X^q) - h_0(X^q)$ possède un facteur irréductible de degré 479, qui est un entier premier. Nous choisissons maintenant U une racine dans la clôture algébrique de $\mathbb{F}_q$ de ce polynôme irréductible et considérons le corps cible $\mathbb{F}_{3^{5\cdot 479}}$ comme étant défini égal à $\mathbb{F}_q[U]$.

Comparaison des tailles des ordres

De ce fait la cardinalité du corps que nous cherchons à attaquer est un entier de 3796 bits. Un point de comparaison pertinent nous est donné par le calcul de logarithmes discrets sur le corps $\mathbb{F}_{2^{12\cdot 367}}$ effectué dans [GKZ14a]. En effet, bien que la taille en bits de ce dernier calcul soit légèrement supérieure à la nôtre, puisqu'elle atteint 4404 bits, cette extension présente un facteur 2 dans le degré de l'extension qui apparait gratuitement lorsque nous considérons des algorithmes par représentation de Frobenius. Cette remarque illustre bien l'inconvénient de notre méthode qui calcule des logarithmes non pas dans le corps $\mathbb{F}_{q^{\delta k}}$ mais uniquement dans le sous-corps $\mathbb{F}_{q^k}$. Les critiques à propos de ce facteur gratuit sont nombreuses, et les discussions vont notamment bon train concernant sa pertinence pratique. De ces discussions semble émerger un intérêt croissant pour les extensions de corps de degré première, plus représentatives de la difficulté réelle du problème sous-jacent que leurs homologues composées.

Plus gros sous-corps d'extension première

Il semble ainsi plus juste de comparer non pas les corps cible eux-mêmes, mais les plus grosses extensions premières qui peuvent être plongées dans chacun d'eux. Nous concernant, il s'agit du sous-corps :

$$\mathbb{F}_{3^{479}}$$

qui expose quant à lui une cardinalité de 760 bits. En ce sens nous pouvons aussi comparer notre calcul avec le plus gros corps de la même forme jamais attaqué, c'est-à-dire le corps binaire $\mathbb{F}_{2^{809}}$ Nous renvoyons à [BBD+14] pour les détails de ce calcul.

Corps non binaire

Cependant, la comparaison la plus naturelle et la raison pour laquelle notre application s'avérait être un record provenait du fait qu'il s'agissait du plus gros corps de caractéristique 3 démonté jusqu'à lors. Le précédent record de calcul [AMOR14] en caractéristique identique concernait le corps cible $\mathbb{F}_{3^{6*163}}$ qui présente une cardinalité de 1551 bits, à comparer avec les 3796 bits du corps de référence de cette section. Preuve de l'activité importante du domaine, un nouveau record en caractéristique 3 vient d'être annoncé [ACMCC+16] à l'heure de la première impression de ce manuscrit. Il concerne le corps :

$$\mathbb{F}_{3^{6\cdot 509}},$$

de cardinalité un entier de 4841 bits dont le plus gros sous-corps d'extension première est de cardinalité 806 bits. Ce nouveau calcul est lui aussi issu d'une application de l'algorithme que nous venons de présenter dans ce chapitre.

4.4.2 Base de friabilité initiale

Dans ce contexte, retrouver le logarithme discret de chacun des éléments de la base de friabilité qui contient 29 646 polynômes irréductibles linéaires ou quadratiques nous a demandé 16 heures séquentielles sur le coeur unique d'un Intel Core *i*7 à 2.7 GHz. Les relations en tant que telles n'ont pris que 35 secondes pour être engendrées. Aussi, la presque totalité du coût de cette étape a été consommée par l'algèbre linéaire exécutée modulo :

$$M = \frac{3^{5\cdot 479} - 1}{488246858}.$$

4.4.3 Base de friabilité étendue

Polynômes cubiques

L'extension de la base de friabilité aux polynômes de degré 3 a nécessité 244 calculs indépendants (correspondant au 244 groupes étudiés), chacun d'eux comportant 19 602 inconnues au sein du système linéaire associé. Le coût séquentiel d'une seule de ces résolutions demande 6.5 heures sur la même machine que précédemment. Puisque tous ces calculs sont indépendants, il n'y a aucune difficulté à les paralléliser. Le temps total aurait été de 1600 heures CPU sur un unique coeur Intel Core *i*7 à 2.7 GHz. Nous avons néanmoins réduit le temps d'horloge par une exécution sur un cluster hétérogène situé au Laboratoire d'Informatique de Paris 6, UPMC, Sorbonnes-Universités, et à CryptoExperts, Paris.

Polynômes quartiques

L'extension au degré 4 commence quant à elle par 243 calculs indépendants (il s'agit des 243 petits groupes du premier moyen groupe), chacun contenant en moyenne 7 385 inconnues au sein du système associé. La plus grande matrice contient ainsi 7 571 inconnues tandis que la plus petite en a 7 212. Pour étendre la base de friabilité à ce dernier degré, nous avons utilisé une légère variation sous-optimale comparée à celle du paragraphe 4.2.3, qui induit notamment un système de dimension légèrement supérieure. L'utilisation de la variante que nous venons de présenter – mais dont certaines parties sont postérieures au calcul pratique – auraient mené à un plus petit nombre d'inconnues par système, de l'ordre de 6100.

Le deuxième moyen groupe à traiter possède en moyenne 3 674 inconnues, le troisième 1 829, le quatrième 909, le cinquième 452. Comme prévu, nous remarquons la décroissance rapide de la taille des matrices successives à traiter, le coefficient de diminution suivant quasiment une loi géométrique de raison 1/2. Par conséquent, le temps de calcul attribué à chacun de ses moyens groupes devient rapidement négligeable comparé à l'essentiel du calcul qui consiste à retrouver les logarithmes des polynômes cubiques. Notre implémentation se montre de nouveau sous-optimale pour cette partie, mais le temps relativement court nécessaire pour ces calculs ne rend pas cet sous-optimalité critique dans notre contexte. En réalité, pour tous les moyens groupes à partir du cinquième considéré, nous avons simplement essayé d'exprimer les logarithmes des éléments en fonction des quatre premiers moyens groupes déjà calculés. En effet, le système résultant de cette démarche s'est montré si petit – environ 450 inconnues – tout en étant agréablement creux que nous avons pu appliquer un algorithme d'algèbre linéaire direct d'élimination Gaussienne pour le résoudre. Aussi, pour ces groupes de taille moyenne à partir du cinquième, le temps de calcul est dominé par la création des relations, environ 2 heures pour chaque groupe, et nous n'avons pas cherché à réduire d'avantage la taille des systèmes linéaires associés. Au final, le traitement de 30 groupes de taille moyenne a suffi pour exprimer les logarithmes de tous les polynômes quartiques rencontrés dans la suite du calcul. Le temps de calcul de cette phase d'extension a été légèrement inférieure à 500 heures CPU.

4.4.4 L'étape de descente de nouveau dominante

Concernant l'étape finale de descente, nous avons suivi l'état de l'art actuel afin d'exprimer le logarithme discret cherché via 41 millions de polynômes de degré 4 ou moins.

Créer de l'arbitraire

Plus précisément, nous cherchons à déterminer le logarithme d'un élément arbitraire dont la valeur est dérivée des décimales de π. Cette façon de poser le défit initial est courante lorsqu'il s'agit de résoudre le problème du logarithme discret, car elle donne une certaine confiance en le caractère arbitraire du choix de celui-ci. En effet, tout autre choix non motivé par une source d'information extérieure lèverait la suspicion d'une triche : il est facile de procéder d'abord au tirage d'un entier x puis de calculer g^x avant de décréter que g^x sera l'élément dont on cherchera le logarithme, mais cette méthode ne produirait que des éléments déstructurés sans rapport avec les constantes classiques utilisées en mathématiques, comme π justement, ou bien le nombre d'Euler e. Ainsi nous souhaitons déterminer le logarithme discret de :

$$h = \sum_{i=0}^{2394} (\lfloor \pi * 3^{i+1} \rfloor \mod 3) * U^{i \div 5} * \alpha^i \mod 5,$$

où $\div$ représente la division euclidienne. Le degré de h, qui est $2394 \div 5 = 478$ est ainsi maximal, puisque l'extension du corps cible est $479 = 2395/5$ au dessus de $\mathbb{F}_{3^5}$.

Descente par fraction continue

Nous utilisons tout d'abord une étape de descente par fractions continues, non détaillée ici, afin d'exprimer h comme un quotient de polynômes de degré relativement bas. Le plus haut degré apparaissant est alors 50. Cette étape demande 5000 heures CPU que l'on peut distribuer. Pour être franche, cette première étape de descente a été exécutée en amont des précalculs et rétrospectivement elle pourrait être grandement améliorée.

Descentes classique, bilinéaire et en Zig-zag

Une descente classique récursive conduit ensuite à l'expression de ces derniers polynômes en fonction de polynômes de degré 16, 12, et 10 ou moins. Une combinaison des descentes par système bilinéaire de [Jou13b] et de l'approche en Zig-zag de [GKZ14b] permet dans un troisième temps de tout exprimer en termes de polynômes de degré au plus 4, donc appartenant à la base de friabilité étendue. Le temps total pour la descente complète jusqu'au degré 4 se montre légèrement inférieur à 1400 heures CPU. Les logarithmes de tous ces polynômes sont obtenus en moins de 8 heures CPU mais nécessitent l'utilisation d'une machine avec 64G de mémoire afin de pouvoir charger les logarithmes déjà calculés dans les phases préliminaires.

Etape	Paramètres pertinents	Heures CPU
Collecte des relations		< 1 min
Algèbre linéaire	30000 polynômes	16
Extension au degré 3	20000×244 syst. linéaires	1600
Extension au degré 4	$\max 7400 \times 244$ syst. linéaires [2]	500
Descente par fraction continue	Depuis deg 478 jusqu'au deg 50	5000 [3]
Descente classique	Depuis deg 50 jusqu'au deg 16	1000
Descentes bilinéaire + Zigzag [4]	Depuis deg 16 jusqu'au deg 4	400
Recombinaison de la descente		30
Pas de bébé, pas de géant		1 min
Total		8600

Table 4.1 – Décomposition des 8 600 heures CPU de calcul (soit environ un an CPU) pour la résolution du problème du logarithme discret dans le corps cible $\mathbb{F}_{3^{5\cdot 479}}$.

4.4.5 Et voici (enfin !) un logarithme

Le parcours de l'arbre de descente qui permet de rassembler tous les résultats intermédiaires et conclure le calcul du logarithme discret attendu prend de l'ordre de 30 heures CPU.

Redécouvrir le logarithme modulo $2 \cdot 11^2 \cdot 2017549$ ne demande finalement qu'une minute avec l'algorithme des Pas de bébé – Pas de géant.

Le temps d'exécution global de notre algorithme environne ainsi 8600 heures CPU, comme reporté dans le tableau 3.1 et détaillé dans le tableau 4.1. A l'issu de celui-ci nous obtenons le logarithme discret de h :

77505588309444688883926502525134195106654673359423275661795094781
62100521513497892136169254531868849080347908279137658196354900390
64549867418890052769323571492159084777305468528471176288203760651
49535155948391315068830375294252999708082054887926813577325480888
10214865840557638578562739705556907694008232973066293462433770649
45406995423174157468474800166506794795531779808097780548025560211
29564151634653332361630361612835510743393721187917852710681067543
94547160460711088939964483155435722546934168473033179427318725272
10679215932698342472719828852892088509456849503867133033112427319
12854342662964589632571637782776220760760823673502138428497219034
06144006720815440449238920557641092436103031596737885884237055588
42738734115305172373596435722205711435175019406137919975700734056
17095817229836805624052758773516846104312439037228717205677060849
46904254911966901259773507365843097443729343081121960690575079307
62668499229307611483965949654230441206800936422812331741331322998
14145152846675883466792733884157371392343737650965235587269785417
44523158025959589543518783121064616279296755145058161579075485840
65816431752640757811901062480592544 83

dont un script de vérification est donnée dans le paragraphe suivant.

4.4.6 Scripts de vérification

Voici un premier script de vérification en Magma.

```
R:=RealField(3000);
pival:=Pi(R);
gf3pol<x>:=PolynomialRing(GF(3));
gfext<a>:=ext<GF(3)|x^5-x+1>;
gf2pol<x2>:=PolynomialRing(gfext);
q:=3^5;
h0:=x2^2 + a^111*x2;
h1:=a*x2 + 1;

I:=x2*Evaluate(h1,x2^q)-Evaluate(h0,x2^q);
F:=Factorization(I);

DefPol:=F[#F][1];
```

```
gfbig<u>:=ext<gfext|DefPol>;

g:=u+a^2;
Z:=&+[(Floor(pival*3^(val+1)) mod 3)*u^(val div 5)*a^(val mod 5):
val in [0..2394]];
lg:=
77505588309444688883926502525134195106654673359423275661795094781
62100521513497892136169254531868849080347908279137658196354900390
64549867418890052769323571492159084777305468528471176288203760651
49535155948391315068830375294252999708082054887926813577325480888
10214865840557638578562739705556907694008232973066293462433770649
45406995423174157468474800166506794795531779808097780548025560211
29564151634653332361630361612835510743393721187917852710681067543
94547160460711088939964483155435722546934168473033179427318725272
10679215932698342472719828852892088509456849503867133033112427319
12854342662964589632571637782776220760760823673502138428497219034
06144006720815440449238920557641092436103031596737885884237055588
42738734115305172373596435722205711435175019406137919975700734056
17095817229836805624052758773516846104312439037228717205677060849
46904254911966901259773507365843097443729343081121960690575079307
62668499229307611483965949654230441206800936422812331741331322998
14145152846675883466792733884157371392343737650965235587269785417
44523158025959589543518783121064616279296755145058161579075485840
65816431752640757811901062480592544 83;
if Z eq g^lg then
print "Verification OK";
else
print "Verification Failed";
end if;
```

Ci dessous se trouve un script de vérification pour Pari/GP, qui nécessite une version postérieure à Pari 2.5.

```
install(FpXQXQ_pow, GGGGG);
#
\p 3000
allocatemem(200000000)
Z=sum(i=0,2394,(floor(Pi*3^(i+1))%3)*x^(i\5)*a^(i%5));
pola=(a^5-a+1)
q=3^5
polx=x*(a*x^q+1)-(x^(2*q)+a^111*x^q)
fact=factorff(polx,3,pola);
polx=lift(lift(fact[5,1]))
lg=
77505588309444688883926502525134195106654673359423275661795094781
62100521513497892136169254531868849080347908279137658196354900390
64549867418890052769323571492159084777305468528471176288203760651
49535155948391315068830375294252999708082054887926813577325480888
10214865840557638578562739705556907694008232973066293462433770649
```

```
45406995423174157468474800166506794795531779808097780548025560211
29564151634653332361630361612835510743393721187917852710681067543
94547160460711088939964483155435722546934168473033179427318725272
10679215932698342472719828852892088509456849503867133033112427319
12854342662964589632571637782776220760760823673502138428497219034
06144006720815440449238920557641092436103031596737885884237055588
42738734115305172373596435722205711435175019406137919975700734056
17095817229836805624052758773516846104312439037228717205677060849
46904254911966901259773507365843097443729343081121960690575079307
62668499229307611483965949654230441206800936422812331741331322998
14145152846675883466792733884157371392343737650965235587269785417
44523158025959589543518783121064616279296755145058161579075485840
65816431752640757811901062480592544 83;
G = FpXQXQ_pow(x+a^2, lg, polx, pola, 3);
if (G==Z, print("Verification OK"), print("Verification FAILED"))
```

4.5 Conclusion en petite caractéristique

Quelques temps après de courtes années d'activité impressionnante, l'état de l'art du problème du logarithme discret en petite caractéristique, quoique moins dynamique, n'est pourtant probablement pas stabilisée. D'un point de vue pratique, la complexité des phases de précalculs polynomiales a ainsi été réduite à $O(q^6)$ pour les corps finis de la forme $\mathbb{F}_{q^k}$ avec $k < 2q$, comme nous venons de le montrer. En pratique il serait donc intéressant d'étudier plus en détail la descente, qui domine de nouveau les temps de calcul depuis notre amélioration présentée dans ce chapitre.

Vers un algorithme rigoureux ?

En revanche, en théorie, les derniers progrès qui datent de 2015 proposent [GKZ15] une méthode pour réduire les heuristiques employées à une unique hypothèse sur la construction du corps cible. Il suffirait ainsi de prouver l'existence d'une paire de polynômes (h_0, h_1) telle que :

$$h_1(X)X^q - h_0(X)$$

possède un facteur irréductible de degré k. Cette question demeure ouverte, bien qu'en pratique le calcul d'une telle paire de polynôme soit aussi simple que rapide.

Vers un algorithme polynomial ?

Un second problème consiste à trouver, s'il existe, un algorithme polynomial en temps, ce qui nécessite de travailler la encore sur l'étape de descente.

Les techniques actuelles qui sont utilisées pour le calcul de logarithme discret individuel ne permettent pas une telle amélioration. De quelle impasse théorique l'obstacle se dresse-t-il ? Rappelons que la descente peut se schématiser par un arbre dont les noeuds sont des polynômes et les fils d'un noeud sont ceux de plus bas degré qui interviennent directement dans la descente du père – c'est-à-dire que l'on peut exhiber une relation multiplicative qui lie père et fils. Malheureusement, chaque noeud interne présente $O(q)$ fils. Puisque la hauteur de l'arbre est elle-même logarithmique en k, le nombre total de feuilles de l'arbre est déjà plus que polynomial en la taille du corps. Plus précisément, pour k de l'ordre de q, le nombre de feuilles à traiter est en :

$$O(q^{O(\log q)}),$$

c'est-à-dire qu'il domine n'importe quel polynôme en q.

Troisième partie

Le problème du logarithme discret dans les corps finis de moyenne et grande caractéristiques

Chapitre 5

Crible par corps de nombres multiple

Sommaire

L'intégralité de cette dernière partie s'attache à l'étude des deux cas de figure qui présentent actuellement la meilleure sécurité en ce qui concerne le problème du logarithme discret dans les corps finis. Nous étudions ainsi le cas des corps de moyenne caractéristique, et celui des corps de grande caractéristique. L'objet de ce chapitre 5 consiste à développer une variante du crible par corps de nombre, ou NFS (de *Number Field Sieve* en anglais) qui permet d'atteindre des complexités asymptotiques plus basses que celles obtenues précédemment. En plus de cette avancée théorique, nous présentons au chapitre 6 une méthode de résolution de systèmes d'algèbre linéaire qui permet de faciliter en pratique le calcul de logarithmes discrets dans cette double plage de caractéristique.

La méthode que nous proposons s'inspire de deux variantes du crible par corps de nombre. La première, celle de Coppersmith [Cop93], visait l'amélioration de la factorisation de grands entiers, tandis que la seconde que nous devons à Matyukhin s'attachait au calcul de logarithmes discrets dans les corps premiers [Mat03]. Chacune des deux variantes repose sur l'idée d'utiliser non pas une paire de corps de nombre, mais un grand nombre d'entre eux. Bien que l'amélioration de Matyukhin n'ait pas été suivie par une accélération des calculs dans la vie réelle, faute d'implémentation sur des tailles suffisamment pertinentes, elle a tout de même permis de réduire la complexité du problème du logarithme discret dans les corps premiers.

La visée de ce chapitre est double. D'une part nous donnons une extension de l'algorithme de Matyukhin pour tous les corps de grande caractéristique, grâce au rappel d'une sélection polynomiale qui n'était pas connue en 2003 lors de la première adaptation de l'idée de Coppersmith au contexte du logarithme discret. La complexité que nous retrouvons pour ces corps, c'est-à-dire :

$$L_{p^n}\left(1/3, ((92+26\sqrt{13})/27)^{1/3}\right),$$

où $(92+26\sqrt{13})/27)^{1/3} \approx 1.90$, est précisément celle obtenue dans [Mat03] pour les corps premiers.

Nous proposons aussi un autre crible multiple qui permet d'atteindre en moyenne caractéristique une complexité de :

$$L_{p^n}\left(\frac{1}{3}, \left(\frac{8\cdot(9+4\sqrt{6})}{15}\right)^{1/3}\right).$$

où $\left(8\,(9+4\sqrt{6})/15\right)^{1/3} \approx 2.156$. Bien qu'il soit apparu depuis de nombreuses variantes performantes en moyenne caractéristique pour certaines extensions, cet algorithme, présenté dans [Pie15] reste le plus efficace en terme de complexité pour des corps pris en toute généralité.

En passant, nous donnons aussi une idée de crible multiple symétrique, qui, bien que dépassée depuis par la variante qui combine crible multiple et méthode de conjugaison, offre un autre point de vue sur la manière d'extraire des relations. Dans ce cas, l'idée consiste à s'appuyer sur une méthode de sélection polynomiale qui facilite l'équilibre entre les degrés des deux polynômes en jeu et la taille de leurs coefficients. Les autres corps de nombres sont définies par combinaisons linéaires de ces deux premiers polynômes, permettant ainsi que chaque corps de nombres du diagramme puisse jouer un rôle

similaire. Ceci permettait d'atteindre une complexité de :

$$L_{p^n}\left(\frac{1}{3}, \left(\frac{4(46+13\sqrt{13})}{27}\right)^{1/3}\right).$$

où $\left((4(46+13\sqrt{13}))/27\right)^{1/3} \approx 2.396$.

Que ce soit dans le cas symétrique ou dissymétrique et que nous criblions sur des polynômes de haut degré – comme nous le faisons en moyenne caractéristique – ou des polynômes affines – comme en grande caractéristique – nous suggérons de nommer cette variante le crible par corps de nombres multiple ou MNFS (de l'anglais *Multiple Number Field Sieve*).

Par ailleurs, les analyses de complexité effectuées pour ces variantes nous ont donné l'occasion de détailler le coût de la phase de logarithme discret individuel, qui a toujours été considérée comme négligeable dans ces cas de figure, bien qu'aucune analyse n'ait jusqu'ici été publiée.

Organisation du chapitre. Nous commençons par détailler l'état de l'art du crible par corps de nombres dans sa version classique en section 5.1. Nous introduisons ensuite le crible multiple, que ce soit sous sa forme symétrique ou dissymétrique, en partie 5.2. Les sections 5.3 et 5.4 permettent de détailler les analyses de complexité asymptotique réalisées, respectivement les plus performantes d'une part, et celles devenues obsolètes dont nous souhaitons d'autre part et malgré tout laisser une trace. Enfin, nous achevons ce chapitre par un exemple pédagogique donné en section 5.5.

5.1 Crible par corps de nombres

Le crible par corps de nombres est un algorithme qui nous provient de la factorisation. Il a été adapté pour le problème du logarithme discret par Gordon en 1993 [Gor93] pour les corps premiers. Gordon propose aussi au sein du même article une variante pour certaines caractéristiques particulières, toujours pour les corps premiers : il s'agit du premier SNFS (de l'anglais, *Special Number Field Sieve*). Schirokauer, déjà connu pour son travail sur les logarithmes virtuels [Sch93], étend NFS aux extensions de degré fixé [Sch00] en l'an 2000, mais il faut attendre 2006 pour que l'algorithme soit généralisé [JLSV06] à la double plage de caractéristique qui nous intéresse ici. Cette section détaille l'algorithme NFS classique tel qu'il a notamment été présenté dans [JLSV06] par Joux, Lercier, Smart et Vercauteren. Nous nous fixons comme corps cible $\mathbb{F}_{p^n}$ où p est la caractéristique du corps et n son

degré d'extension. Nous notons $Q = p^n$ la cardinalité de ce corps fini. La caractéristique $p = L_Q(l_p, c_p)$ est considérée dans toute la suite telle que $l_p > 1/3$ pour c_p de l'ordre de 1.

5.1.1 Préliminaires et notations

Soit $f \in \mathbb{Z}[X]$ un polynôme irréductible, K le corps de nombres associé à f et θ une racine complexe de f. Alors, pour tout polynôme $\phi \in \mathbb{Z}[X]$, la norme de $\phi(\theta)$ notée $\mathrm{N}(\phi(\theta))$ satisfait :

$$\mathrm{Res}(\phi, f) = \pm f_d^{\deg \phi} \mathrm{N}(\phi(\theta)),$$

où la constante f_d représente le coefficient de tête de f. Puisque nous traiterons les constantes f_d à part, nous ne faisons pas de distinctions ici entre la friabilité de la norme et celle du résultant associé. Soit $\|f\|_\infty$ le plus grand coefficient du polynôme f en valeur absolue. Bien qu'il existe une quantité non négligeable de bornes sur les résultants, nous serons souvent amenés à n'utiliser que l'une d'entre elle, qui n'est autre que la majoration suivante :

$$|\mathrm{Res}(f, \phi)| \leq (\deg f + \deg \phi)! \cdot \|f\|_\infty^{\deg \phi} \cdot \|\phi\|_\infty^{\deg f}. \tag{5.1}$$

Nous avons déjà évoqué le théorème phare de l'évaluation des probabilités de friabilité des entiers. Sous sa version formelle, le voici à nouveau :

Théorème 5.1.1 (Canfield, Erdős, Pomerance [CEP83]). *Soit $\psi(A, B)$ le nombre d'entiers naturels inférieurs à A qui sont B-friables. Si nous avons de plus $\epsilon > 0$ et $3 \leq u \leq (1 - \epsilon) \log A / \log \log A$, alors :*

$$\psi(A, A^{1/u}) = A u^{-u + o(u)}.$$

La notation en L_q permet en réalité de simplifier l'usage de ce résultat. Si nous réécrivons les deux entiers A et B grâce à celle-ci, nous obtenons le corollaire utile :

Corollaire 5.1.2. *Soit $(\alpha_1, \alpha_2, c_1, c_2) \in [|0, 1|]^2 \times [|0, \infty[^2$ quatre réels tels que $\alpha_1 > \alpha_2$ ou tels que $\alpha_1 = \alpha_2$ and $c_1 > c_2$. Désignons par ailleurs par $\mathcal{P}$ la probabilité qu'un entier naturel aléatoire inférieur à $A = L_q(\alpha_1, c_1)$ puisse se factoriser en termes tous inférieurs à $B = L_q(\alpha_2, c_2)$. Alors :*

$$\mathcal{P}^{-1} = L_q\left(\alpha_1 - \alpha_2, (\alpha_1 - \alpha_2) c_1 c_2^{-1}\right).$$

5.1.2 Sélections polynomiales

Représentation du corps cible

Comme pour les corps de petite caractéristique, nous commençons par une étape préliminaire visant à trouver une représentation agréable du corps cible. Afin de résoudre le problème du logarithme discret dans $\mathbb{F}_{p^n}$, nous choisissons tout d'abord deux polynômes f_1 et f_2 dans $\mathbb{Z}[X]$ tels qu'ils partagent une racine commune m dans $\mathbb{F}_{p^n}$. En d'autres termes, nous sélectionnons f_1 et f_2 pour que leur plus grand facteur commun possède lui-même un facteur irréductible de degré n au dessus du corps de base $\mathbb{F}_p$. C'est ce polynôme irréductible qui permet de définir le corps cible. Par conséquent, nous pouvons tracer le diagramme commutatif de la figure 5.1. Afin de ne pas tracer un diagramme trivial, c'est-à-dire pour lequel les deux branches seraient identiques, nous veillons à ce que les deux polynômes f_1 et f_2 diffèrent dans $\mathbb{Z}[X]$.

Désignons par $\mathbb{Q}(\theta_1)$ le corps de nombres défini par le polynôme f_1, c'est-à-dire $\mathbb{Q}[X]/(f_1(X))$ et par $\mathbb{Q}(\theta_2)$ celui défini par le second polynôme f_2, donc égal à $\mathbb{Q}[X]/(f_2(X))$. Ainsi, θ_1 et θ_2 sont des racines respectives de ces deux polynômes dans $\mathbb{C}$.

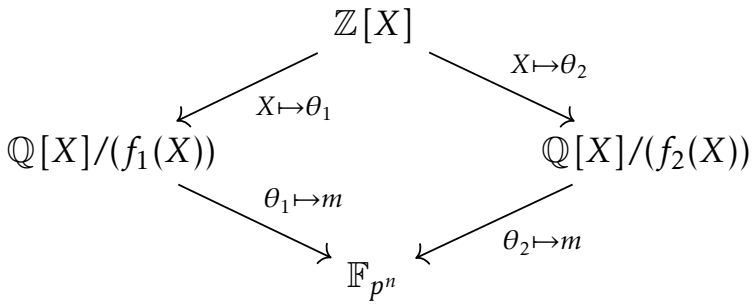


Figure 5.1 – Diagramme commutatif du crible par corps de nombres

Choix des polynômes

De nombreuses méthodes ont été proposées concernant le choix de ces polynômes. Nous ne traitons ici que le cas de celles qui nous seront utiles pour la description de nos algorithmes.

Méthode de Joux-Lercier-Smart-Vercauteren simple. La première option donnée par [JLSV06] et probablement le plus naturel des exemples à avancer débute avec l'idée suivante. Nous souhaitons construire le corps cible avec l'aide d'un polynôme irréductible de degré n au-dessus de $\mathbb{F}_p$, aussi, nous

	degré	taille des coefficients
f_1	n	$o(1)$
f_2	n	p

Table 5.1 – Paramètres obtenus pour la méthode de JLSV simple, utilisée en moyenne caractéristique

prenons tout d'abord un polynôme $f_1 \in \mathbb{Z}[X]$ de degré n, irréductible sur $\mathbb{F}_p$ et à petits coefficients. Comme nous cherchons un second polynôme dont un facteur commun définisse le corps cible, c'est-à-dire puisque nous cherchons un multiple de f_1 modulo p, qui diffère pourtant de f_1, nous posons simplement $f_2 = f_1 + p$.

Sélection polynomiale par fraction continue. Toujours dans [JLSV06], les auteurs proposent une sélection polynomiale qui permet d'équilibrer les normes calculées au cours de l'algorithme, et qui passe par le choix de polynômes de même degré et même taille de coefficients. Plus précisément, $f_1 \in \mathbb{Z}[X]$ est choisi pour être un polynôme de degré n, irréductible modulo p. Il sera donc utilisé implicitement pour définir le corps cible. Nous supposons qu'il s'écrit comme :

$$f_1 = g_a + c \cdot g_b$$

avec g_a et g_b deux polynômes auxiliaires de petits coefficients, et tel que c soit de taille $O(\sqrt{p})$. Par conséquent, la taille des coefficients est de l'ordre de $\sqrt{p}$. Grâce à la méthode des fractions continues, nous pouvons trouver a et b tels que $c \equiv a/b \pmod{p}$ avec a et b deux entiers[1] tous deux de l'ordre de $\sqrt{p}$. Nous définissons ensuite f_2 par l'égalité modulaire :

$$f_2 \equiv b f_1 \mod p.$$

Par conséquent, f_1 constitue le plus grand facteur commun des deux polynômes modulo p, il est irréductible et de bon degré, donc permet de construire

1. Une autre manière de trouver a et b consiste à décomposer p en base c. On a alors $p = b'c + a'$, donc $c = -a'/b' \mod p$. Par ailleurs b' est bien de l'ordre de de $\sqrt{p}$ et a' aussi.

	degré	taille des coefficients
f_1	n	$O(\sqrt{p})$
f_2	n	$O(\sqrt{p})$

Table 5.2 – Paramètres obtenus pour la méthode des fractions continues, utilisée en moyenne caractéristique.

le diagramme comme attendu. Nous notons que les deux polynômes jouent un rôle symétrique : f_1 et f_2 sont tout deux de degré n et possèdent des coefficients de taille $O(\sqrt{p})$, comme récapitulé dans le tableau 5.2. En effet, puisque l'on peut écrire l'égalité modulaire $f_2 = bf_1 = b \cdot g_a + a \cdot g_b \mod p$, le polynôme f_2 expose des coefficients du même ordre de grandeur que a et b.

Méthode par conjugaison. Si les deux premières méthodes remontent à 2006, ce choix de polynômes a été proposé en 2014 dans l'article [BGGM15, Paragraphe 6.3]. Nous transformons légèrement l'exposé du procédé, sans changer l'idée sous-jacente, afin de simplifier le détail ultérieur du crible multiple.

La méthode proposée commence par la sélection de deux polynômes auxiliaires dans $\mathbb{Z}[X]$, que nous notons g_a et g_b, que nous supposons à petits coefficients, et tels que $\deg g_a = n$ et $\deg g_b < n$. Nous cherchons ensuite un polynôme irréductible quadratique et unitaire $X^2 + uX + v$ sur $\mathbb{Z}[X]$, de telle sorte que u et v soient de petits entiers de taille $O(\log p)$, et tel que les racines λ et λ' de ce polynôme appartiennent au corps de base $\mathbb{F}_p$. Puisque nous cherchons encore une fois un polynôme de degré n irréductible sur $\mathbb{F}_p[X]$ pour construire le corps cible, nous conservons ce polynôme $X^2 + uX + v$ si l'un des deux polynômes $g_a + \lambda g_b$ ou $g_a + \lambda' g_b$, qui sont de degré n, se trouve être aussi irréductible sur $\mathbb{F}_p[X]$. Supposons par exemple que :

$$g_a + \lambda g_b$$

soit irréductible sur $\mathbb{F}_p[X]$. Nous définissons alors le premier polynômes $f_1 \in \mathbb{Z}[X]$ via l'égalité :

$$f_1 = g_a^2 - u g_a g_b + v g_b^2.$$

	degré	taille des coefficients
f_1	$2n$	$O(\log p)$
f_2	n	$O(\sqrt{p})$

Table 5.3 – Paramètres obtenus pour la méthode de conjugaison, utilisée en moyenne caractéristique.

De manière équivalente, f_1 est défini dans l'article originel [BGGM15] comme étant égal à $\mathrm{Res}_Y(Y^2 + uY + v, g_a(X) + Yg_b(X))$. Puisque λ et λ' sont les racines de $X^2 + uX + v$ dans $\mathbb{F}_p$, nous pouvons écrire l'égalité polynomiale $f_1 \equiv g_a^2 + (\lambda + \lambda')g_a g_b + \lambda\lambda' g_b^2 \mod p$. En d'autres termes :

$$f_1 \equiv (g_a + \lambda g_b)(g_a + \lambda' g_b) \mod p.$$

Nous obtenons donc un polynôme f_1 de degré $2n$ à coefficients de taille $O(\log p)$, qui est un multiple de $g_a + \lambda g_b$ dans $\mathbb{F}_p[X]$.

Nous construisons ensuite le second polynôme par fraction continue. En effet, puisque nous pouvons écrire :

$$\lambda \equiv \frac{a}{b} \mod p$$

avec a et b de l'ordre de $\sqrt{p}$, il est possible de poser :

$$f_2 = bg_a + ag_b.$$

Ce polynôme f_2 est donc bien lui aussi de degré n et à coefficients de l'ordre de $\sqrt{p}$. De plus, comme $f_2 \equiv b(g_a + \lambda g_b) \mod p$, il partage bel et bien une racine commune dans $\mathbb{F}_{p^n}$ avec le polynôme f_1. Les paramètres de ces deux polynômes sont résumés dans le tableau 5.3.

Remarque 5.1.3. Nous corrigeons en détaillant cette méthode une légère erreur commise initialement dans [BGGM15, Paragraph 6.3] lors de l'évaluation de la taille des entiers u et v. Les auteurs proposent en effet de choisir un polynôme quadratique unitaire dont les coefficients sont de taille constante. Malheureusement, si les valeurs $|u|$ et $|v|$ sont toutes deux majorées par une

constante C, alors il n'existe que $4C^2$ polynômes quadratiques satisfaisants. Puisque chacun d'entre eux a une probabilité $1/2$ d'avoir ses racines dans $\mathbb{F}_p$ pour un choix aléatoire de p, pour un nombre premier p donné, la probabilité qu'aucun de ces polynômes quadratiques ne permette notre construction est de 2^{-4C^2}. Si nous tentons de choisir notre polynôme X^2+uX+v pour approximativement 2^{4C^2} nombres premiers, nous tomberons sur un corps premier $\mathbb{F}_p$ pour lequel cette méthode ne pourra fonctionner. En revanche, chercher des polynômes quadratiques à coefficients de l'ordre de $O(\log p)$ contourne cet obstacle et n'interfère par sur l'évaluation finale des complexités asymptotiques.

Méthode de Joux-Lercier généralisée. En 2003, Joux et Lercier [JL03] proposent un choix de polynôme adapté au calcul de logarithmes discrets dans les corps premiers. Cette construction, qui utilise de la réduction de réseau, offre en 2006 la possibilité à Joux, Lercier, Smart et Vercauteren d'atteindre tous les corps de grande caractéristique [JLSV06]. Barbulescu, Gaudry, Guillevic et Morain généralisent cette méthode en 2015 [BGGM15, Paragraphe 6.2]. Si les tableau 5.4 et 5.5 récapitulent les tailles des paramètres obtenus dans [JLSV06] et [BGGM15] pour un certain paramètre d entier à choisir, nous ne détaillons ici que la construction la plus récente. Néanmoins nous souhaitons souligner que l'essentiel du gain asymptotique donné par l'utilisation de la réduction de réseau était déjà présent dans [JLSV06]. Cette dernière construction a uniquement modifié les complexités du cas frontière, dans le cas particulier où $p = L_Q(2/3, c)$ pour certains réels c.

Quelle est donc cette sélection polynomiale dite de Joux-Lercier généralisée présentée dans [BGGM15] ? Afin de construire le diagramme contenant notre corps cible $\mathbb{F}_{p^n}$, nous choisissons tout d'abord un polynôme irréductible f_1 dans $\mathbb{F}_p[X]$, à petits coefficients (pour les mêmes raisons que précédemment nous supposons que les tailles de ceux-ci sont en $O(\log p^n)$), et tel qu'il possède un facteur irréductible φ du bon degré, c'est-à-dire de degré n modulo p. Nous pouvons supposer sans perte de généralité que ce facteur est unitaire. Ecrivons maintenant pour plus de clarté $\varphi = X^n + \sum_{i=0}^{n-1} \varphi_i X^i$ et notons $d+1$ le degré de f_1. Nous obtenons alors l'inégalité stricte $d+1 > n$. Contrairement à d'autres sélections polynomiales, nous exigeons ici que le facteur irréductible φ soit différent de f_1, puisque nous devons nous assurer que f_2 ne soit pas égal à $f_1 \bmod p$.

Pour certifier que le second polynôme à construire partage bien ce facteur irréductible modulo p, nous le définissons comme combinaison linéaire de polynômes de la forme φX^k et pX^k. Au passage, nous remarquons que $f_1 = \varphi$ pourrait bel et bien entrainer $f_1 \equiv f_2 \mod p$, ce que nous souhaitons éviter.

	degré	taille des coefficients
f_1	n	$p^{n/(d+1)}$
f_2	d	$p^{n/(d+1)}$

Table 5.4 – Paramètres obtenus pour la méthode de Joux, Lercier, Smart et Vercauteren, utilisée en grande caractéristique.

Cette idée de considérer une combinaison linéaire de ces polynômes, donc le réseau formé par ceux-ci, s'améliore naturellement grâce à un algorithme de réduction de réseau, qui nous permet alors d'obtenir de petits coefficients pour f_2. Plus précisément, notons M la matrice de dimension $(d+1)\times(d+1)$ qui suit :

$$M = \underbrace{\begin{pmatrix} p & & & \\ & \ddots & & \\ & & p & \\ & & & \end{pmatrix}}_{n \text{ colonnes}} \underbrace{\begin{pmatrix} 1 & & 1 \\ \varphi_{n-1} & \ddots & \varphi_{n-1} \\ \vdots & \ddots & \vdots \\ \varphi_0 & & \varphi_0 \\ & \ddots & \\ & & \end{pmatrix}}_{d+1-n \text{ colonnes}} \begin{matrix} X^d \\ X^{d-1} \\ \vdots \\ X^{n-1} \\ \vdots \\ 1 \end{matrix}$$

Un générateur de ce réseau de polynômes est représenté par une colonne : chacun de ses coefficients est ainsi retranscrit dans la ligne correspondant au monôme associé. Les indications à droite de la matrice éclairent cette représentation. Le déterminant de cette matrice est p^n, sa dimension est $d+1$. Par conséquent, l'exécution de l'algorithme LLL sur la matrice M permet d'obtenir un polynôme de degré au plus d dont les coefficients sont de taille au plus $p^{n/d+1}$. Nous faisons l'hypothèse en effet que $2^{(d+1)/4}$ reste petit comparativement à $p^{n/d+1}$.

Le tableau 5.5 résume la taille des paramètres que nous obtenons. Les deux polynômes f_1 et f_2 partagent par ailleurs un facteur irréductible de degré n sur $\mathbb{F}_p[X]$.

	degré	taille des coefficients
f_1	$d+1>n$	$O(\log p^n)$
f_2	d	$p^{n/(d+1)}$

Table 5.5 – Paramètres obtenus pour la méthode de Joux-Lercier généralisée, utilisée en grande caractéristique.

5.1.3 Collecte des relations

Une fois le diagramme tracé, cette première étape vise à la création de relations dans le corps de nombres et crible pour cela sur un ensemble de polynômes de degré $t-1$, pris dans l'ensemble de départ du diagramme. Plus précisément, nous fixons au préalable deux bornes, l'une que l'on nomme S et qui désigne la borne de crible (de l'anglais *Sieve*), l'autre notée B et qui représente la borne de friabilité. La base de friabilité est ainsi définie comme l'image dans le corps cible des idéaux premiers de normes inférieures à B dans chacun des deux corps de nombres. Par abus de langage, il est souvent considéré que la base de friabilité n'est pas dans le corps cible, mais constituée de l'union des deux ensembles antécédents à celle-ci.

Nous considérons ensuite tous les t-uplets d'entiers $(a_0, \cdots, a_{t-1}) \in [1, S] \times [-S, S]^{t-1}$ tels que, pour $\phi(x) = \sum_{j=0}^{t-1} a_i x^i$, les normes $\mathrm{N}(\phi(\theta_1))$ et $\mathrm{N}(\phi(\theta_2))$ dans chacun des deux corps de nombres soient B-friables. Les relations s'obtiennent après un post-traitement complexe décrit dans [JLSV06] et que nous épargnons au lecteur dans ce manuscrit. Chacun de ces t-uplets donne ainsi naissance à une équation linéaire entre les *logarithmes virtuels des idéaux* provenant de chacun des deux corps de nombres et appartenant à la base de friabilité.

Grande caractéristique

La phase de crible comme décrite précédemment constitue l'un des deux points de divergence entre le crible par corps de nombres en moyenne caractéristique et celui en grande caractéristique. Tout d'abord, les choix de polynômes pour la définition du diagramme changent avec la taille de la ca-

ractéristique car les sélections polynomiales optimales différent suivant chacun des deux cas considérés. Ensuite, lorsque la caractéristique grandit relativement à la taille du corps cible, il est possible de cribler sur un espace plus petit. En grande caractéristique, poser $t = 2$, c'est-à-dire cribler sur des polynômes de degré 1, suffit en pratique à l'obtention d'un nombre adéquats de relations.

Le cas frontière

Lorsque $p = L_Q(2/3)$, rien n'interdit aucune des deux variantes ci-dessus de s'appliquer. Cette configuration particulière nécessite donc la prise en compte des complexités asymptotiques des différents algorithmes possibles pour déterminer lequel d'entre-eux sera le plus performant. Comme nous le verrons plus loin, cette étude donne lieu à d'étranges phénomènes de complexité. En particulier, le cas frontière entre la moyenne et la grande caractéristique présente en certains points les complexités les plus basses obtenues pour un corps fini général de caractéristique supérieure à $L_Q(1/3)$.

5.1.4 Algèbre linéaire

Une fois les relations récoltées, nous devons résoudre le système linéaire creux associé modulo $p^n - 1$, la cardinalité du groupe multiplicatif $\mathbb{F}_{p^n}^*$, afin de retrouver les *logarithmes des idéaux* de la base de friabilité. Plus précisément, cette étape d'algèbre linéaire est de nouveau exécutée modulo un grand facteur de cette cardinalité, tandis que les petits facteurs sont traités à part. Les logarithmes finaux sont retrouvés grâce à une combinaison des algorithmes Rho de Pollard et de Pohlig-Hellman.

5.1.5 Logarithme individuel

Les logarithmes des éléments de la base de friabilité ayant été déterminés, nous cherchons ici à retrouver le logarithme d'un élément arbitraire, que nous appelons h. Pour cette dernière étape de descente, nous nous appuyons sur l'approche mise en avant par [JLSV06] qui repose sur une technique dite en *spécial*-q.

Si h est un élément de $\mathbb{F}_{p^n} \simeq \mathbb{F}_p[X]/(f_1(X))$ et est représenté par le polynôme $h(X) \in \mathbb{Z}[X]$ à coefficients normalisés dans $[\![-p/2, p/2]\!]$, nous rappelons que nous pouvons le relever dans chacun des deux corps de nombres. Nous noterons ainsi $\bar{h}$ l'élément $h(\theta_1)$ du corps de nombres $\mathbb{Q}(\theta_1)$. Afin de retrouver le logarithme discret de h, nous décomposons l'algorithme en deux

sous-étapes : la première consiste en une descente par fraction continue (*continued fraction descent* ou *smoothing step* en anglais), la seconde en une descente par spécial-$\mathfrak{q}$. Au sein de la descente par fraction continue il est d'usage de chercher un entier e tel que, pour :

$$h' = h^e,$$

la norme de $\bar{h'}$ est C–friable et sans carrée. La dernière exigence sur les carrés absents implique que seuls des idéaux de degré 1 apparaitront dans la factorisation de $\bar{h'}$. Une fois un tel h' en main, nous factorisons l'idéal principal engendré par $\bar{h'}$ en produits d'idéaux premiers de petites normes. Certains d'entre-eux n'appartiennent pas à la base de friabilité – ce sont ceux dont la norme est inférieure à C mais supérieure à B. Notons $\mathfrak{q}$ un tel idéal. Pour calculer son logarithme, nous procédons de manière récursive en diminuant progressivement la borne sur la norme des idéaux, jusqu'à atteindre B. Au final nous retrouvons le logarithme de $\bar{h'}$ et par conséquent celui de h.

Remarque 5.1.4. La méthode que nous venons d'esquisser n'est ni la plus récente, ni la plus rapide. En revanche, il s'agit de celle, classique, utilisée à l'heure de la publication de l'article [BP14] dont nous avions donnée une analyse détaillée. Le lecteur intéressé par cette étape – qui ne domine ni en théorie, ni en pratique les calculs – est invité à se diriger vers les travaux achevés dans [Gui15, Gui16].

5.1.6 Complexités asymptotiques

Cas d'un corps général

La base de friabilité comme l'espace de crible sont choisis de l'ordre de $L_Q(1/3)$. Avec de tels paramètres, une fois optimisé, le crible par corps de nombres atteint les complexités heuristiques asymptotiques suivantes :

- $L_Q(1/3, (128/9)^{1/3})$ pour les corps finis de moyenne caractéristique, en utilisant la méthode de sélection polynomiale par fraction continue [JLSV06].
- $L_Q(1/3, (96/9)^{1/3})$ toujours pour les corps finis de moyenne caractéristique, en utilisant cette fois-ci la sélection polynomiale par conjugaison [BGGM15].
- $L_Q(1/3, (64/9)^{1/3})$ pour les corps finis de grande caractéristique, en utilisant soit la méthode de sélection polynomiale par réduction de réseau proposée dans [JLSV06], soit la méthode de Joux-Lercier généralisée de [BGGM15].

Les complexités asymptotiques du cas frontières seront quant à elles représentées plus loin. Hors crible multiple à venir, une comparaison des différentes

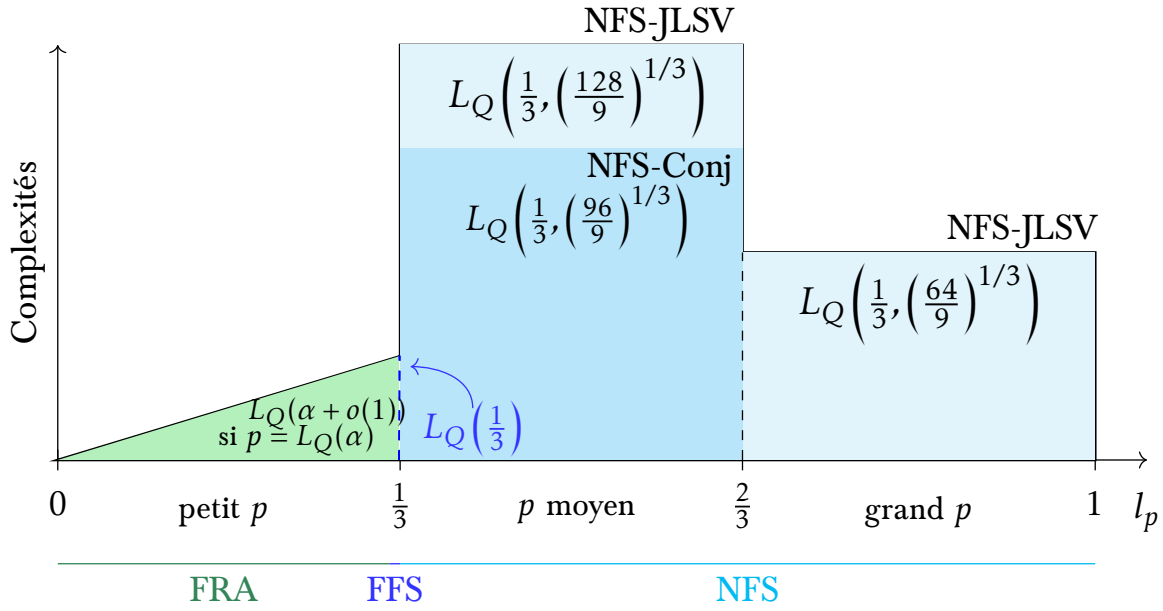


Figure 5.2 – *Algorithmes de calcul de logarithme discret dans les corps finis généraux, et leurs complexités, avant le crible multiple.* Pour un corps fini de cardinalité $Q = p^n$ fixée, ce dessin montre comment les complexités asymptotiques varient en fonction de l_p avec $p = L_{p^n}(l_p)$.

complexités se trouve illustrée aux figures 5.2 et 5.3. La première figure résume l'algorithme à choisir étant donné un corps fini fixé tandis que la seconde montre quels sont les types de corps qui sont faibles à complexité donnée. Les axes de la figure 5.3 peuvent sembler surprenant au lecteur non initié ; ils traduisent en réalité la comparaison qui se fait naturellement entre n et $\log p$ – et non p – lorsque l'on souhaite déterminer la taille de la caractéristique.

Quelques variantes et cas particuliers

Certaines variations autours de NFS permettent d'atteindre des complexités plus basses pour les corps finis présentant soit des caractéristiques particulières, soit des extensions particulières. Dans le premier cas, on parle alors d'un SNFS comme dans [Gor93, JP13] ; dans le second, il s'agit par exemple d'un degré d'extension composé, comme dans les différentes variantes [KB16, SS16d, SS16a, JK16, SS16b] de l'algorithme exTNFS (de l'anglais *EXtended Tower Number Field Sieve*), provenant toutes du TNFS (de l'anglais *Tower Number Field Sieve*) introduit initialement par Schirokauer en 2000 [Sch00] puis analysé dans [BGK15, SS16c]. Il est aussi possible de combiner ces deux variantes pour viser des corps finis de caractéristique *et* de degré d'extension

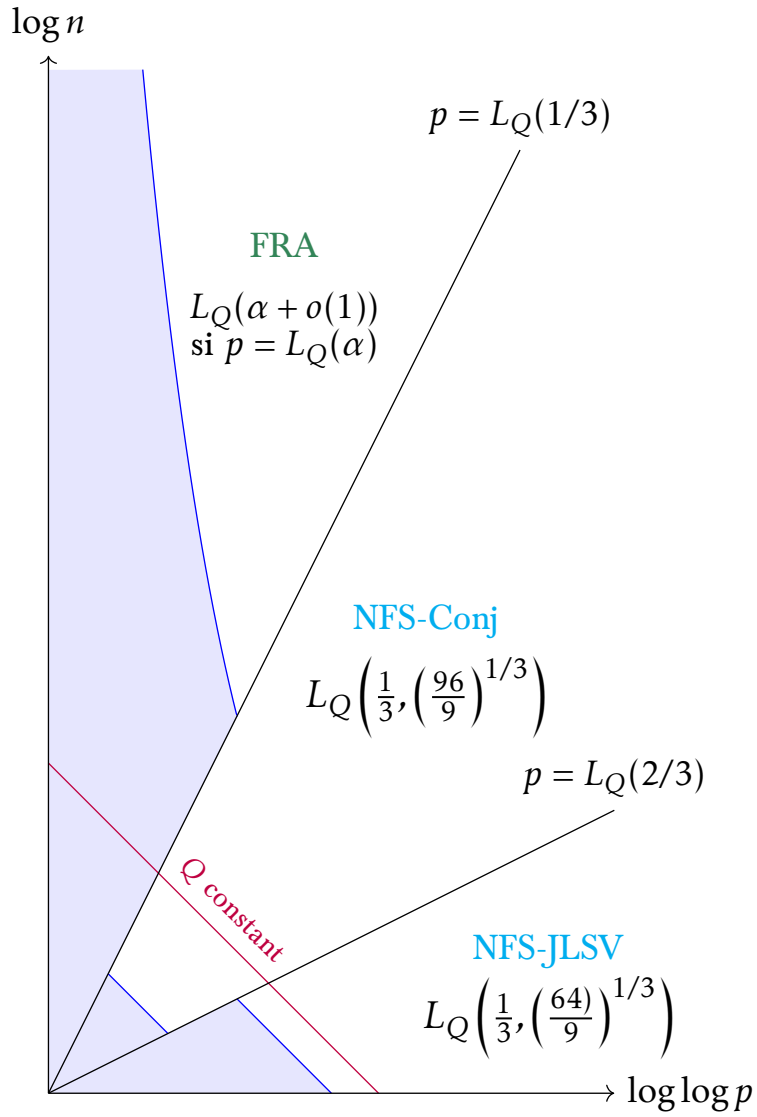


Figure 5.3 – *Domaines des algorithmes de calcul de logarithme discret sur* $\mathbb{F}_{p^n}$. À chaque cardinalité $Q = p^n$ peuvent correspondre plusieurs valeurs de p et de n, comme l'indique la ligne rouge. En fonction de la taille relative de l'un et de l'autre, ce schéma détermine le domaine dans lequel se trouve le corps cible correspondant. Mieux, la ligne bleue est une ligne d'iso-complexité : pour une complexité donnée c, le problème du logarithme discret sur chacun des points de la ligne (respectivement dans la partie bleue du dessin) peut être résolu avec un algorithme dont la complexité est exactement c (respectivement inférieure à c).

particulières – c'est le très élégant SexTNFS proposé dans [JK16].

5.1.7 NFS en pratique

Un lecteur intéressé par les calculs pratiques récents trouvera des informations dans [GGV16], qui travaille sur la collecte des relations et discute des différents choix de sélection polynomiale ou au sein des articles [Gui15, Gui16] qui traitent des questions relatives à la phase de logarithme discret individuel.

5.2 Crible par corps de nombres multiple

5.2.1 Diagramme multi-branches

Le crible par corps de nombres classique repose sur un diagramme à deux branches de la forme de celui donné dans la figure 5.1, qui mènent toutes les deux au corps cible $\mathbb{F}_{p^n}$. L'amélioration que nous proposons dans ce chapitre s'attache précisément à modifier la nature de ce diagramme, en construisant non plus une seule paire de corps de nombres, mais un grand nombre de branches et donc de chemins possibles vers le corps cible.

Supposons que l'on puisse construire $\mathbb{F}_{p^n}$ de V façons différentes, de sorte de pouvoir expliciter un diagramme commutatif comme celui de la figure 5.4. De nouveau, pour chaque $i \in [\![1, V]\!]$, θ_i est une racine dans $\mathbb{C}$ du polynôme f_i et $\mathbb{Q}(\theta_i)$ désigne le corps de nombres $\mathbb{Q}[X]/(f_i(X))$. Pour garantir la commutativité du diagramme, nous supposons qu'il existe une racine commune $m \in \mathbb{F}_{p^n}$ à tout ces polynômes. La connaissance de deux polynômes f_1 et f_2 qui possèdent une racine commune m nous permet par exemple, par combinaisons linéaires de f_1 et f_2, de sélectionner un grand nombre de polynômes qui partageront cette même racine dans $\mathbb{F}_{p^n}$. S'il s'agit de l'idée la plus simple pour augmenter le nombre de branches du diagramme, nous verrons plus loin d'autres manières de procéder.

A première vue, on pourrait imaginer construire un système linéaire relatif à chacune des paires de polynômes de définition (f_i, f_j) avec i et j des indices tels que $1 \leq i < j \leq V$. Notre algorithme va pourtant au delà de cette première intuition : en bref, il s'agit de tirer partie du fait que chaque équation linéaire concerne les logarithmes des éléments du corps cible, et oublie d'une certaine manière la paire de corps de nombres qui l'a produite. Nous parvenons ainsi à décrire une méthode ne traitant qu'une seule et unique matrice d'algèbre linéaire.

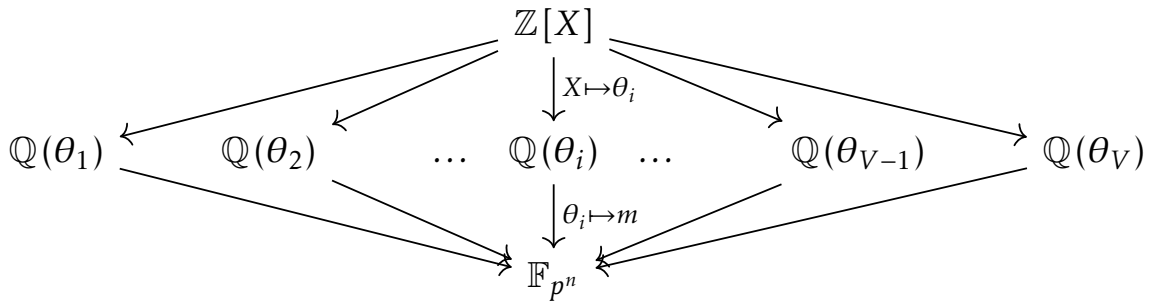


Figure 5.4 – Diagramme commutatif pour le crible par corps de nombres multiples

5.2.2 Crible symétrique

Ce paragraphe reprend l'idée originale du crible par corps de nombres telle que présentée en moyenne caractéristique dans [BP14]. Il s'agit aujourd'hui d'une variante dépassée par un autre type de crible multiple, dissymétrique cette fois-ci, présenté quant à lui dans [Pie15].

Construction du diagramme

Supposons que f_1 et f_2 soient donnés par l'une ou l'autre des sélections polynomiales classiques de NFS. Par exemple, dans le cas des corps finis de moyenne caractéristique, nous proposions dans [BP14] de s'appuyer sur la méthode par fraction continue, qui présente deux polynômes équilibrés : f_1 et f_2 sont tous les deux de degré n et possèdent des coefficients de même taille, c'est-à-dire de l'ordre de $\sqrt{p}$.

Les polynômes suivants sont construits par combinaison linéaire. Ainsi, pour tout indice $i \in [\![3, V]\!]$ nous posons :

$$f_i = \alpha_i f_1 + \beta_i f_2.$$

Puisque nous souhaitons obtenir V corps de nombres au final, les coefficients α_i et β_i sont de l'ordre de $\sqrt{V}$. Nous vérifierons à la section 5.4.1 que V reste négligeable par rapport à d'autres paramètres. En effet, si le crible par corps de nombres classique utilise une valeur de V égale à 2, nous conseillons dans ce chapitre de choisir V de l'ordre de $L_Q(1/3)$, qui reste négligeable comparé à une caractéristique de taille moyenne ou grande.

Si f_1 et f_2 ont une racine commune dans $\mathbb{F}_{p^n}$, il en va de même pour chacun des nouveaux polynômes créés, ce qui autorise le tracé d'un diagramme commutatif de la forme donnée par la figure 5.4, Par ailleurs, l'utilisation de

combinaisons linéaires permet de transférer toutes les propriétés de degré et de tailles de coefficients de f_1 et f_2 aux polynômes nouvellement construits.

Une unique borne de friabilité

Comme dans le crible classique, t désigne le nombre de coefficients des polynômes sur lesquels nous criblons, S est la borne de crible et B la borne de friabilité. Dans cette variante symétrique, nous ne choisissons qu'une seule borne de friabilité, identique à chacun des corps de nombres qui apparaissent dans le diagramme. La nouvelle base de friabilité consiste en l'image dans le corps fini de l'union sur chacun des V corps de nombres de tous les idéaux premiers de degré 1 dont les normes sont inférieures à B. Encore une fois, par abus de langage, la base de friabilité sera souvent identifiée à ces petits idéaux.

Polynômes doublement friables

Nous considérons tous les polynômes de degré $t-1$ de la forme $\phi = a_0 + \cdots + a_{t-1}x^{t-1}$ où les coefficients sont bornés par S en valeur absolue. En pratique, nous criblons uniquement sur les t-uplets d'entiers premiers entre-eux $(a_0, \ldots, a_{t-1})$ pour lesquels a_0 est positif. Ces deux dernières restrictions n'ont cependant pas d'effet sur les calculs de complexités asymptotiques, aussi nous ne les prendrons pas en considération dans la suite. Nous conservons ensuite tous les polynômes ϕ pour lesquels il existe une paire d'indices $(i,j) \in [\![1, V]\!]^2$ avec $i \neq j$ telle que $\mathrm{N}(\phi(\theta_i))$ et $\mathrm{N}(\phi(\theta_j))$ soient toutes les deux B-friables. Nous dirons d'un tel polynôme qu'il est alors *doublement friable*. Un post traitement identique à celui de [JLSV06], nous donne ensuite, pour chaque polynôme doublement friable, une équation linéaire entre les *logarithmes des idéaux* provenant des deux corps de nombres $\mathbb{Q}(\theta_i)$ et $\mathbb{Q}\left(\theta_j\right)$. Il s'agit ainsi d'une équation linéaire entre un petit nombre de logarithmes des idéaux de la base de friabilité.

L'algèbre linéaire est effectuée comme au paragraphe 5.1.4. En ce qui concerne la dernière étape de logarithme individuel, il est possible de suivre une descente similaire à celle du paragraphe 5.1.5. D'un point de vue théorique, nous pouvons aussi adapter celle-ci au cas du crible multiple : si la descente par fraction continue reste inchangée, la descente par spécial-$\mathfrak{q}$ peut tirer parti du grand nombre de corps de nombres à disposition. Le logarithme de chaque idéal $\mathfrak{q}$ peut ainsi s'exprimer comme combinaison des idéaux premiers de degré 1 de norme friable dans n'importe quelle paire de corps de nombres (et non plus uniquement pour les deux premiers). Nous donnerons plus de

détails sur cette étape, somme toute négligeable, au moment de l'analyse de complexité de la phase correspondante, c'est-à-dire au paragraphe 5.4.3.

5.2.3 Crible dissymétrique

Le crible dissymétrique que nous proposons est une extension de la variante de Matyukhin dans laquelle le premier corps de nombres joue un rôle particulier. Cette différence provient du caractère déséquilibré des sélections polynomiales sur lesquelles se basent chacune des deux méthodes que nous exposons ici, l'une pour les corps de grande caractéristique, l'autre pour les corps de moyenne caractéristique. L'idée reste malgré tout identique dans l'une et l'autre de ces deux variantes.

Sélection polynomiale en grande caractéristique

Lorsque le corps cible présente une grande caractéristique, le choix des polynômes de définition des corps de nombres demeure d'une facilité impressionnante. Il suffit par exemple de choisir f_1 et f_2 comme décrits dans la méthode par réduction de réseau de [JLSV06] qui, étant donné un paramètre d, permet d'obtenir :

$$\begin{aligned} \deg f_1 &= n, \quad \|f_1\|_\infty = p^{n/(d+1)}, \\ \deg f_2 &= d, \quad \|f_2\|_\infty = p^{n/(d+1)}. \end{aligned}$$

avec $\|f\|$ une borne sur la valeur absolue du plus gros coefficient de f. L'un des polynômes possède un degré plus haut que l'autre, mais les tailles de coefficients sont du même ordre. Aussi, considérer les combinaisons linéaires de ces deux objets initiaux ne donnera pas des paramètres plus mauvais que ceux déjà obtenus pour le polynôme de plus haut degré. Plus précisément, nous posons une nouvelle fois pour tout $i \in [\![3, V]\!]$:

$$f_i = \alpha_i f_1 + \beta_i f_2,$$

où α_i et β_i sont des entiers de taille $\sqrt{V}$. Lorsque nous choisissons V de l'ordre de $L_Q(1/3)$, il reste négligeable par rapport à $\|f_1\|_\infty$. Tous les polynômes créés ont donc même taille de coefficients, et même degré, sauf pour l'un d'entre eux, que nous notons à nouveau f_1, et qui possède un degré plus bas.

La variante dissymétrique en grande caractéristique diffère aussi par son espace de crible, que nous pouvons prendre plus petit, comme nous le verrons. En effet, comme pour NFS dans sa version classique, cribler sur des polynômes de degré 1 suffit heuristiquement à obtenir le nombre de relations que nous souhaitons.

Sélection polynomiale en moyenne caractéristique

Propagation des paramètres discordants. Pour aller au delà de l'amélioration du crible symétrique proposé plus haut en moyenne caractéristique, nous nous appuyons sur la sélection polynomiale par méthode de conjugaison, qui permet d'abaisser les complexités de ce cas de figure par rapport au choix des polynômes par fraction continue dans l'algorithme classique à deux branches. Construire un crible multiple avec cette sélection répercuterait le gain obtenu.

L'inconvénient de la méthode de conjugaison réside dans la création de polynômes aux paramètres déséquilibrés : si l'un présente un petit degré et de grands coefficients, l'autre en revanche expose un grand degré et des coefficients de taille constante. Par conséquent, si nous pouvons bel et bien effectuer des combinaisons linéaires pour construire un diagramme commutatif multi-branches comme précédemment, la nature même de ces combinaisons propagerait les inconvénients de chacun des deux polynômes initiaux sur la totalité des polynômes créés – à savoir, haut degré et gros coefficients. Or nous cherchons à générer facilement des éléments de petites normes ; par la borne donnée sur le résultant nous constatons que plus les degrés des polynômes de définitions des corps de nombres sont bas, plus les normes seront réduites. De même, plus les tailles des coefficients des polynômes de définition sont petites, plus les normes sont basses.

Reconstruction d'un troisième polynôme de définition. Pour contourner cet obstacle, nous nous appuyons sur un troisième polynôme qui partagera les mêmes paramètres que l'un des deux initiaux, et qui présentera la même racine commune dans le corps cible. Ainsi, n'importe quelle combinaison linéaire de ce troisième polynôme et de celui qui lui est similaire, disons f_2, mènera à un autre polynôme aux propriétés identiques, sans propager l'inconvénient provenant de f_1. Plus précisément, nous reprenons les notations du paragraphe 5.1.2, *méthode par conjugaison*, et supposons f_1 fixé. Au moment de reconstruire la racine λ par fraction continue nous remarquons que nous pouvons en réalité écrire :

$$\lambda \equiv \frac{a}{b} \equiv \frac{a'}{b'} \mod p$$

avec a, b, a' et b' de la taille de $\sqrt{p}$. Nous soulignons que ces deux reconstructions (a, b) et (a', b') de λ sont linéairement indépendantes sur $\mathbb{Q}$. Nous posons alors :

$$f_2 = bg_a + ag_b \quad \text{et} \quad f_3 = b'g_a + a'g_b.$$

Le polynôme f_2 est inchangé. En revanche notre nouveau polynômef_3, qui est indépendant de ce dernier, possède deux avantages. Puisque nous avons $f_3 \equiv b'(g_a + \lambda g_b) \mod p$, il partage dans un premier temps la même racine commune que f_1 et f_2 dans $\mathbb{F}_{p^n}$. Par ailleurs, f_3 est tout comme f_2 de degré n et à coefficients de l'ordre de $\sqrt{p}$.

Finalement, pour tout entier $i \in [\![4, V]\!]$ nous posons :

$$f_i = \alpha_i f_2 + \beta_i f_3$$

avec α_i et β_i de la taille de $\sqrt{V}$. Encore une fois, avec les valeurs que nous choisirons, la taille de V sera négligeable par rapport à la taille de la caractéristique p. Par combinaison linéaire nous savons maintenant que pour tout $i \in [\![2,, V]\!]$, f_i est de degré n, a des coefficients de taille $\sqrt{p}$ et se trouve être divisible par $g_a + \lambda g_b$ dans $\mathbb{F}_p[X]$.

Deux bornes de friabilité. Le diagramme étant tracé, la suite de l'algorithme diffère peu du crible symétrique proposé ci-dessus. L'importance majeure provient du rôle particulier joué par le premier corps de nombres, celui défini par f_1. Comme il possède des propriétés différentes des autres corps de nombres, nous traitons son cas à part. Dans le crible, cette idée se traduit par la collecte de polynômes qui sont B-friables dans le premier corps de nombres, pour une certaine borne B, et B'-friables dans n'importe lequel des autres corps de nombres, pour une seconde borne de friabilité.

Si le crible s'effectue encore sur des polynômes de haut degré dans le cas des corps de moyenne caractéristique, un espace de crible constitué uniquement de polynômes de degré 1 suffit dans le cas des corps de grande caractéristique. L'algèbre linéaire et la phase de logarithme discret individuel sont identiques à celles présentées pour le crible symétrique.

Symétrique, dissymétrique, ou intermédiaire

Prenons un peu de recul et comparons à haut niveau les cribles symétriques et dissymétriques avant de pénétrer le détail des calculs de complexité.

A première vue, le crible symétrique offre l'opportunité de choisir n'importe quelle paire de corps de nombres pour obtenir une relation satisfaisante, sans forcer le passage par le premier corps de nombres. De ce fait, la probabilité de double friabilité est meilleure. Plus précisément, si nous avons V corps de nombres, la probabilité de friabilité est multipliée par un facteur V^2 dans le cas du crible symétrique, tandis qu'elle ne s'accroit que d'un facteur V pour le crible dissymétrique. Implicitement nous comparons ici les probabilités de friabilité à un crible classique sur deux corps de nombres uniquement.

Malheureusement, cet avantage que semble prendre le crible symétrique sur son homologue dissymétrique n'est qu'une illusion : il est contrebalancé par l'inconvénient du temps passé à chercher l'une de ces bonnes relations. En effet, pour trouver une relation satisfaisante dans un crible dissymétrique, nous devons tester pour chacun des S^t polynômes de l'espace de crible si celui-ci donne une norme friable dans le premier corps de nombres ; puis, si tel est le cas, nous considérons sa friabilité dans les corps suivants. Le coût des tests dans les V corps de nombres est ici amorti par le fait que seul un petit nombre de polynômes passent le premier test dans le premier corps. Aussi nous pouvons approcher le temps de crible par la taille de l'espace de crible lui-même, c'est-à-dire S^t.

Dans le cas d'un crible symétrique en revanche, nous devons tester, pour chacun des S^t polynômes de l'espace de crible, si sa norme est friable dans le premier corps de nombres, puis, si elle l'est, tester si c'est aussi le cas dans l'un des corps suivants. Cependant, même lorsque la norme n'est pas friable dans le premier corps, il faut tester les corps suivants. Cette fois-ci, le temps de crible est donc multiplié par un facteur V ! Ce qui contrebalance précisément le gain que semblait avoir celui-ci.

Remarque 5.2.1. Aucune conclusion ne semble évidente intuitivement lorsqu'il s'agit de préférer construire un crible symétrique ou un crible dissymétrique. Le déséquilibre des polynômes nous donnera le plus souvent la possibilité d'envisager la seconde option, et non la première ; toutefois, en dehors des calculs asymptotiques concrets, il parait difficile de conclure sur l'optimalité de l'une ou l'autre des deux solutions. Pour aller plus loin, il serait néanmoins intéressant d'envisager un crible intermédiaire aux cribles symétrique et dissymétrique. Si l'on imagine que l'on dispose d'un grand nombres de branches dans notre diagramme, il est ainsi possible de construire un crible qui force chacune des relations à passer par certaines branches, disons b branches particulières. Cette idée serait judicieuse dans le cas d'une sélection polynomiale favorisant b polynômes de définition. Aussi, le crible dissymétrique ne concrétiserait que le cas particulier $b = 1$ d'une telle variante, tandis que le crible symétrique se trouverait à l'opposé, pour le cas $b = V$.

5.3 Analyses de complexité

Nous donnons ici les détails des complexités asymptotiques heuristiques que nous pouvons obtenir par crible dissymétrique, et qui sont toujours actuellement les plus basses pour le cas général. Dans l'ordre, nous traitons au paragraphe 5.3.1 le crible MNFS-Conj (NFS multiple par méthode de conjugaison) en moyenne caractéristique, le crible MNFS-Conj dans le cas frontière

au paragraphe 5.3.2, et le crible multiple par réduction de réseaux en grande caractéristique au paragraphe 5.3.3. Fixons les notations communes à tout ces exemples. Nous réécrivons n le degré d'extension et p la caractéristique du corps cible $\mathbb{F}_Q$ comme étant égaux à :

$$n = \frac{1}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{1-l_p} \quad \text{et} \quad p = \exp(c_p(\log Q)^{l_p}(\log\log Q)^{1-l_p})$$

avec $1/3 < l_p \leqslant 2/3$ pour les moyennes caractéristiques, $l_p = 2/3$ pour le cas frontière, et enfin $2/3 \leqslant l_p$ pour les grandes caractéristiques.

Les paramètres qui interviennent dans l'analyse des complexités asymptotiques sont : la borne de crible S, le degré des polynômes sur lesquels s'effectue le crible $t-1$, le nombre de corps de nombres V, la borne de friabilité B associée au premier corps de nombres et celle B' que l'on relie à tous les corps de nombres qui suivent. L'analyse dans tous les cas de figure cherche à optimiser le temps total de l'exécution de la recherche des relations et de l'algèbre linéaire tout en prenant en compte deux contraintes importantes.

Equilibrer le coût des deux premières étapes

Nous exigeons d'abord que le temps d'exécution de la phase de crible S^t soit égal au coût du déroulement de l'algèbre linéaire. Puisque le système obtenu est creux, le coût d'exécution d'un algorithme d'algèbre linéaire est asymptotiquement en $(B + VB')^2$. En plus de chercher un équilibre entre les deux premières phases de calcul, nous souhaitons équilibrer les deux parties qui jouent un rôle distinct dans la base de friabilité, c'est-à-dire que nous posons :

$$B = VB'.$$

En oubliant la constante égale à 4 qui est évidemment négligeable face aux paramètres que nous considérons, la première contrainte qui consiste à égaliser les deux premières étapes peut s'écrire simplement comme :

$$S^t = B^2. \tag{5.2}$$

Equilibrer le nombre d'équations et le nombre d'inconnues

Pour pouvoir espérer résoudre correctement le système linéaire créé au court de la phase de collection des relations, nous souhaitons obtenir autant de bonnes équations que d'inconnues, c'est-à-dire de l'ordre de B. En notant $\mathcal{P}$ la probabilité qu'un polynôme du crible mène à une relation satisfaisante, cela

signifie que nous demandons l'égalité $S^t\mathcal{P} = B$. En combinant cette dernière exigence avec la contrainte (5.2), il vient :

$$B = 1/\mathcal{P}.$$

5.3.1 MNFS-Conj en moyenne caractéristique

Nous poursuivons l'analyse pour tous les corps finis de moyenne caractéristique, dans le cas d'un crible multiple initié par la méthode de conjugaison.

Evaluation des probabilités de friabilité

Nous rappelons que f_1 est de degré $2n$ et possède des coefficients de taille constante tandis que tous les autres polynômes f_i sont de degré n et montrent des coefficients de taille $\sqrt{p}$. En utilisant la majoration (5.1) sur les résultants et donc sur les normes, nous déduisons que la norme d'un polynôme ϕ du crible est majorée par S^{2n} dans le premier corps de nombre, et par $S^n p^{t/2}$ dans chacun des corps de nombres suivants. En effet, la factorielle de la somme des degrés est négligeable dès lors que la caractéristique est de taille moyenne.

Afin d'évaluer la probabilité de friabilité de ces normes par rapport à B et B', nous les réécrivons sous une forme plus simple, utile à l'application directe du corollaire 5.1.2. Pour cela, nous posons :

$$t = \frac{c_t}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{2/3-l_p}, S^t = L_Q(1/3, c_s c_t), B = L_Q(1/3, c_b) \text{ et } V = L_Q(1/3, c_v).$$

Avec ces notations nous remarquons dans un premier temps que la première contrainte peut s'écrire maintenant sous la forme :

$$c_s c_t = 2c_b. \tag{5.3}$$

Nous pouvons alors appliquer le corollaire 5.1.2 pour reformuler la seconde contrainte. Notons $L_Q(1/3, p_r)$ (respectivement $L_Q(1/3, p_{r'})$) la probabilité d'obtenir une norme B-friable dans le premier corps de nombres (respectivement une norme B'-friable dans au moins l'un des autres corps de nombres). La seconde contrainte devient $c_b = -(p_r + p_{r'})$. A l'aide de l'équation (5.3), les constantes dans les probabilités peuvent ainsi être explicitées :

$$p_r = \frac{-2c_s}{3c_b} = \frac{-2(2/c_t)c_b}{3c_b} \qquad \text{et} \qquad p_{r'} = c_v - \frac{(2/c_t)c_b + c_t/2}{3(c_b - c_v)}.$$

Ceci entraine l'égalité $c_b = -(-4/(3c_t) + c_v - (4c_b + c_t^2)/(6c_t(c_b - c_v)))$ puis d'exiger $6c_t(c_b^2 - c_v^2) = 8(c_b - c_v) + 4c_b + c_t^2$. Finalement nous souhaitons écrire :

$$(6c_t)c_b^2 - 12c_b - 6c_t c_v^2 + 8c_v - c_t^2 = 0. \tag{5.4}$$

Optimisation de la complexité finale

La complexité asymptotique globale de la variante que nous avons présentée est donnée par le coût d'exécution de l'algorithme d'algèbre linéaire creuse sous-jacent en $L_Q(1/3, 2c_b)$. Nous rappelons en effet que nous avons déjà égalisé ce temps de calcul avec celui nécessaire à la collecte des relations. Par conséquent, nous cherchons à minimiser c_b sous la contrainte (5.4) précédemment donnée. La méthode des multiplicateurs de Lagrange indique que c_b, c_v et c_t doivent être des solutions du système suivant :

$$\begin{cases} 2 + \lambda(12c_t c_b - 12) = 0 \\ \lambda(-12c_v c_t + 8) = 0 \\ \lambda(6c_b^2 - 6c_v^2 - 2c_t) = 0 \end{cases}$$

avec $\lambda \in \mathbb{R}^*$. De la deuxième ligne nous pouvons écrire $c_t = 2/(3c_v)$ tandis que la troisième nous permet de noter que $c_b = (c_v^2 + 2/(9c_v))^{1/2}$. Assemblées avec l'égalité (5.4), ceci induit l'équation en une variable : $405c_v^6 + 126c_v^3 - 1 = 0$. Nous en déduisons que $c_v = ((3\sqrt{6} - 7)/45)^{1/3}$ ce qui nous autorise enfin à retrouver $c_b = ((9+4\sqrt{6})/15)^{1/3}$. Finalement, la complexité asymptotique heuristique du crible par corps de nombres reposant sur la sélection polynomiale par méthode de conjugaison est :

$$L_Q\left(\frac{1}{3}, \left(\frac{8 \cdot (9 + 4\sqrt{6})}{15}\right)^{1/3}\right).$$

Elle est à comparer avec la variante classique du crible s'appuyant lui aussi sur la méthode de conjugaison. Proposée dans [BGGM15], celle-ci atteignait une complexité de $L_Q(1/3, (96/9)^{1/3})$. Notre seconde constante peut être approchée par $(8(9 + 4\sqrt{6})/15)^{1/3} \approx 2.156$, tandis que $(96/9)^{1/3} \approx 2.201$.

5.3.2 MNFS-Conj dans le cas frontière $p = L_Q(2/3, c_p)$

L'analyse de cette configuration particulière suit celle que nous venons d'effectuer, à l'exception du fait qu'il nous faut maintenant considérer le paramètre t avec plus de précaution. En effet, nous étudions ici une famille d'algorithmes indicés par le degré $t-1$ des polynômes du crible. Nous cherchons donc à expliciter la complexité totale de chacun de ces algorithmes comme une fonction de c_p (et t). Nous souhaitons en revanche souligner que l'erreur volontaire d'arrondie entre t et $t-1$ dans le calcul des normes ne peut plus être négligeable dans ce cas de figure.

Crible sur des polynômes de degré $t-1$

Pour faciliter l'analyse et notamment l'évaluation des probabilités de friabilité, nous utilisons les paramètres suivants :

$$V = L_Q(1/3, c_v),\ B = L_Q(1/3, c_b),\ B' = L_Q(1/3, c_b - c_v) \text{ et } S = L_Q(1/3, c_s).$$

Avec ces nouvelles notations, la première contrainte devient cette fois-ci :

$$c_s t = 2c_b. \tag{5.5}$$

De plus les normes peuvent être majorées par $S^{2n} = L_Q(2/3, 2c_s/c_p)$ dans le premier corps de nombres et par $S^n p^{(t-1)/2} = L_Q(2/3, c_s/c_p + c_p(t-1)/2)$ dans chacun des suivants. L'application du théorème de Canfield-Erdős-Pomerance et l'utilisation des mêmes notations qu'au paragraphe précédent entraine les égalités $p_r = -2c_s/(3c_b c_p)$ d'une part et $p_{r'} = c_v - (c_s/c_p + c_p(t-1)/2)/(3(c_b - c_v))$ d'autre part. En utilisant l'équation (5.5), la seconde contrainte $c_b = -(p_r + p_{r'})$ se voit reformulée comme $3tc_p(c_b - c_v)(c_b + c_v) = 4(c_b - c_v) + 2c_b + t(t-1)c_p^2/2$. Aussi, nous souhaitons avoir :

$$(6tc_p)c_b^2 - 12c_b - 6tc_p c_v^2 + 8c_v - t(t-1)c_p^2 = 0. \tag{5.6}$$

Sans surprise, nous cherchons encore à minimiser $2c_b$ sous la contrainte (5.6). La méthode de Lagrange montre alors qu'il faut que la dérivée de $(6tc_p)c_b^2 - 12c_b - 6tc_p c_v^2 + 8c_v - t(t-1)c_p^2$ par rapport à la variable c_v s'annule. Ceci explique pourquoi nous voulons égaliser $c_v = 2/(3tc_p)$. En injectant cette dernière valeur à l'intérieur de l'équation (5.6) il découle :

$$(18t^2c_p^2)c_b^2 - (36tc_p)c_b + 8 - 3t^2(t-1)c_p^3 = 0.$$

Finalement, après résolution de cette dernière équation en c_b nous déduisons l'égalité $c_b = (6 + (20 + 6t^2(t-1)c_p^3)^{1/2})/(6tc_p)$. La complexité asymptotique totale du crible multiple par corps de nombres avec la méthode de sélection polynomiale par conjugaison est, dans ce cas frontière, de :

$$L_Q\left(\frac{1}{3}, \frac{2}{c_p t} + \sqrt{\frac{20}{9(c_p t)^2} + \frac{2}{3}c_p(t-1)}\right)$$

où $t-1$ représente le degré des polynômes sur lesquels s'oppère le crible. La figure 5.6 compare ce MNFS-Conj avec d'autres variantes ayant cours au même endroit. Pour la plupart des variantes de NFS présentées dans cette figure, chaque creux de courbe correspond à un degré particulier des polynômes du crible.

5.3.3 MNFS-JLSV en grande caractéristique

Le crible multiple a souffert dans cette configuration d'un souci de chronologie, puisque rien n'empêche l'analyse de Matyukhin de 2003 de s'appliquer aux corps de grande caractéristique, excepté qu'aucune sélection polynomiale n'existait pour ces paramètres. La première a autorisé l'extension de NFS (et donc de MNFS) à ces corps n'a été publiée que trois ans plus tard [JLSV06]. Ceci explique pourquoi l'analyse que nous traitons ici n'est que très succincte, le but étant simplement de rattacher la sélection polynomiale de Joux, Lercier, Smart, et Vercauteren au résultat de Matyukhin.

Notons θ_i une racine complexe de f_i pour tout i. Nous criblons sur les triplets (a, b, i) avec $|a|, |b|$ bornés par S et $i \in [\![1,, V]\!]$ tels que $\text{pgcd}(a, b) = 1$, $\text{N}(a - b\theta_1)$ est B-friable et $\text{N}(a - b\theta_i)$ est B'-friable.

Pour tout $i \in [\![1, V]\!]$, si a et b sont plus petits que S nous avons les deux bornes : $\text{N}(a - b\theta_1) = |b^{\deg f_1} f_1(a/b)| \leq (n+1)\|f_1\|_\infty S^n$ et $\text{N}(a - b\theta_i) = |b^{\deg f_i} f_i(a/b)| \leq (d+1)\|f_i\|_\infty S^d$.

Comme précédemment nous posons $n = c_p^{-1}(\log Q/(\log\log Q))^{1-l_p}$, puis de la même manière $d = \delta(\log Q/(\log\log Q))^{1/3}$ et $S = L_Q(1/3, c_s c_p)$. Ainsi $S^n = L_Q(l)$, avec $l \leqslant 2/3$. Pour $i \in [\![1, V]\!]$ ceci mène à :

$$|\text{N}(a - b\theta_1)| \leq (Q^{1/(d+1)})^{1+o(1)}$$

et :

$$|\text{N}(a - b\theta_i)| \leq (Q^{1/(d+1)} S^d)^{1+o(1)}.$$

Ces deux majorations sont indépendantes de n et coincident précisément avec les conditions (34) et (35) de l'article de Matyukhin. Une transcription immédiate (depuis le russe...) des calculs effectués nous permet d'obtenir les paramètres :

$$\begin{aligned}
&B = S = L_Q(1/3, ((46 + 13\sqrt{13})/108)^{1/3}),\\
&B' = \exp(\log B(\sqrt{13} - 1)/3),\\
&V = L_Q(1/3, 1 - ((\sqrt{13} - 1)/3)^{1/3}),\\
&d = ((46 + 13\sqrt{13})(4\sqrt{13} - 10)^3 (2^2 3^6)^{-1} \log Q(\log\log Q)^{-1})^{1/3}.
\end{aligned}$$

La complexité asymptotique du crible par corps de nombres multiples avec méthode de réduction de réseau en grande caractéristique est ainsi de :

$$L_Q\left(\frac{1}{3}, \left(\frac{92 + 26\sqrt{13}}{27}\right)^{1/3}\right),$$

où $((92 + 26\sqrt{13})/27)^{1/3} \approx 1.902$. Ce résultat est à comparer avec la complexité des corps de grande caractéristique obtenue par le crible classique,

qui a une complexité en $L_Q(1/3, (64/9)^{1/3})$, avec $(64/9)^{1/3} \approx 1.923$. Notons que la seconde constante barbare que nous obtenons au final est identique à celle donnée par MNFS-GJL dans le cas frontière, et MNFS pour la factorisation.

Conclusion à propos des complexités actuelles

Le résumé des complexités actuelles obtenues pour les corps finis généraux est illustré par la figure 5.5. La figure 5.6 reprend quant à elle les différentes courbes de complexités du cas frontière entre les corps de moyenne caractéristique et ceux de grande caractéristique, pour lequel nous avons effectués plusieurs analyses. L'algorithme $\mathcal{A}$ dont nous n'avons pas donné de détail est celui présenté par Sarkar et Singh dans [SS16c], qui propose une manière de généraliser à la fois les sélections polynomiales par conjugaison, et de Joux-Lercier généralisée.

L'intuition de l'importance – ou non – d'un changement dans la seconde constante de la complexité n'est cependant pas aisée. A des fins cryptographiques il est intéressant de comprendre comment cet abaissement de la difficulté du problème se traduit en terme de sécurité, par exemple lors du passage d'une complexité approximative en $L_Q(1/3, 2.201)$ dans le cas de la moyenne caractéristique à une complexité approximative de $L_Q(1/3, 2.156)$. Imaginons que nous communiquons grâce à un protocole qui repose sur le problème du logarithme discret dans l'un de ces corps de cardinalité Q, et que ce protocole nous assure un certain niveau de sécurité, disons s. C'est-à-dire que :

$$s = L_Q(1/3, 2.201).$$

La question soulevée est donc de trouver pour quelle nouvelle cardinalité Q' aurions-nous le même niveau de sécurité, connaissant les progrès effectués sur le problème sous-jacent. C'est-à-dire, encore une fois, pour quel Q' a-t-on :

$$s = L'_Q(1/3, 2.156)\,?$$

L'égalité $L'_Q(1/3, 2.156) = L_Q(1/3, 2.201)$ devient alors :

$$2.156^3 \log Q'(\log\log Q')^2 = 2.201^3 \log Q(\log\log Q)^2.$$

En approximant $\log Q'(\log\log Q')^2 \approx 1.064 \log Q(\log\log Q)^2$ nous obtenons :

$$\log Q' \approx 1.064 \log Q.$$

Cette dernière approximation signifie que nous devons augmenter la taille en bit du corps fini sur lequel est fondé notre protocole de 6.4% pour retrouver le même niveau de sécurité.

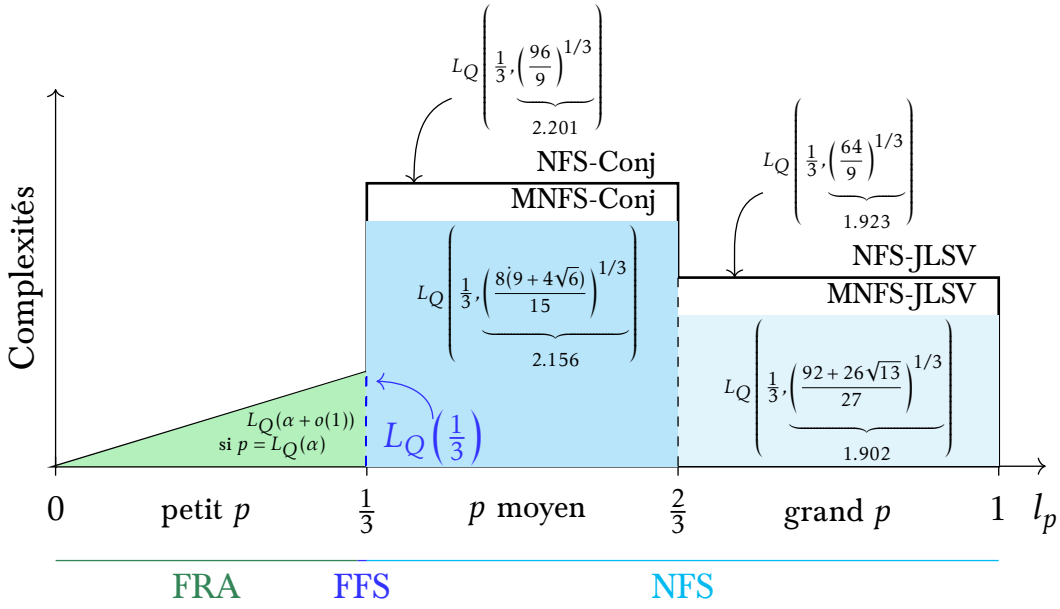


Figure 5.5 – *Algorithmes de calcul de logarithme discret dans les corps finis généraux, et leurs complexités.* Pour un corps fini de cardinalité $Q = p^n$ fixée, ce dessin montre comment les complexités asymptotiques varient en fonction de l_p, avec $p = L_{p^n}(l_p)$.

5.4 D'autres petites analyses pour le plaisir

5.4.1 Un crible symétrique obsolète en moyenne caractéristique

Nous attaquons ici $\mathbb{F}_{p^n}$ un corps fini général pour lequel la caractéristique est telle que $p = L_Q(l_p, c_p)$ avec $1/3 < l_p < 2/3$. En appliquant un crible symétrique avec la méthode par fraction continue, tous les V polynômes de définition des corps de nombres ont degré n et des coefficients de taille $O(\sqrt{p})$. Nous supposons encore que nous pouvons écrire les quatre paramètres V, B, S et t tels que :

$$V = L_Q(1/3, c_v),\ B = L_Q(1/3, c_b),\ S = L_Q(l_p - 1/3, c_s c_p),\ t = \frac{c_t}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{2/3-l_p}.$$

où c_v, c_b, c_s et c_t sont à déterminer plus tard. Le coût total du crible – respectivement la taille de la base de friabilité – est $VS^t = L_Q(1/3, c_v + c_s c_t)$ – respectivement $VB = L_Q(1/3, c_v + c_b)$.

Nous donnons l'analyse de complexité sans rentrer dans les détails, comme celle-ci suit point par point les exigences soulevées dans les analyses précédentes. L'équilibre du crible et de l'algèbre linéaire donne $VS^t = (VB)^2$ et par

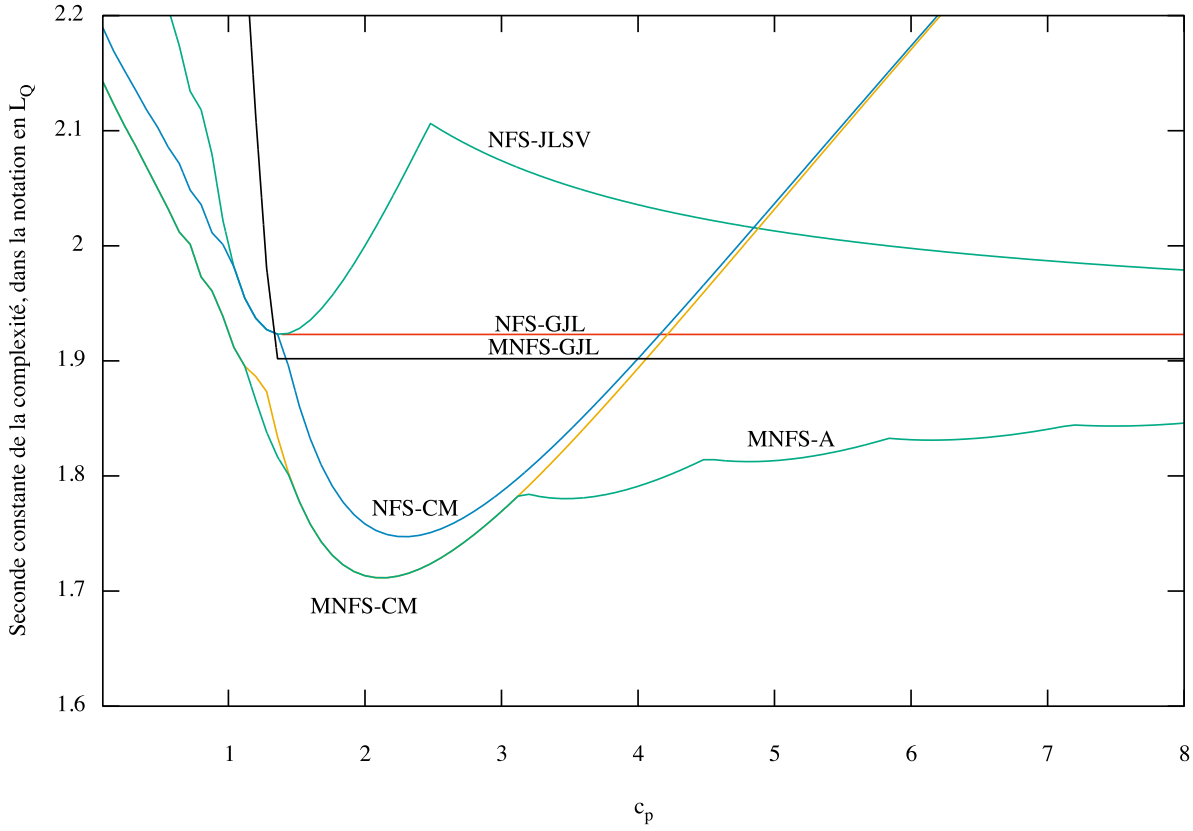


Figure 5.6 – *Complexités asymptotiques* $L_Q(1/3, C(c_p))$ *dans le cas frontière, comme une fonction de* c_p *lorsque* $p = L_Q(2/3, c_p)$. La courbe rouge représente la complexité obtenue par NFS avec la sélection polynomiale de Joux-Lercier généralisée [BGGM15], et la courbe noire sa version multiple que nous venons de détailler. La courbe bleue la complexité de NFS avec la méthode de conjugaison [BGGM15], et la courbe jaune sa variante multiple, aussi présente dans ce chapitre. A titre de comparaison nous avons gardé la courbe verte (la plus haute) de NFS-JLSV [JLSV06] qui faisait référence jusqu'en 2014. Enfin, la courbe verte la plus basse correspond à l'algorithme MNFS-$\mathcal{A}$ de [SS16c]. Une partie de cette courbe se confond avec la courbe de MNFS-CM.

conséquent :

$$2c_b + c_v = c_s c_t. \tag{5.7}$$

Nous demandons encore $S^t\mathcal{P} = VB$, c'est-à-dire :

$$B \approx 1/\mathcal{P} \tag{5.8}$$

Mais comment évaluer $\mathcal{P}$? Si $\mathcal{P}'$ désigne la probabilité de friabilité d'un

élément arbitraire dans un corps de nombres alors :

$$\mathcal{P} = \frac{V(V-1)}{2} \cdot \mathcal{P}'^2. \tag{5.9}$$

Sous les hypothèses heuristiques habituelles $\mathcal{P}'$ est égale à la probabilité que $S^n p^{t/2}$ soit B-friable. Comme $S^n p^{t/2} = L_Q(2/3, c_s + c_t/2)$, on en déduit $\mathcal{P}' = L_Q(1/3, -(c_s + c_t/2)/(3c_b)$ en applicant un corollaire du théorème de Canfield, Erdös et Pomerance. Grâce à (5.9) il vient :

$$2\mathcal{P} = L_Q\left(\frac{1}{3}, 2c_v - \frac{2c_s + c_t}{3c_b}\right).$$

L'équation (5.8) signifie que nous souhaitons $1/B = \mathcal{P} = (1/2) \cdot L_Q(1/3, 2c_v - (2c_s + c_t)/3c_b)$. Puisque $1/2$ est négligeable cela entraine $3c_b^2 = 2c_s + c_t - 6c_v c_b$. La seconde condition est donc :

$$6c_v c_b + 3c_b^2 = 2c_s + c_t \tag{5.10}$$

Rappelons que la complexité finale est en $L_Q(1/3, 2(c_b + c_v))$. Grâce à l'équation (5.7) cela signifie que nous voulons minimiser $2(c_b + c_v)$ sous la contrainte :

$$3c_b^2 c_s + 6c_v c_s c_b - 2c_s^2 - 2c_b - c_v = 0. \tag{5.11}$$

Par Lagrange on considère donc le système :

$$\begin{cases} 2 + \lambda(6c_b c_s + 6c_v c_s - 2) = 0 \\ 2 + \lambda(6c_b c_s - 1) = 0 \\ \lambda(3c_b^2 + 6c_v c_b - 4c_s) = 0 \end{cases}$$

où $\lambda \in \mathbb{R}^*$. Des deux premières lignes il vient $c_v = (6c_s)^{-1}$ et de la dernière nous écrivons $3c_b^2 c_s^2 + c_b c_s - 4c_s^3 = 0$. Mis en commun avec l'équation (5.11) nous déduisons que $c_b = (12c_s^3 - 1)/(12c_s)$. En traduisant la contrainte (5.11) avec ces deux valeurs, nous souhaitons que le paramètre c_s vérifie $48c_s^6 - 56c_s^3 - 1 = 0$. Donc $c_s = ((7 + 2\sqrt{13})/12)^{1/3}$, $c_v = ((2\sqrt{13} - 7)/54)^{1/3}$ et $c_b = ((16 + 4\sqrt{13})/27)^{1/3}$. Finalement, la complexité asymptotique du crible par corps de nombres multiples est :

$$L_Q\left(\frac{1}{3}, \left(\frac{4(46 + 13\sqrt{13})}{27}\right)^{1/3}\right).$$

A titre de comparaison, on peut approcher $\left((4(46 + 13\sqrt{13}))/27\right)^{1/3} \approx 2.396$ tandis que $(128/9)^{1/3} \approx 2.42$, qui était la valeur de NFS classique en moyenne

caractéristique jusqu'à l'apparition du crible multiple. Maintenant cette complexité issue de [BP14] est dépassée par un crible dissymétrique reposant sur la méthode de conjugaison [Pie15] et dont la seconde constante dans la complexité est $(8(9+4\sqrt{6})/15)^{1/3} \approx 2.156$, comme étudié précédemment.

Nous ne donnons pas les calculs fastidieux de l'analyse de complexité pour le crible symétrique dans le cas frontière, mais le lecteur avide de s'entrainer peut les trouver dans l'appendice de l'article [BP14].

5.4.2 MNFS-GJL dans le cas frontière, obsolète

Le crible multiple par réduction de réseau qui suit la sélection polynomiale de Joux-Lercier généralisée emprunte au crible par réseau de 2006 le fait de cribler uniquement sur des polynômes de degré 1. Comme précédemment, nous supposons que $B = VB'$. La contrainte de l'équation (5.2) se traduit par l'exigence d'égalité entre la borne de crible S et la borne de friabilité B. Avec les mêmes notations que celles employées dans les paragraphes précédents, nous souhaitons ainsi que l'égalité $B = 1/\mathcal{P}$ soit satisfaite. Rappelons enfin que la sélection polynomiale considérée demande $n < d+1$. Si nous posons maintenant :

$$d = \delta\left(\frac{\log Q}{\log\log Q}\right)^{1/3},$$

où δ est un paramètre à définir, alors nous devrons nous souvenir que nos résultats de complexité seront pertinents dès lors que $\delta \geqslant 1/c_p$.

Puisque f_1 a des petits coefficients et degré $d+1$, les normes dans le premier corps de nombres sont majorées par $L_Q(2/3, c_b\delta)$. La probabilité d'obtenir une norme B-friable est alors $L_Q(1/3, p_r)$ avec $p_r = -\delta/3$. De la même façon, les normes dans les $V-1$ corps de nombres suivants sont majorées par $L_Q(2/3, c_b\delta + 1/\delta)$. La probabilité d'obtenir une norme B'-friable dans au moins l'un de ces corps de nombres est $L_Q(1/3, p_{r'})$ avec $p_{r'} = -(c_b\delta + 1/\delta)/(c_b - c_v) + c_v$.

En supposant les événements indépendants, de l'égalité $c_b = -(p_r + p_{r'})$ nous tirons :

$$\begin{aligned} & c_b + c_v = \frac{\delta}{3} + \frac{\delta^2 c_b + 1}{3\delta(c_b - c_v)} \\ \Leftrightarrow \quad & 3\delta(c_b^2 - c_v^2) = 2\delta^2 c_b - \delta^2 c_v + 1 \\ \Leftrightarrow \quad & 3\delta c_b^2 - 2\delta^2 c_b + \delta^2 c_v - 3\delta c_v^2 - 1 = 0. \end{aligned}$$

La méthode des multiplicateurs de Lagrange indique encore une fois que nous

cherchons des solutions au système :

$$\begin{cases} 3\delta c_b^2 - 2\delta^2 c_b + \delta^2 c_v - 3\delta c_v^2 - 1 = 0 \\ 3c_b^2 - 4\delta c_b + 2\delta c_v - 3c_v^2 = 0 \\ \delta^2 - 6\delta c_v = 0 \end{cases} \tag{5.12}$$

De la troisième ligne du système (5.12) nous retrouvons $\delta = 6c_v$. En substituant à l'intérieur de la deuxième ligne nous déduisons que $c_b^2 - 8c_v c_b + 3c_v^2 = 0$. Ainsi, en écrivant c_v comme une fonction de c_b il vient : $c_v = ((4-\sqrt{13})/3)c_b$. En remplaçant δ par sa valeur dans la première ligne du système précédent nous obtenons alors $18c_v c_b^2 - 72c_v^2 c_b + 18c_v^3 - 1 = 0$, puis, en substituant de nouveau c_v par sa valeur nous pouvons conclure que $c_b = (46 + 13\sqrt{13}/108)^{1/3}$. Tout lecteur attentif, s'il me fait parvenir par mail la valeur de cette constante $c_b = (46+13\sqrt{13}/108)^{1/3}$, gagnera la tablette de chocolat de son choix. Grâce à cette constante nous pouvons calculer la valeur de δ qui est $(4\sqrt{13}-14)^{1/3}$. Par conséquent, dès lors que :

$$c_p \geqslant \left(\frac{7+2\sqrt{13}}{6}\right)^{1/3},$$

qui est environ égal à 1.33, la complexité du crible multiple par corps de nombres avec la méthode de Joux-Lercier généralisée est en :

$$L_Q\left(\frac{1}{3}, \left(\frac{2\cdot(46+13\sqrt{13})}{27}\right)^{1/3}\right).$$

Nous trouvons exactement la même complexité que pour le cas du crible multiple par réduction de réseau en grande caractéristique, ce qui n'est pas surprenant, le crible classique par GJL [BGGM15] ayant lui aussi une complexité en $L_Q(1/3, (64/9)^{1/3})$. Pour faciliter les comparaisons nous rappelons que $(64/9)^{1/3} \approx 1.92$ tandis que $(2(46+13\sqrt{13})/27)^{1/3} \approx 1.90$.

Lorsque $c_p < \left((7+2\sqrt{13})/6\right)^{1/3}$, de la contrainte $\delta > 1/c_p$ nous obtenons $\delta > (4\sqrt{13}-14)^{1/3}$ et la simplification précédente ne s'applique plus. Cependant la double égalité $c_b = 3c_v/(4-\sqrt{13}) = \delta/(2(4-\sqrt{13}))$ indique que nous minimisons la complexité lorsque $\delta = 1/c_p$. Par conséquent nous pouvons écrire $c_b = (4+\sqrt{13})/(6c_p)$. Enfin, dès lors que :

$$c_p < \left(\frac{7+2\sqrt{13}}{6}\right)^{1/3},$$

la complexité asymptotique de MNFS avec cette méthode est :

$$L_Q\left(\frac{1}{3}, \frac{4+\sqrt{13}}{3c_p}\right).$$

La figure 5.6 trace les variantes de complexité en fonction de c_p.

5.4.3 Une phase de descente négligeable

L'étape de logarithme discret individuel est considéré négligeable dans toutes les analyses précédentes du crible par corps de nombres, en théorie, mais aussi en pratique. Dans notre configuration multiple, une approche naïve entraine pourtant une complexité asymptotique en $L_Q(1/3, c)$ avec une constante c qui domine celle des phases préliminaires. Nous proposons donc une modification qui permet de contourner ce problème. L'analyse de la complexité asymptotique suivante permet de conclure que cette étape est bel et bien négligeable. Signalons qu'il existe une méthode plus récente – et plus performante – pour effectuer cette descente dans l'article [Gui15].

Imaginons plutôt que nous commençons l'algorithme par la phase de descente. Ici, nous utilisons $W = L_Q(1/3, c_w)$, corps de nombres (avec $c_w > 0$) sous la forme $\mathbb{Q}(\theta_i)$ construits comme précédemment. La valeur de W est éventuellement plus grande que celle de V utilisée pour le calcul des logarithmes de la base. Soit $h \in \mathbb{F}_Q$ l'élément dont nous cherchons le logarithme discret. Comme dans un NFS classique, la descente se décompose en deux étapes. Nous cherchons tout d'abord à écrire le logarithme de h comme combinaison linéaire de logarithmes d'idéaux $\mathfrak{q}$ de degré 1 dans $\mathbb{Q}(\theta_1)$ de norme inférieure à une constante $C > B$ rendue explicite plus loin. Ensuite, nous effectuons une descente par special-$\mathfrak{q}$ pour exprimer les logarithmes de chacun de ces idéaux $\mathfrak{q}$ comme une combinaison linéaire d'idéaux premiers de degré 1 dans n'importe lequel des W corps de nombres, tant que les normes sont inférieures à B. Nous vérifierons que seul $\exp((\log\log Q)^2)$ idéaux premiers apparaissent dans ce procédé. Aussi, ils appartiennent à un ensemble E de corps de nombres dont l'ordre est négligeable comparé à la valeur de V. Ceci peut nous permettre d'étendre la cardinalité de E à V. Supposons maintenant que la phase de collecte des relations et d'algèbre linéaire vient d'être effectuée sur cet ensemble E de corps de nombres. Nous obtenons alors le logarithme discret des éléments de la base de friabilité, en particulier les logarithmes de ceux qui appartiennent aux corps de nombres de E. En remontant dans l'écriture des relations, nous pouvons ainsi retrouver le logarithme discret de h. L'analyse précise est la suivante.

Descente par fraction continue

Soit $h \in \mathbb{F}_{p^n}$ l'élément dont on cherche le logarithme. Nous rappelons qu'à chaque $s = \sum_{j=0}^{n-1} s_j m^j \in \mathbb{F}_{p^n} = \mathbb{F}_p(m)$ on associe $\bar{s} = \sum_{j=0}^{n-1} s_j \theta_1^j \in \mathbb{Q}(\theta_1)$. Nous cherchons un entier $e \in [\![0, Q-1]\!]$ tel que $h' = h^e$ donne $\bar{h'}$ sans carré et C-friable, où C est une constante telle que $C = L_Q(2/3, c)$ pour une constante c à choisir plus tard. La norme de $\bar{h'}$ est majorée par $(2n)! p^n \sqrt{p}^n = Q^{3/2+o(1)} = L_Q(1, 3/2)$. La probabilité de friabilité de chaque norme est alors $1/L_Q(1/3, 1/(2c))$. Le temps d'un test de C-friabilité par la méthode ECM est en $L_C(1/2, \sqrt{2})(\log Q)^{O(1)}$, par conséquent chaque teste demande un temps de $L_Q(1/3, 2\sqrt{c/3})$. Ceci est optimal pour $c = (3/4)^{1/3}$ ce qui donne une complexité de :

$$L_Q\left(\frac{1}{3}, \left(\frac{9}{2}\right)^{1/3}\right)$$

pour cette première sous étape. Puisque $(9/2)^{1/3} \approx 1.65$ est inférieure à nos complexités, cette partie de la descente est clairement négligeable.

Descente par special-$\mathfrak{q}$

Soit $\mathfrak{q} = \langle q, \theta_i - \rho \rangle$ un idéal premier de degré 1 dans l'un des corps de nombres $\mathbb{Q}(\theta_i)$. Pour simplifier, supposons que $i = 1$. Pour un paramètre D, nous considérons le réseau de degré $D-1$ des polynômes ϕ tel que $\phi(\theta_1)$ est divisible par $\mathfrak{q}$. Une application de l'algorithme LLL nous donne d polynômes ϕ_1, …, ϕ_D de degré $D-1$ et à coefficients de taille $q^{1/D}$ tels que $\phi_1(\theta_1), \ldots, \phi_D(\theta_1)$ est divisible par $\mathfrak{q}$. Ensuite, nous parcourons les polynômes $\phi = \sum_{j=1}^{D} \alpha_j \phi_j$ avec $\alpha_j \in [\![-A, A]\!]$, pour un paramètre A et nous testons avec ECM si les deux conditions si dessous sont simultanément satisfaites :

— la norme de $\phi(\theta_1)$ s'écrit comme q fois un nombre $q^{0.99}$-friable et sans carré,

— pour au moins l'un des indices $i \in [\![1, W]\!]$, la norme de $\phi(\theta_i)$ est $q^{0.99}$-friable et sans carré.

Notons que la constante 0.99 peut être remplacée par n'importe quelle autre valeur de l'intervalle $[0, 1]$. Nous quittons le parcours lorsque l'un des polynômes ϕ est friable, ce qui permet alors d'exprimer $\log \mathfrak{q}$ comme la somme de logarithmes d'idéaux plus petits. Nous poursuivons récursivement le même procédé avec les nouveaux idéaux introduits par $\phi(\theta_i)$ jusqu'à ce que la valeur courante de $q^{0.99}$ soit inférieure à B.

La taille en bits $\log q$ des idéaux $\mathfrak{q}$ introduits est multipliée par 0.99 à chaque étape, aussi l'arbre de descente a une hauteur de $\log_{0.99}(\log B)/(\log C)$ que l'on peut aussi écrire $O(1)\log_{1/2}(\log 2)/(\log Q) = O(\log \log Q)$. Chaque

étape introduit par ailleurs au pire $2\log \mathrm{Res}(\phi, f_1) \leq 2\log Q$ nouveaux idéaux, ce qui nous permet de conclure que la largeur de l'arbre est au plus $O(\log Q)$. Au total, il y a :

$$\exp\left(O(1)(\log\log Q)^2\right)$$

noeuds dans la descente. Ainsi le procédé récursif complet prend un temps de $T^{1+o(1)}$, où T représente le temps de chaque petite étape de descente. Afin d'évaluer maintenant T, nous posons :

$$D = \frac{c_D}{c_p}\left(\frac{\log Q}{\log\log Q}\right)^{2/3-l_p} \text{ et } A = \text{ constante.}$$

Ceci nous donne donc des normes de taille $L_Q(2/3)$. Nous testons par ailleurs $L_Q(1/3)$ polynômes dans chaque étape de descente, comme pour la variante classique de NFS. Soit $\mathfrak{q} = \langle q, \theta_1' - \rho\rangle$ un idéal premier avec $B < q < C$, que l'on cherche à descendre. Grâce au théorème habituel, la probabilité qu'un polynôme ϕ satisfasse les deux conditions précédentes est alors $Wu^{-u+o(u)}$, où $u = 2\log \mathrm{Res}(f,\phi)(0.99\log q)^{-1}$. La plus grande valeur obtenue l'est à la fin de la descente, c'est-à-dire lorsque $q^{0.99} = B$. Dit autrement, c'est la dernière étape qui domine cette descente en spécial-$\mathfrak{q}$. La probabilité de succès est alors $L_Q(1/3, (2/c_D + c_D/c_b + 2c_a/(c_D c_b))/3 - c_w)^{-1}$.

Le nombre de polynômes à tester est $O(1)$. Chaque polynôme est testé sur W corps, donc le temps de descente est $W^{1+o(1)} = L_Q(1/3, c_w)$. Pour minimiser ce temps, nous imposons que $3c_w = 2/c_D + c_D/c_b$. En fixant par ailleurs $c_D = \sqrt{2c_b}$ il vient $c_w = 2\sqrt{2}/(3\sqrt{c_b}) = 2^{5/6}3^{-1/2} \approx 1.03$. Ceci entraine la complexité finale de :

$$L_Q(1/3, (9/2)^{1/3}),$$

pour la phase de logarithme discret individuel, qui est donc bel et bien négligeable par rapport aux deux autres phases.

5.5 Un exemple jouet avec un crible symétrique

Nous proposons dans cette dernière partie de chapitre un exemple jouet pour mieux comprendre les différentes étapes du crible multiple. Cet exemple peut être exécuté avec Sage. Comme nous n'avons pas évoqué les applications de Schirokauer, nous bornons notre exemple au cas du corps $\mathbb{F}_{p^n}$ avec $p = 103$ et $n = 3$. Nous calculons ici un logarithme modulo le facteur premier 3571 de $p^n - 1$. Dans le crible et l'algèbre linéaire nous utilisons $V = 11$ corps de nombres.

5.5.1 Sélection polynomiale

Nous utilisons $a = \lceil\sqrt{103}\rceil = 11$, $g_1 = x^3 + 11$ et $g_2 = ax^3 + (a^2 - 103) = 11x^3 + 18$. Nous construisons ensuite les autres polynômes par combinaisons linéaires de g_1 et g_2 avec les plus petits coefficients en valeur absolue : $f_1 = -9x^3 + 4$, $f_2 = -10x^3 - 7$, $f_3 = x^3 + 11$, $f_4 = -8x^3 + 15$, $f_5 = -18x^3 + 8$, $f_6 = 11x^3+18$, $f_7 = -19x^3-3$, $f_8 = -17x^3+19$, $f_9 = -20x^3-14$, $f_{10} = 2x^3+22$ et $f_{11} = -21x^3 - 25$. Tous les polynômes sont divisble par $x^3 + 11$ modulo 103. Aussi nous représentons le corps cible par $\mathbb{F}_{103^3} = \mathbb{F}_{103}[x]/\langle x^3 + 11\rangle = \mathbb{F}_{103}(m)$, où m est une racine de $x^3 + 11$ dans le corps $\mathbb{F}_{103^3}$.

5.5.2 Construction de la base de friabilité

Pour tout polynôme f_i, K_i est le corps de nombres associé, h_i le nombre de classes de K_i, $\mathcal{O}_i$ l'anneau des entiers, θ_i une racine de f_i dans K_i et $\theta_i' = l(f_i)\theta_i$, avec $l(f_i)$ le coefficient de tête de f_i. Pour chaque corps K_i nous calculons un système d'unités fondamentales. Nous fixons la borne de friabilité à $B = 300$. Nous calculons ensuite la base de friabilité en ajoutant, pour chaque polynôme f_i, les idéaux premiers de degré 1 et de norme inférieure à B, en plus des idéaux premiers de degré arbitraire qui divisent l'indice $[\mathcal{O}_i : \mathbb{Z}[\theta_i']]$. Par exemple, pour $f_4 = -8x^3 + 15$ nous ajoutons 59 idéaux premiers, y compris l'idéal de degré 2 donné par $I = \langle 2, \theta_4'^2/16 + \theta_4'/4 + 1\rangle$. La base de friabilité est alors d'ordre 697. Pour chaque idéal premier $\mathfrak{q}$ de la base de friabilité, nous choisissons un générateur arbitraire $\gamma_\mathfrak{q}$ de $\mathfrak{q}^{h_i}$, où h_i est le nombre de classe du corps contenant l'idéal $\mathfrak{q}$.

5.5.3 Collecte des relations

Nous fixons $t = 3$. Pour le crible, nous considérons tous les polynômes $\phi \in \mathbb{Z}[x]$ de degré 2 à coefficients bornés en valeur absolue par $S = 50$.[2] Nous gardons ensuite les polynômes ϕ qui sont B-friable pour au moins deux corps de nombres. Nous obtenons ainsi un fichier dont une ligne par exemple est de la forme :

$$3x^2 + 5x + 3 : 4, 9, 10, 11 \tag{5.13}$$

qui indique que $\mathrm{Res}(\phi, f_i)$ est B-friable pour $\phi = 3x^2+5x+3$ et $i = 4, 9, 10, 11$.

2. Sans effet sur la complexité nous utilisons aussi les polynômes ϕ de degré 1 et de coefficients bornés par $S' = S\|f_1\|_\infty^{1/n} \approx 136$, qui permet d'avoir des nombre algébriques $\phi(\theta_i)$ de même norme environ.

5.5.4 Construction de la matrice

En considérant une ligne comme celle donnée par (5.13), chaque paire d'indices permet de dériver une équation entre les logarithmes, mais une paire (i_1, i_3) ne donnera pas plus d'information que les paires (i_1, i_2) et (i_2, i_3) prises ensemble. Ici, nous n'ajoutons que trois équations à la matrice, celle provenant de $(4, 9)$, $(4, 10)$ et $(4, 11)$. Avec 2239 polynômes ϕ, nous obtenons une matrice qui possède 3302 lignes. Pour écrire la ligne d'une paire $(4, 9)$, nous factorisons $\phi(\theta_4)\mathcal{O}_4$ et $\phi(\theta_9)\mathcal{O}_9$. Ensuite $\phi(\theta_4) = u_{\phi,4}\prod_{\mathfrak{q}} \gamma_{\mathfrak{q}}^{\mathrm{val}(\phi(\theta_4),\mathfrak{q})}$ avec $u_{\phi,4}$ qui est une unité. Nous écrivons alors $u_{\phi,4}$ comme un produit de puissance d'unités fondamentales. Nous procédons de la même manière pour $\phi(\theta_9)$ afin de trouver l'unité $u_{\phi,9}$. Une ligne représente alors les valuations $\mathrm{val}(\phi(\theta_4), \mathfrak{q})$, les exposants de $u_{\phi,4}$, les valuations de $\mathrm{val}(\phi(\theta_9))$ de signe opposé, et les exposants de $u_{\phi,9}$ de signe opposé. Les 708 colonnes de la matrices sont indicées par les idéaux premiers dans K_1, puis les unités fondamentales dans K_1, et ainsi de suite de K_2 à K_{11}.

5.5.5 Algèbre linéaire

En utilisant directement Sage pour obtenir le noyau de la matrice, nous sommes en présence d'un espace vectoriel de dimension 9. Bien qu'en théorie nous cherchons un espace de dimension 1, ce premier noyau surdimensionné n'est pas surprenant puisqu'il répond à un résultat courant pour les variantes classiques de NFS : ces vecteurs supplémentaires artefacts dans le noyau proviennent en effet des idéaux (sept dans notre cas) qui n'apparaissent pas dans les relations. Nous effaçons donc les colonnes correspondantes, qui sont vides, et calculons de nouveau un noyau de la matrice. Cette fois-ci, sans surprise, l'espace vectoriel est de dimension 2. Il contient un vecteur qui correspond aux logarithmes, et un second vecteur parasite qui vient de l'idéal I de degré 2 présenté plus haut. Par exemple, lorsque nous posons $\mathfrak{q}_3 = \langle 3, \theta_1'/3\rangle$, $\mathfrak{q}_5 = \langle 5, \theta_1'/3 - 2\rangle$ et $\mathfrak{q}_{11} = \langle 11, \theta_1'/3 + 1\rangle$, nous connaissons maintenant les logarithmes $\log \mathfrak{q}_3 \equiv 1 \mod 3751$, $\log \mathfrak{q}_5 \equiv 681 \mod 3751$ et $\log \mathfrak{q}_{11} \equiv 160 \mod 3751$.

Vérification des logarithmes de la base de friabilité

L'utilisation explicite des unités à la place des applications de Schirokauer nous autorise à vérifier les solutions données par l'algèbre linéaire. Considérons par exemple le générateur $g = m + 4$ de $\mathbb{F}_{103^3}^*$ Par Pollard Rho nous calculons $\log_g \gamma_{\mathfrak{q}_3}(m) \equiv 599 \mod 3751$, $\log_g \gamma_{\mathfrak{q}_5}(m) \equiv 825 \mod 3751$ et $\log_g \gamma_{\mathfrak{q}_{11}}(m) \equiv 2994 \mod 3751$. Ces valeurs sont proportionnelles à $h_1 \log \mathfrak{q}_3$,

$h_1 \log \mathfrak{q}_5$ et $h_1 \log \mathfrak{q}_{11}$.

5.5.6 Descente

Il s'agit maintenant de retrouver le logarithme discret d'un élément arbitraire du corps cible, disons $h = 55m^2 + 17m + 26$.

Descente par smoothing

Nous testons au hasard des valeurs de $e \in [\![0, 3571-1]\!]$ jusqu'à $e = 989$. En effet, pour cette valeur, $h' = h^e = 64m^2 + 98m + 79$ et $\bar{Z}' = 64\theta_1^2 + 98\theta_1 + 79$ est C-friable pour $C = 500 > B$ notre seconde borne de friabilité auxiliaire à la descente. Nous factorisons ensuite $\bar{h}'$: $\bar{h}'\mathcal{O}_1 = \langle 3, \theta_1'/3\rangle^{-4}\langle 13, \theta_1'/3 - 3\rangle\langle 17, \theta_1'/3 + 6\rangle\langle 71, \theta_1'/3 - 7\rangle\langle 353, \theta_1' + 17\rangle$. Tous les idéaux précédents sont dans la base de friabilité, sauf $\mathfrak{q} := \langle 353, \theta_1' + 17\rangle$.

Descente par spécial-$\mathfrak{q}$

Nous réduisons le réseau dont la matrice correspondante est :

$$\begin{pmatrix} 1 & 17 & 0 \\ 0 & 1 & 17 \\ 0 & 0 & 353 \end{pmatrix}.$$

Les trois vecteurs que nous obtenons alors correspondent aux polynômes $\phi_1 = 5x^2 + 2x + 1$, $\phi_2 = x^2 - 4x - 4$ et $\phi_3 = -2x^2 + 7x - 9$. Par construction on remarque que $\phi_i(\theta_1')$ est divisible par $\mathfrak{q}$ pour tout $i \in \{1, 2, 3\}$. Nous énumérons ensuite les combinaisons linéaires de ϕ_1, ϕ_2 et ϕ_3 à coefficients dans $[\![-A, A]\!]$ pour $A = 100$. Nous trouvons alors $\phi = x^2 - 4x - 4$ qui est tel que $\phi(\theta_i')$ soit B-friable pour $i \in \{1, 2, 3\}$, mais par pour les corps de nombres restant, c'est-à-dire pas pour $i \in [\![4, 11]\!]$. En utilisant la relation entre les logarithmes qui correspondent à $\phi(\theta_1')$ et $\phi(\theta_2')$, nous retrouvons le logarithme de l'idéal $\mathfrak{q}$, et donc celui de h' puis celui de h.

Ces phases de descente par smoothing ou special-$\mathfrak{q}$ peuvent être exécutées indifféremment soit en amont du crible et de l'algèbre linéaire, soit en aval. Le nombre de corps de nombres que l'on utilise dans la descente est indépendant de celui utilisé dans les phases précédentes.

Conclusion

Dans ce chapitre nous avons détaillé différentes variantes du crible multiple par corps de nombres. Initialement présenté en 1993 pour la factorisation des entiers, nous l'avons étendu pour la première fois au cas général des corps finis de caractéristique moyenne et grande. Aujourd'hui, les nombreux et très récents algorithmes [BGK15, SS16c, KB16, SS16d, SS16a, JK16, SS16b] proposés pour s'attaquer au problème du logarithme discret dans les corps finis (souvent spéciaux), considèrent tous dans la mesure du possible l'option d'un diagramme multibranches. En théorie, il s'agit donc maintenant d'une variante incontournable. En pratique néanmoins, aucun record sur des tailles pertinentes n'a utilisé d'implémentation sur plusieurs corps de nombres.

Chapitre 6

Algèbre linéaire presque creuse

Sommaire

Dans ce chapitre nous nous proposons de résoudre des problèmes d'algèbre linéaire liés à des matrices presque creuses : tout en exigeant de celles-ci qu'elles présentent une grande vacuité, nous autorisons l'existence de quelques colonnes denses comprenant d'éventuels lourds coefficients. Issue de [JP16a], cette méthode s'articule autour de la modification de l'algorithme de Block Wiedemann. Sous des conditions bien précises concernant le choix des vecteurs initiaux intervenant dans l'algorithme, nous prouvons que notre variante produit non seulement une solution du système linéaire mais donne en sus une base complète de l'ensemble des solutions. Par ailleurs, lorsque le nombre de colonnes lourdes reste petit, le coût de leur traitement demeure négligeable. Ceci facilite le calcul de logarithme discret dans les corps finis de moyenne et grande caractéristiques où ces *matrices presque creuses* apparaissent naturellement.

6.1 A propos des matrices creuses

6.1.1 Matrices creuses et cryptographie

L'algèbre linéaire est un outil incontournable aussi bien en mathématiques qu'en informatique. A la frontière de ces deux disciplines, la cryptographie n'échappe pas à cette règle, mais présente quelques différences notables : les cryptographes utilisent majoritairement de l'algèbre linéaire sur des corps finis, ce qui appelle aussi bien des avantages – aucun problème de stabilité ne peut se manifester – que des inconvénients – la notion de convergence n'a plus de sens dans un tel univers. Comme en analyse combinatoire ou lors de la résolution d'équations différentielles partielles, la cryptographie affiche aussi la particularité de faire apparaitre naturellement des matrices creuses.

Une *matrice creuse* est une matrice qui ne contient qu'un nombre relativement petit de coefficients non nuls par rapport à sa dimension. Elle prend souvent la forme d'une matrice pour laquelle chaque ligne (ou colonne) ne présente que quelques entrées non nulles.

Les systèmes linéaires creux sur des corps finis se manifestent dès les années 70 lors de la publication des premiers algorithmes de calcul de logarithmes discrets sous-exponentiels [Adl79]. Aujourd'hui encore, chaque algorithme de la famille du calcul d'indice manipule une matrice creuse, comme indiqué au Chapitre 2 : tous les records récents de calcul de logarithmes nécessitent l'extraction d'une solution d'un système linéaire creux modulo un grand entier. De la même manière, toutes les factorisations récentes d'entiers composés, qui s'appuient elles aussi sur des variantes du crible par corps de nombres (donc sur un algorithme de calcul d'indice), exécutent une sous-routine d'algèbre linéaire modulo 2.

6.1.2 Quelques méthodes de résolutions

Plusieurs méthodes existent pour calculer un élément du noyau d'une matrice creuse, ou trouver un antécédent d'un vecteur donné, par l'image de celle-ci. Nous ne nous intéressons dans ce chapitre qu'à la famille des algorithmes de Wiedemann et ses généralisations. Au lieu de calculer une famille orthogonale de vecteurs, comme d'autres algorithmes évoqués dans la partie 2.3.2, Wiedemann propose en 1986 [Wie86] de reconstruire le polynôme minimal de la matrice en question. Plus précisément, cet algorithme calcule une suite de scalaires de la forme :

$$^t w A^i v$$

où v et w sont des vecteurs et A la matrice creuse intervenant dans le problème initial. Il tente ensuite d'extraire une relation de récurrence linéaire qui se vérifie pour toutes les sous-séquences glissantes de la suite. En 1994, afin de réaliser des calculs atteignables en pratique, Coppersmith [Cop94] parallélise l'algorithme de Wiedemann sur le corps fini binaire $\mathbb{F}_2$. Mieux, il parvient à distribuer les calculs sur des processeurs indépendants. Un an plus tard, Kaltofen [Kal95] généralise non seulement cet algorithme à tous les corps finis, mais donne une variante rigoureuse de la méthode heuristique de Coppersmith. L'idée principale de Coppersmith consiste à calculer une suite de matrices de la forme :

$$^t W A^i V$$

où V et W ne sont plus des vecteurs comme précédemment, mais des ***blocs*** de vecteurs – donnant ainsi à ce procédé le nom qu'on lui connait. Cette étape

se parallélise en distribuant les vecteurs du bloc V à différents processeurs ou CPU, disons c d'entre-eux. La complexité asymptotique de l'extraction d'une relation de récurrence linéaire correspondant à la suite de matrices est en $\tilde{O}(cN^2)$ où N est la plus grande des deux dimensions de la matrice A qui est donnée dans le problème initial. Beckerman et Labahn [BL94] présentent en 1994 une amélioration pour remplacer cette étape en temps sous-quadratique, elle-même optimisée ensuite en 2002 [Tho02]. La variante de Thomé réduit en effet la complexité de la recherche d'une relation de récurrence à $\tilde{O}(c^2N)$. La méthode qui constitue l'état de l'art de la question aujourd'hui apparait un an plus tard. Il s'agit du résultat de Giorgi, Jeannerod et Villard [GJV03] qui s'exécute en temps $\tilde{O}(c^{\omega-1}N)$, où ω est l'exposant de la complexité de la multiplication matricielle. A titre indicatif la valeur actuelle de cette exposant est approximée à $\omega \approx 2.37286$. Elle provient d'une légère modification de l'algorithme de Coppersmith-Winograd [CW90] proposée par Le Gall [Gal14] et présentée en 2014 à ISSAC. La même conférence offre à ce propos un cadre à la présentation d'une amélioration pratique de Giorgi et Lebreton pour l'algorithme de Block Wiedemann : ils proposent ainsi un algorithme online [GL14] qui requiert potentiellement moins d'informations en entrée et peut se terminer plus tôt.

6.2 Des algorithmes quadratiques pour les matrices creuses

Cette section reprend les problèmes classiques d'algèbre linéaire qui se posent, entre autres, pour les matrices creuses. Nous y expliquons comment la matrice initiale est pré-transformée en une matrice carrée au paragraphe 6.2.1. Le paragraphe 6.2.2 brosse les grandes lignes de l'algorithme proposé par Wiedemann pour résoudre des systèmes linéaires possédant autant d'inconnues que d'équations, tandis que le paragraphe 6.2.3 détaille la variante parallélisée que l'on doit à Coppersmith. Plus précisément, le but est de résoudre le problème suivant :

Problème 6.2.1. Soit $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ un corps fini d'ordre premier et $S \in \mathcal{M}_{n\times N}(\mathbb{K})$ une matrice creuse (potentiellement rectangulaire) avec λ coefficients non nuls par ligne. Soit $\vec{v}$ un vecteur à n coefficients. Le problème consiste à trouver un vecteur $\vec{x}$ à N coefficients tel que :

$$S \cdot \vec{x} = \vec{v}$$

ou, de manière alternative, un vecteur non nul $\vec{x}$ tel que :

$$S \cdot \vec{x} = 0.$$

En pratique, ce problème est souvent généralisé sur des anneaux $\mathbb{Z}/\mathcal{N}\mathbb{Z}$ pour un entier $\mathcal{N}$ de factorisation inconnue. Cependant, afin de simplifier l'exposition de notre propos, et considérant que l'algorithme de [GJV03] n'est prouvé que sur des corps, nous nous restreignons dans la suite au cas des corps premiers.

6.2.1 Pré-transformation d'une matrice creuse vers une matrice carrée

Pour calculer des produits matrice-vecteur de la forme $(A^i\vec{y})_{i>0}$, les algorithmes de Wiedemann et de Block Wiedemann doivent tous manipuler des matrices carrées – en l'absence de quoi la définition des puissances de la matrice ne pourrait avoir de sens. Aussi, si $N \neq n$, il est nécessaire de réfléchir à une étape préliminaire pour transformer la matrice d'entrée S potentiellement rectangulaire en une matrice carrée. Par exemple, il est possible de compléter la matrice avec des zéros et d'appliquer l'analyse de [KS91] pour la résolution de systèmes linéaires qui ne sont pas de rang plein. Sans rentrer dans les détails, il s'agit de multiplier à gauche et à droite par des matrices aléatoires, puis de tronquer le résultat de ce produit pour obtenir une matrice carrée inversible de dimension certes plus petite, mais préservant le rang de la matrice originale.

En pratique, des méthodes heuristiques sont le plus souvent utilisées. L'exemple classique consiste à créer une matrice aléatoire creuse $R \in \mathcal{M}_{N\times n}(\mathbb{K})$ avec au plus λ coefficients non nuls par ligne, et de transformer après coup les deux problèmes précédents en la recherche d'un vecteur $\vec{x}$ tel que : $(RS)\vec{x} = R\vec{v}$ ou, de manière alternative, tel que : $(RS)\vec{x} = 0$. En posant $A = RS$ et $\vec{y} = R\vec{v}$, nous pouvons réécrire le problème 6.2.1 comme étant la recherche d'un vecteur $\vec{x}$ tel que :

$$A \cdot \vec{x} = \vec{y}$$

ou tel que :

$$A \cdot \vec{x} = 0,$$

selon le problème initial. Par ailleurs, afin d'écarter la solution triviale nulle lors de la recherche d'un élément du noyau de A, il est fréquent de calculer au préalable $\vec{y} = A\vec{r}$ pour un vecteur aléatoire $\vec{r}$, puis de résoudre $A\vec{x} = \vec{y}$ et de renvoyer enfin $\vec{x} - \vec{r}$ comme élément du noyau.

Nous n'irons pas plus loin concernant les détails de pré-transformation qu'il est d'usage d'effectuer. En effet, dans le cadre de notre algorithme dédié aux matrices presque creuses – et donc aux matrices creuses – nous proposons au paragraphe 6.3.2 une technique alternative, à la fois simple, et dont

nous pouvons démontrer qu'elle renvoie le résultat escompté, sous certaines conditions techniques explicitées plus tard.

6.2.2 L'algorithme de Wiedemann

Considérons maintenant une matrice carrée A de taille $N \times N$ et notons m_A le nombre d'opérations nécessaires pour calculer le produit d'un vecteur quelconque de $\mathbb{K}^N$ par A. L'algorithme de Wiedemann cherche une suite non triviale de coefficients $(a_i)_{0 \leq i \leq N}$ tels que :

$$\sum_{i=0}^{N} a_i A^i = 0. \tag{6.1}$$

Résolution du problème $A\vec{x} = \vec{y}$.

Si A est inversible, alors nous pouvons supposer que $a_0 \neq 0$. En effet, si $a_0 = 0$ nous pouvons réécrire $0 = \sum_{i=1}^{N} a_i A^i = A^\delta(\sum_{i=1}^{N} a_i A^{i-\delta})$ où a_δ correspond au premier coefficient non nul. En multipliant par $(A^{-1})^\delta$ il découle l'égalité matricielle $\sum_{i=0}^{N-\delta} a_{i+\delta} A^i = 0$. Aussi, en décalant les coefficients jusqu'à déceler le premier qui n'est pas nul, nous pouvons bel et bien noter $a_0 \neq 0$. Appliquons maintenant l'équation (6.1) au vecteur $\vec{x}$ que nous cherchons. Il vient alors $-a_0\vec{x} = \sum_{i=1}^{N} a_i A^i \vec{x} = \sum_{i=1}^{N} a_i A^{i-1}(A\vec{x})$. Finalement, nous retrouvons :

$$\vec{x} = -(1/a_0) \sum_{i=1}^{N} a_i A^{i-1} \vec{y}.$$

Cette dernière somme peut se calculer en effectuant N multiplications séquentielles entre la matrice A et le vecteur initial $\vec{y}$ – à la première itération, la multiplication suivante prenant en entrée A puis le vecteur résultat de la multiplication précédente, et ainsi de suite. Le coût total de calcul de $\vec{x}$ s'élève alors à $O(N \cdot m_A)$ opérations.

Résolution du problème $A\vec{x} = 0$.

Supposons qu'il existe un élément non trivial du noyau de A. Nous en déduisons alors que $a_0 = 0$. En notant encore une fois δ le premier indice tel que $a_\delta \neq 0$, nous obtenons pour tout vecteur $\vec{r}$ l'égalité vectorielle $0 = \sum_{i=\delta}^{N} a_i A^i \vec{r} = A^\delta(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r})$. Ainsi, la matrice $\sum_{i=\delta}^{N} a_i A^{i-\delta}$ est non nulle. En effet, dans le cas contraire, l'égalité $a_\delta \mathrm{Id} + \sum_{i=\delta+1}^{N} a_i A^{i-\delta} = a_\delta \mathrm{Id} + A(\sum_{i=\delta+1}^{N} a_i A^{i-\delta-1}) = 0$ mènerait au produit matriciel $A(-(1/a_\delta)\sum_{i=\delta+1}^{N} a_i A^{i-\delta-1}) = \mathrm{Id}$, alors que la matrice A est supposée non inversible. Pour un vecteur aléatoire $\vec{r}$, la somme

Algorithm 1 L'algorithme de Wiedemann pour $A\vec{x} = \vec{y}$

Entrée : Une matrice A de taille $N \times N$, un vecteur $\vec{y} \neq 0$ à N coefficients
Sortie : $\vec{x}$ tel que $A \cdot \vec{x} = \vec{y}$.

Calcul d'une séquence de scalaires

1: $\vec{v}_0 \leftarrow\in \mathbb{K}^N$, $\vec{w} \leftarrow\in \mathbb{K}^N$ deux vecteurs aléatoires
2: **for** $i = 0, \cdots, 2N$ **do**
3: $\quad \lambda_i \leftarrow \vec{w} \cdot \vec{v}_i$
4: $\quad \vec{v}_{i+1} \leftarrow A\vec{v}_i$
5: **end for**

Algorithme de Berlekamp-Massey

6: Depuis $\lambda_0, \cdots, \lambda_{2N}$ retrouver les coefficients $(a_i)_{0 \leq i \leq N}$ *t.q.* $\sum_{i=0}^{N} a_i A^i = 0$ et $a_0 \neq 0$.

Résolution

7: **return** $-(1/a_0) \sum_{i=1}^{N-1} a_{i+1} A^i \vec{y}$.

vectorielle $\sum_{i=\delta}^{N} a_i A^{i-\delta} \vec{r}$ est donc non nulle avec grande probabilité. Plus précisément, elle est nulle si et seulement si le vecteur $\vec{r}$ appartient au noyau de la matrice non nulle $\sum_{i=\delta}^{N} a_i A^{i-\delta}$. Le noyau d'une matrice non nulle étant au plus de dimension $N-1$, la probabilité qu'un vecteur aléatoire soit dans ce noyau est majorée par $|\mathbb{K}|^{N-1}/|\mathbb{K}|^N = 1/|\mathbb{K}|$.

Enfin, le calcul itératif de la suite $A(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r})$, $A^2(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r})$, $\cdots, A^{\delta}(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r})$ donne un élément du noyau de A en $O(N \cdot m_A)$ opérations. En effet, le premier indice j dans $[\![1, \delta]\!]$ tel que $A^j(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r}) = 0$ montre que $A^{j-1}(\sum_{i=\delta}^{N} a_i A^{i-\delta}\vec{r}) \neq 0$ appartient au noyau de A. Par conséquent, cette méthode trouve un élément non trivial de $\mathrm{Ker}(A)$ avec une probabilité plus grande que $(|\mathbb{K}|-1)/|\mathbb{K}|$, qui tend rapidement vers 1 lorsque la cardinalité du corps augmente.

Comment trouver des coefficients a_i satisfaisant l'équation (6.1).

Le théorème de Cayley-Hamilton témoigne de l'existence d'un polynôme défini comme $P = \det(A - X \cdot \mathrm{Id})$ qui annule la matrice A, *i.e.* tel que $P(A) = 0$. L'idéal des polynômes annulateurs de A n'est pas vide, et il possède un polynôme unitaire du plus petit degré que l'on appelle le polynôme minimal de A. Nous savons donc qu'il existe un polynôme de degré au plus N dont les coefficients satisfont l'équation (6.1), aussi nous en déduisons que le degré du polynôme minimal est au plus N. Le calcul direct d'un tel polynôme serait pourtant bien trop coûteux ; aussi, pour trouver un tel polynôme nous procédons plutôt par conditions nécessaires.

Soit $(a_i)_{i \in [\![0,N]\!]}$ tel que :

$$\sum_{i=0}^{N} a_i A^i = 0.$$

Alors, pour tout vecteur $\vec{v}$ nous obtenons :

$$\sum_{i=0}^{N} a_i A^i \vec{v} = 0.$$

De même, pour tout vecteur $\vec{w}$ et pout tout entier j nous écrivons :

$$\sum_{i=0}^{N} a_i {}^t\vec{w} A^{i+j} \vec{v} = 0.$$

Réciproquement, si l'égalité $\sum_{i=0}^{N} a_i {}^t\vec{w} A^{i+j} \vec{v} = 0$ est vraie pour toute paire de vecteurs aléatoires $\vec{v}$ et $\vec{w}$, et pour tout entier j dans l'intervalle $[\![0, N]\!]$, alors la probabilité d'obtenir des coefficients vérifiant l'équation (6.1) est très élevée, sous l'hypothèse que la cardinalité du corps soit suffisamment grande [Kal95]. Ainsi, l'algorithme de Wiedemann cherche des coefficients a_i qui annulent la suite de scalaires ${}^t\vec{w} A^i \vec{v}$. Pour se faire, il est possible d'utiliser l'algorithme classique de Berlekamp-Massey [Ber68, Mas69] qui trouve le polynôme minimal d'une suite récursive linéaire dans un corps arbitraire. En quelques mots, il s'agit de considérer la fonction génératrice f de la suite ${}^t\vec{w}\vec{v}, {}^t\vec{w}A\vec{v}, {}^t\vec{w}A^2\vec{v}, \cdots, {}^t\vec{w}A^{2N}\vec{v}$ et de trouver ensuite deux polynômes g et h tel que $f = g/h \mod X^{2N}$. Une alternative consiste à remplacer l'algorithme de Berlekamp-Massey par un algorithme de recherche de demi pgcd étendu, ce qui entraine une méthode quasi-linéaire en la taille de la matrice A.

6.2.3 L'algorithme de Block Wiedemann

Block Wiedemann est la variante parallélisée de l'algorithme de Wiedemann introduite pour la première fois par Coppersmith. Elle s'adresse aux situations dans lesquelles il est possible de calculer les suites de produits matrice-vecteur en s'appuyant non plus un mais sur ℓ processeurs. Dans ce cas, plutôt que de résoudre l'équation (6.1), nous cherchons, étant donnés ℓ vecteurs $\vec{v}_1, \cdots, \vec{v}_\ell$, des coefficients a_{ij} tels que :

$$\sum_{j=1}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{v}_j = 0 \tag{6.2}$$

Nous remarquons que le nombre de coefficients (ici les a_{ij}) à calculer demeure du même ordre de grandeur que celui qui apparait dans la version non parallélisée de l'algorithme, c'est-à-dire N.

Algorithm 2 L'algorithme de Block Wiedemann pour $A\vec{x} = \vec{0}$

Entrée : Une matrice A de dimension $N \times N$
Sortie : $\vec{x}$ tel que $A \cdot \vec{x} = \vec{0}$.

Calcul d'une suite de matrices

$\vec{r}_1 \leftarrow\in \mathbb{K}^N, \cdots, \vec{r}_\ell \leftarrow\in \mathbb{K}^N$ et $\vec{w}_1 \leftarrow\in \mathbb{K}^N, \cdots, \vec{w}_\ell \leftarrow\in \mathbb{K}^N$
$\vec{v}_1 \leftarrow A\vec{r}_1, \cdots, \vec{v}_\ell \leftarrow A\vec{r}_\ell$
for chacun des ℓ processeurs indicés par j **do**
  $u_0 \leftarrow v_j$
  for $i = 0, \cdots, 2\lceil N/\ell \rceil$ **do**
    for $k = 1, \cdots, \ell$ **do**
      $\lambda_{i,j,k} \leftarrow \vec{w}_k \cdot \vec{u_i}$
    end for
    $\vec{u_{i+1}} \leftarrow A\vec{u_i}$
  end for
end for
for $i = 0, \cdots, 2\lceil N/\ell \rceil$ **do**
  $M_i \leftarrow (\lambda_{i,j,k})$ la matrice $\ell \times \ell$ qui contient tous les produits de la forme ${}^t\vec{w}A^i\vec{v}$
end for

Algorithme de Thomé ou de Giorgi, Jeannerod, et Villard

Depuis $M_0, \cdots, M_{2\lceil N/\ell \rceil}$ retrouver les coefficients a_{ij} *t.q.* $\sum_{j=1}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{v}_j = \vec{0}$.

Résolution

$\delta \leftarrow$ le premier indice dans $[\![1, \lceil N/\ell \rceil]\!]$ tel qu'il existe j in $[\![1, \ell]\!]$ vérifiant $a_{\delta j} \neq 0$.
$\vec{b} \leftarrow \sum_{j=1}^{\ell} \sum_{i=\delta}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{r}_j$.
$\vec{k} \leftarrow$ Erreur : élément du noyau trivial
while $\vec{b} \neq 0$ **do**
  $\vec{k} \leftarrow \vec{b}$
  $\vec{b} \leftarrow A\vec{k}$
end while
return $\vec{k}$

Résolution du problème $A\vec{x} = \vec{0}$.

Nous choisissons ici ℓ vecteurs aléatoires $\vec{r}_1, \cdots, \vec{r}_\ell$ puis nous posons $\vec{v}_i = A\vec{r}_i$. Soit δ le premier indice dans l'intervalle $[\![1, \lceil N/\ell \rceil]\!]$ pour lequel il existe un entier j dans l'intervalle $[\![1, \ell]\!]$ qui vérifie $a_{\delta j} \neq 0$. L'équation (6.2) entraine alors $\sum_{j=1}^{\ell} \sum_{i=\delta}^{\lceil N/\ell \rceil} a_{ij} A^{i+1} \vec{r}_j = \vec{0}$, *i.e.* $A^{\delta+1}(\sum_{j=1}^{\ell} \sum_{i=\delta}^{\lceil N/\ell \rceil} a_{ij} A^{i-\delta} \vec{r}_j) = \vec{0}$. Soit maintenant $\vec{b}$ le vecteur $\sum_{j=1}^{\ell} \sum_{i=\delta}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{r}_j$. Selon le résultat de Kaltofen [Kal95], le vecteur $\vec{b}$ est non nul avec une très forte probabilité. Aussi, calculer itérativement $A\vec{b}$, $A^2\vec{b}$, $\cdots$, $A^\delta\vec{b}$ permet de nouveau d'obtenir un élément du noyau de A en $O(N \cdot m_A)$ opérations. En effet, le premier indice k dans l'intervalle $[\![1, \delta]\!]$ tel que $A^k\vec{b} = 0$ montre que $A^{k-1}\vec{b}$ est bel et bien un élément non trivial du noyau de A.

Résolution du problème $A\vec{x} = \vec{y}$.

Différentes approches sont envisageables pour résoudre $A\vec{x} = \vec{y}$. Ainsi, dans l'article [Kal95] la taille de A est par exemple augmentée de 1 : le vecteur $\vec{y}$ est ajouté à l'intérieur d'une nouvelle colonne, disons à l'extrême droite, et la création d'une ligne de zéros au bas de la matrice rend de nouveau celle-ci carrée. Un élément aléatoire du noyau de la matrice construite par la méthode précédente produit alors une solution de $A\vec{x} = \vec{y}$, dès lors que le dernier coefficient de cet élément solution est non nul.

Une autre option consiste à poser $\vec{v}_1 = \vec{y}$ et à choisir pour tout $i \in [\![2, \ell]\!]$ des vecteurs $\vec{v}_i = A\vec{r}_i$, où chaque $\vec{r}_i$ est un vecteur aléatoire de taille adéquate, et où l'on suppose que $a_{01} \neq 0$. L'équation (6.2) donne alors :

$$\sum_{i=0}^{\lceil N/\ell \rceil} a_{i1} A^i \vec{y} + \sum_{j=2}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} A^{i+1} \vec{r}_j = 0.$$

En multipliant par l'inverse de A, nous obtenons :

$$a_{01}\vec{x} + \sum_{i=1}^{\lceil N/\ell \rceil} a_{i1} A^i \vec{y} + \sum_{j=2}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{r}_j = 0.$$

Il est alors possible de retrouver $\vec{x}$ en calculant simplement :

$$(-1/a_{01}) \cdot \left(\sum_{i=1}^{\lceil N/\ell \rceil} a_{i1} A^{i-1} \vec{y} + \sum_{j=2}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} A^i \vec{r}_j \right).$$

Ceci peut être exécuté pour un coût total de $O(N \cdot m_A)$ opérations parallélisées sur ℓ processeurs : chacun démarre avec son propre vecteur initial $\vec{v}_j$ et calcule une suite de produits matrice-vecteur de la forme $A^i \vec{v}_j$. Le coût de calcul de chaque suite est de $O(N \cdot m_A/\ell)$ opérations arithmétiques. Nous ne traitons pas dans ce paragraphe du cas déviant pour lequel $a_{01} = 0$ puisque la section 6.3 couvrira tous les cas de figures pour les matrices presque creuses, donc pour les matrices creuses.

Comment trouver des coefficients a_i **satisfaisant l'équation** (6.2).

Soit $\vec{v}_1, \cdots, \vec{v}_\ell$ un ensemble de ℓ vecteurs ; considérons les $\ell(\lceil N/\ell \rceil)$ éléments résultats des produits matrice-vecteur de la forme $A^i \vec{v}_j$ qui apparaissent dans la somme de l'équation (6.2). Puisque $\ell(\lceil N/\ell \rceil) > N$, tous ces vecteurs ne

peuvent être indépendants. Il existe donc des coefficients qui vérifient l'équation (6.2). Nous prenons exemple sur l'algorithme de Wiedemann et nous procédons maintenant par conditions nécessaires. Soit $\vec{w}_1, \ldots, \vec{w}_\ell$ un ensemble de ℓ vecteurs pour lequel nous supposons que, pour tout entier κ dans l'intervalle $[\![0, \lceil N/\ell \rceil]\!]$ et tout entier k dans $[\![1, \ell]\!]$, nous avons :

$$\sum_{j=1}^{\ell} \sum_{i=0}^{\lceil N/\ell \rceil} a_{ij} {}^t\vec{w}_k A^{i+\kappa} \vec{v}_j = 0.$$

La probabilité que les coefficients a_{ij} satisfassent l'équation (6.2) est alors proche de 1 lorsque $\mathbb{K}$ est grand – de nouveau, nous renvoyons à [Kal95]. En revanche, si le corps $\mathbb{K}$ est petit, il est facile d'augmenter la probabilité en montant le nombre de vecteurs w au delà de ℓ, en suivant par exemple l'analyse faite par Coppersmith dans [Cop94]. L'algorithme de Block Wiedemann cherche donc des coefficients qui annulent la suite de $2\lceil N/\ell \rceil$ petites matrices de dimension $\ell \times \ell$ données par :

$$\left({}^t\vec{w}_k A^\nu \vec{v}_j \right).$$

Ici, $\nu \in [\![0, 2\lceil N/\ell \rceil]\!]$ dénombre les matrices, tandis que les indices k et j représentent respectivement les numéros de colonnes et de lignes au sein de chacune des matrices. Il est possible de calculer les coefficients a_{ij} en temps sous-quadratique – nous renvoyons à la section 6.2.4 pour de plus amples détails. L'algorithme de Giorgi, Jeannerod, Villard permet ainsi d'atteindre une complexité en $\tilde{O}(\ell^{\omega-1}N)$. Ceci achève de structurer les étapes de Block Wiedemann telles qu'elles sont décrites dans l'algorithme 2.

Par ailleurs, la considération conjointe du temps de calcul nécessaire aux produits matrice-vecteur et à la recherche des coefficients nous permet d'exprimer la complexité totale de l'algorithme comme $O(N \cdot m_A) + \tilde{O}(\ell^{\omega-1}N)$. Notons que la partie en $O(N \cdot m_A)$ peut être efficacement distribuées sur ℓ processeurs tandis que la partie en $\tilde{O}(\ell^{\omega-1}N)$ résulte d'un calcul séquentiel.

Remarque 6.2.2. Dans ce paragraphe tout avons toujours implicitement supposé que le nombre de suites ℓ à calculer était précisément égal au nombre de processeurs c à disposition. Il s'agit en effet du choix le plus naturel lors de l'application de Block Wiedemann, puisque l'augmentation de ℓ au delà du nombre de processeurs ne peut que dégrader la performance générale de l'algorithme. Plus précisément, un tel changement laisserait la contribution en $O(N \cdot m_A)$ inaltérée, mais gonflerait la partie en $\tilde{O}(\ell^{\omega-1}N)$. Cependant, puisque nous serons amenés à considérer dans la section 6.3 des valeurs de ℓ plus grandes que celles de c, il est utile de comprendre que l'on peut

avoir un intérêt à calculer séquentiellement plusieurs suites indépendantes sur chacun des processeurs. Dans un tel cas de figure il est alors ingénieux de rendre en pratique le nombre de suites à calculer égal à un multiple du nombre de processeurs, afin de minimiser le temps d'horloge des produits matrice-vecteur.

6.2.4 Bases minimales

Nous reprenons dans cette partie un résultat important de Giorgi, Jeannerod et Villard [GJV03], évoqué au paragraphe précédent 6.2 et nécessaire à l'exécution de Block Wiedemann. Bien qu'il existe plusieurs variantes (moins efficaces toutefois asymptotiquement) de ce résultat qui puissent être utilisées en pratique pour le cas des matrices creuses, il s'agit dans la configuration que nous présenterons d'un élément clef. Il n'admet, à notre connaissance, pas de remplacement possible lorsque l'on souhaite obtenir un algorithme rigoureux. Soit $\mathbb{K}$ un corps fini et G une matrice de séries formelles sur $\mathbb{K}$ de dimension $m \times n$ avec $n < m$, c'est-à-dire un élément de $K[[X]]^{m\times n}$. Pour une approximation d'ordre β, nous considérons des vecteurs lignes $\vec{u}(X)$ m-dimensionnels de polynômes qui vérifient l'équation :

$$\vec{u} \cdot G \equiv \vec{0} \mod X^{\beta}. \tag{6.3}$$

Pour un vecteur $\vec{u}$ de polynômes, nous rappelons que nous définissons son degré $\deg(\vec{u})$ comme étant égal au maximum des degrés de ses coordonnées.

Définition 6.2.3. Une σ-base d'un ensemble de solutions de l'équation (6.3) est une matrice carrée M de dimension $m \times m$ de polynômes de $\mathbb{K}[X]$ telle que :

- Chaque vecteur ligne $\vec{M}_i$ de M est une solution de (6.3).
- Pour toute solution $\vec{u}$ de (6.3), il existe une unique famille constituée de m polynômes $c_1, \ldots, c_m$ telle que, pour chaque indice $i \in [\![1, m]\!]$:

$$\deg(c_i \vec{M}_i) \leq \deg(\vec{u}),$$

et, de plus :

$$\vec{u} = \sum_{i=1}^{m} c_i \vec{M}_i.$$

Intuitivement, une σ-base permet donc d'obtenir toute solution sous forme d'une combinaison linéaire des lignes avec une notion de minimalité dans le

degré des termes obtenus. Giorgi, Jeannerod et Villard proposent un algorithme qui calcule une σ-base de l'équation (6.3) en utilisant $\tilde{O}(m^\omega b)$ opérations algébriques dans $\mathbb{K}$. Lorsque l'on considère des implémentations pratiques de cet algorithme, et particulièrement lorsque les valeurs de m restent petites, nous remarquons que, l'exposant ω étant remplacé par 3, cette complexité correspond à celle donnée par l'algorithme de Thomé dans [Tho02].

6.3 Un algorithme pour les matrices presque creuses

Comme vu précédemment, les méthodes des sous-espaces de Krylov tout comme les algorithmes de Wiedemann coûtent un nombre de multiplications matrice-vecteur égal à un petit multiple de la dimension de la matrice : pour une matrice contenant λ entrées non nulles par ligne en moyenne, le coût de ces multiplications est $O(\lambda N^2)$. Avec Block Wiedemann, il est possible de distribuer le coût de ces produits sur c machines. Dans ce cas, la recherche de relation de récurrence ajoute un coût supplémentaire de la forme $\tilde{O}(c^{\omega-1}N)$. Pour une *matrice presque creuse*, qui comprend ∂ colonnes denses juxtaposées à sa partie creuse, le coût des multiplications matrice-vecteur augmente. Aussi, la complexité totale devient $O((\lambda+\partial)N^2)$ avec un coût supplémentaire de $\tilde{O}(c^{\omega-1}N)$ pour Block Wiedemann.

Dans cette section notre visée est triple. Nous cherchons tout d'abord à adapter l'algorithme de Block Wiedemann pour abaisser le coût de l'algèbre linéaire des matrices qui présentent un caractère presque creux, et réduire celui-ci à une complexité en :

$$O(\lambda N^2) + \tilde{O}(\max(c,\partial)^{\omega-1}N).$$

En particulier, lorsque le nombre de colonnes denses est inférieur au nombre de processeurs, nous montrons que la présence de ces colonnes malvenues n'affecte pas la complexité de la résolution des systèmes linéaires associés.

Parallèlement, nous donnons des conditions plus précises sur le choix des vecteurs $\vec{v}_i$ et $\vec{w}_i$ que celles implicitement données dans l'algorithme de Block Wiedemann. Plutôt que d'insister sur l'importance de choix aléatoires comme dans [Kal95], nous proposons des conditions explicites à propos de ces choix, qui permettent de rendre notre algorithme rigoureux.

Lorsque celles-ci sont satisfaites, nous démontrons ainsi que notre algorithme ne trouve pas simplement une solution aléatoire du système d'équations comme cela était précédemment le cas, mais renvoie une description explicite et complète de l'ensemble de toutes les solutions.

La clef de notre méthode repose sur la manipulation de la partie creuse de la matrice d'entrée séparément des colonnes denses. Celles-ci sont utili-

sées pour initialiser les suites de matrices calculées par l'algorithme de Block Wiedemann. Détaillons maintenant cette idée.

6.3.1 Entre vacuité et densité

Dans toute la suite nous nous attachons à résoudre des problèmes d'algèbre linéaire de la forme suivante :

Problème 6.3.1. Soit M une matrice de taille $N \times (s + \partial)$ avec des coefficients dans un corps $\mathbb{K}$. Nous supposons qu'il existe deux matrices de plus petite taille $M_s \in \mathcal{M}_{N\times s}(\mathbb{K})$ et $M_\partial \in \mathcal{M}_{N\times\partial}(\mathbb{K})$ telles que :

1. $M = M_s|M_\partial$, où | est la concaténation matricielle.[1]
2. M_∂ est quelconque.
3. M_s est creuse. Nous faisons l'hypothèse supplémentaire qu'elle contient au plus λ coefficients non nuls par ligne.

Si $\vec{y}$ est un vecteur à N coefficients, le problème consiste à trouver tous les vecteurs $\vec{x}$ à $s + \partial$ coefficients tels que :

$$M \cdot \vec{x} = \vec{y}$$

ou, de façon alternative, tels que :

$$M \cdot \vec{x} = \vec{0}.$$

Une telle matrice M est dite *∂-presque creuse*, ou, de manière abrégée, *presque creuse* lorsque la valeur de ∂ est implicite. Nous soulignons la grande plage de matrices qui correspondent à cette définition : il n'y a a priori aucune restriction sur le nombre de colonnes denses de cette matrice. Plus particulièrement, une matrice creuse est 0-presque creuse tandis qu'une matrice dense est N-presque creuse.

L'une des conséquences intéressantes du fait que nous souhaitons construire l'ensemble des solutions d'un tel problème d'algèbre linéaire est le fait qu'il suffit de résoudre le second sous-problème non homogène du problème 6.3.1. En effet, il est facile de transformer une résolution de $M \cdot \vec{x} = \vec{y}$ en une résolution de $M' \cdot \vec{x'} = \vec{0}$ pour une matrice presque creuse M' très proche de M. Il suffit de poser $M' = M|\vec{y}$, c'est-à-dire de définir la seconde matrice comme

1. Notre méthode fonctionne aussi bien pour une définition plus large dans laquelle les ∂ colonnes denses de la matrice se situent à n'importe quelles positions. Il suffirait alors de localiser celles-ci puis de réordonner les colonnes du problème d'algèbre linéaire en jeu. Néanmoins, en vue de simplifier l'exposition de notre propos, nous supposons que toutes les colonnes denses de la matrice se situent précisément à droite de celle-ci.

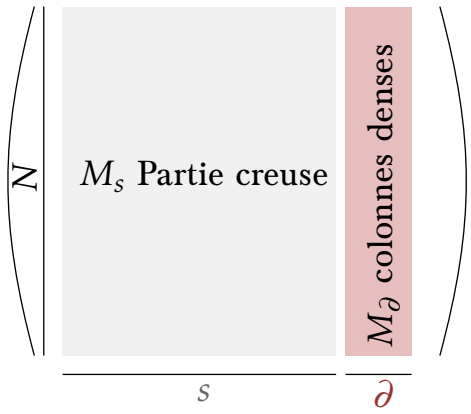


Figure 6.1 – Paramètres d'un problème d'algèbre linéaire presque creux

étant la concaténation de la première avec une colonne dense supplémentaire égale à $\vec{y}$. Nous observons alors que le vecteur $\vec{x}$ est une solution de $M \cdot \vec{x} = \vec{y}$ si et seulement si $\vec{x'} = {}^t({}^t\vec{x} \mid -1)$ est une solution de $M' \cdot \vec{x'} = \vec{0}$. En conservant cette transformation à l'esprit, nous détaillons dans la suite comment calculer une base du noyau d'une matrice presque creuse ; lorsque nous aurons résolu le premier sous-problème, il faudra simplement à la fin du procédé veiller à sélectionner uniquement dans le noyau de M' les vecteurs avec le coefficient -1 en dernière position.

Aussi, les deux variantes qui apparaissent dans le problème 6.3.1 sont naturellement plus liées l'une à l'autre dans notre configuration que leurs homologues données par le problème 6.2.1, dans le contexte plus classique de l'algorithme de (Block) Wiedemann.

La figure 6.2 indique le chemin formé par les différentes étapes que nous nous proposons de suivre pour construire une bijection entre le noyau de M et un sous-ensemble de solutions résultant du calcul d'une base minimale, elle-même obtenue grâce à l'algorithme de Giorgi, Jeannerod et Villard.

6.3.2 Pré-transformation pour une matrice presque creuse

Si $N = s$, c'est-à-dire si la partie creuse est carrée d'emblée, aucune manipulation n'est nécessaire. Par ailleurs, le cas $N < s$ n'apparait pas en pratique, car il suffit d'ajouter de nouvelles équations pour augmenter le nombre de lignes de la matrice. Pour les calculs de logarithmes discrets par exemple, ces équations sont facilement générées. Néanmoins, dans les rares cas de figures où cette dégénérescence serait incontournable, l'approche la plus simple consiste à déplacer artificiellement $s - N$ colonnes de la partie creuse vers la partie dense de la matrice. Après cette transformation, la partie creuse est

alors carrée, de dimension $(N \times N)$, tandis que le nombre de colonnes dans M_∂ est augmenté. Il devient $\partial' = \partial + s - N$.

Ajouts de zéros

Dorénavant nous nous attachons donc à la configuration pour laquelle la partie creuse comporte plus de lignes que de colonnes, c'est-à-dire lorsque $N > s$. Pour rendre cette sous-matrice M_s carrée, une méthode simple consiste à plonger cette matrice rectangulaire M_s à l'intérieur d'une matrice carrée A, qui n'est rien d'autre que la matrice M_s elle-même concaténée avec un nombre suffisant de colonnes nulles à droite de cette matrice. Bien que cette proposition dite d'ajouts de zéros (*zero-padding method* en anglais) suffise à l'analyse théorique, d'autres méthodes de pré-transformations aléatoires restent valides en pratique.

Trouver un élément ${}^t(x_s, x_\partial)$ du noyau de la matrice M est équivalent à trouver un vecteur plus long ${}^t(x_s, x_{tra}, x_\partial)$ du noyau de $(A|M_\partial)$, où x_s, x_{tra} et x_∂ sont des vecteurs lignes qui appartiennent respectivement à $\mathbb{K}^s$, $\mathbb{K}^{N-s}$ et $\mathbb{K}^\partial$. Le lecteur taquin aura compris que le mauvais jeu de lettres sur x_{tra} désigne les coordonnées supplémentaires.

Définition de A par une matrice aléatoire.

Pour préconditionner notre matrice presque creuse M de dimension $N \times (s+\partial)$ telle que $N \geq s$, nous pouvons aussi choisir de passer par une matrice R définie comme étant égale à :

$$R = \left(\begin{array}{c|c} R_s & 0 \\ \hline 0 & I_\partial \end{array} \right),$$

où R_s est une matrice creuse aléatoire de dimension $s \times N$ qui est supposée surjective, et I_∂ est la matrice identité $\partial \times \partial$. Nous remarquons que R est surjective si et seulement si R_s est aussi surjective. Par ailleurs, une multiplication matricielle par bloc nous indique que l'on peut écrire :

$$MR = (A|M_\partial),$$

avec $A = M_s R_s$ une matrice creuse carrée de dimension N.

L'idée consiste de nouveau à calculer une base du noyau de $(A|M_\partial)$. En effet, si nous avons un élément $x \in \mathrm{Ker}(A|M_\partial)$ alors $R \cdot x \in \mathrm{Ker}(M)$. Réciproquement, puisque R est surjective, si $y \in \mathrm{Ker}(M)$ alors il existe x tel que $Rx = y$ et nous obtenons $x \in \mathrm{Ker}(A|M_\partial)$.

Dans la suite nous considérons la matrice carrée A sans nous préoccuper de la façon dont elle a été construite.

6.3.3 Conditions sur les blocs V et W

Soit β un entier à déterminer plus tard – on trouvera une définition de β au paragraphe 6.3.6 – et notons $\vec{\delta}_1$, …, $\vec{\delta}_\partial$ l'ensemble des vecteurs colonnes de M_∂. Nous choisissons ensuite $\ell - \partial$ vecteurs aléatoires dans $\mathbb{K}^N$ que nous nommons $\vec{r}_{\partial+1}, \cdots, \vec{r}_\ell$. De ces vecteurs nous construisons la famille :

$$\mathcal{F} := \left\{ \begin{array}{c} \vec{\delta}_1, A\vec{\delta}_1, \cdots, A^{\beta-1}\vec{\delta}_1, \\ \cdots \\ \vec{\delta}_\partial, A\vec{\delta}_\partial, \cdots, A^{\beta-1}\vec{\delta}_\partial, \\ \vec{r}_{\partial+1}, A\vec{r}_{\partial+1}, \cdots, A^{\beta-1}\vec{r}_{\partial+1}, \\ \cdots \\ \vec{r}_\ell, A\vec{r}_\ell, \cdots, A^{\beta-1}\vec{r}_\ell \end{array} \right\}.$$

Notre première condition est précisément l'hypothèse que la famille $\mathcal{F}$ génère l'espace vectoriel $\mathbb{K}^N$ tout entier. Nous discuterons de la validité de cette hypothèse au paragraphe 6.3.5.

Conditions sur le bloc V.

Afin d'initialiser les ℓ suites à calculer, l'idée principale consiste à forcer les ∂ premières d'entre elles à commencer par les ∂ colonnes denses de M_∂. En d'autres termes, nous posons $\vec{v}_i = \vec{\delta}_i$ pour tout indice i dans l'intervalle $[\![1, d]\!]$ et $\vec{v}_i = \vec{r}_i$ pour tout i dans l'intervalle $[\![\partial+1, \ell]\!]$. Nous remarquons alors que l'hypothèse faite sur la famillle $\mathcal{F}$ peut se réécrire de la façon suivante :

$$\mathrm{Vect}\left(\{A^i\vec{v}_j|\ {}^{i=0,\cdots,\beta-1}_{j=1,\cdots,\ell}\ \}\right) = \mathbb{K}^N. \tag{6.4}$$

Soit ${}^t({}^t\vec{x}|x'_1|\cdots|x'_\partial)$ un vecteur du noyau de la matrice $(A|M_\partial)$. Si l'équation (6.4) est vérifiée, alors il existe en particulier des coefficients $\lambda_{ij} \in \mathbb{K}$ tels que :

$$\vec{x} = \sum_{j=1}^{\ell} \sum_{i=0}^{\beta-1} \lambda_{ij} A^i \vec{v}_j.$$

Aussi nous pouvons en déduire :

$$\begin{aligned}
(A|M_\partial)\,{}^t({}^t\vec{x}|x'_1|\cdots|x'_\partial) = \vec{0} \quad &\Leftrightarrow \\
A\sum_{j=1}^{\ell}\sum_{i=0}^{\beta-1}\lambda_{ij}A^i\vec{v}_j + M_\partial\,{}^t(x'_1|\cdots|x'_\partial) = \vec{0} \quad &\Leftrightarrow \\
\sum_{j=1}^{\partial}\sum_{i=1}^{\beta}\lambda_{(i-1)j}A^i\vec{\delta}_j + \sum_{j=\partial+1}^{\ell}\sum_{i=1}^{\beta}\lambda_{(i-1)j}A^i\vec{r}_j + \sum_{j=1}^{\partial}x'_j\vec{\delta}_j = \vec{0} \quad &\Leftrightarrow \\
\sum_{j=1}^{\ell}\sum_{i=0}^{\beta}a_{ij}A^i\vec{v}_j = \vec{0}
\end{aligned}$$

où les coefficients a_{ij} sont définis via les alternatives suivantes :

$$a_{ij} = \begin{cases} \lambda_{(i-1)j} & \text{si } i > 0. \\ x'_j & \text{si } i = 0 \text{ et } j \leq \partial. \\ 0 & \text{si } i = 0 \text{ et } j > \partial. \end{cases}$$

Pour résumer, aussitôt que les conditions données par l'équation (6.4) sur la matrice $V = (\vec{v}_1|\cdots|\vec{v}_\ell)$ sont satisfaites, chaque élément du noyau de la matrice $(A|M_\partial)$ apporte une solution de :

$$\sum_{j=1}^{\ell}\sum_{i=0}^{\beta}a_{ij}A^i\vec{v}_j = \vec{0}, \tag{6.5}$$

où les coefficients a_{0j} sont nuls dès que $j > \partial$. Réciproquement, (que la condition (6.4) soit vérifiée ou non) n'importe quelle solution de l'équation (6.5) qui comporte des zéros précisément dans ces positions donne un élément du noyau de $(A|M_\partial)$. Par conséquent, en supposant (6.4), déterminer le noyau de la matrice $(A|M_\partial)$ est équivalent à trouver une base de l'ensemble des solutions de l'équation (6.5) en prenant en compte les $\ell - \partial$ zéros sus-mentionnés.

Condition sur le bloc W.

L'équation (6.5) peut se voir bien évidemment comme un système de N équations linéaires sur $\mathbb{K}$. Cependant, résoudre un tel système ne serait pas plus efficace que de calculer directement le noyau de M. Au lieu de cela, nous remarquons que pour chaque matrice $W = (\vec{w}_1|\cdots|\vec{w}_\ell)$ consistant en la concaténation de ℓ vecteurs colonnes de $\mathbb{K}^N$, une solution (a_{ij}) de l'équation (6.5)

entraine une solution de :

$$\sum_{j=1}^{\ell}\sum_{i=0}^{\beta} a_{ij}\,{}^t\vec{w}_k A^{i+\kappa}\vec{v}_j = 0 \tag{6.6}$$

pour tout $k \in [\![1,\ell]\!]$ et tout $\kappa \in \mathbb{N}$.

Réciproquement, supposons que nous connaissons une solution $(a_{ij})_{i\in[\![1,\ell]\!]}^{j\in[\![0,\beta]\!]}$ qui vérifie l'équation (6.6) pour tout $k \in [\![1,\ell]\!]$ et pour tout $\kappa \in [\![0,\beta-1]\!]$. En supposant de plus que :

$$\mathrm{Vect}\Big(\{{}^t\vec{w}_j A^i |\ {}^{i=0,\cdots,\beta-1}_{j=1,\cdots,\ell}\ \}\Big) = \mathbb{K}^N, \tag{6.7}$$

nous pouvons voir que les coefficients a_{ij} forment aussi une solution de l'équation (6.5). En effet, par hypothèse, le vecteur $\sum_{j=1}^{\ell}\sum_{i=0}^{\beta} a_{ij}A^i\vec{v}_j$ est orthogonal à n'importe quel vecteur de la base de $\mathbb{K}^N$ listé par la condition (6.7). Par conséquent, il s'agit donc du vecteur nul.

Réécriture de l'équation (6.6) **à l'aide de matrice de séries formelles.**

Nous regroupons maintenant les ℓ copies de l'équation (6.6) données par les ℓ valeurs de k dans l'intervalle $[\![1,\ell]\!]$. Plus précisément, notons $\vec{a_i}$ le vecteur colonne égal à ${}^t(a_{i1}, a_{i2}, \cdots, a_{il})$. Avec cette notation les ℓ équations peuvent être regroupées facilement en une unique relation :

$$\sum_{i=0}^{\beta} ({}^t W A^{i+\kappa} V)\cdot\vec{a_i} = \vec{0}. \tag{6.8}$$

Définissons maintenant la série formelle de matrice $S(X)$ et le polynôme vectoriel $P(X)$ comme :

$$S(X) = \sum_{i\in\mathbb{N}} ({}^t W A^i V)X^i \quad \text{et} \quad P(X) = \sum_{i=0}^{\beta} \vec{a_i} X^{B-i}.$$

Remarque 6.3.2. Il existe une bijection canonique ϕ entre les séries formelles de matrices (resp. les polynômes vectoriels) et les matrices de séries formelles (resp. les vecteurs de polynômes). Ainsi, si $S(X) = \sum_{i\in\mathbb{N}}(M_i)X^i$ est une série formelle de matrices, alors $\phi(S(X)) = \left(\sum_{i\in\mathbb{N}} m^i_{j,k}X^i\right)_{j,k}$ avec $M_i = \left(m^i_{j,k}\right)_{j,k}$ est la matrice de séries formelles associée, et réciproquement.

Nous pouvons ainsi multiplier $S(X)$ par $P(X)$. Par définition de la multiplication des séries formelles, nous observons que le coefficient correspondant au monôme $X^{\beta+\kappa}$ dans le produit $S(X)P(X)$ est exactement $\sum_{i=0}^{\beta}({}^tWA^{i+\kappa}V)\cdot\vec{a_i}$. D'après l'équation (6.8), il s'agit donc de $\vec{0}$ pour tout $\kappa \in \mathbb{N}$.

Par conséquent, la série formelle de vecteurs donné par $S(X)P(X)$ est en réalité un polynôme vectoriel $Q(X)$ de degré au plus $\beta-1$. Ainsi, étant donné $S(X)$ nous cherchons des polynômes vectoriels $P(X)$ et $Q(X)$ de degré respectif au plus β et $\beta-1$ tels que $S(X)P(X)-Q(X)=\vec{0}$. Pour mieux correspondre aux notations de la section 6.2.4 précédente, nous définissons $G(X)=(\phi(S(X))|-\mathrm{Id}(X))$ comme la matrice de taille $\ell\times 2\ell$ de séries formelles construite par la concaténation de l'opposé de la matrice identité de taille $\ell\times\ell$ avec la matrice $\phi(S(X))$ associée à la série formelle $S(X)$. Notons par ailleurs $\vec{u}(X)$ le vecteur ligne de dimension 2ℓ obtenue par concaténation du vecteur ${}^t\phi(P(X))$ et de ${}^t\phi(Q(X))$. Nous avons alors l'égalité $G(X){}^t\vec{u}=\vec{0}$, que nous pouvons transposer et réécrire :

$$\vec{u}(X)\cdot{}^tG(X)=\vec{0}. \tag{6.9}$$

Il est important de souligner que $\vec{u}(X)$ présente un degré au plus β sur ses ℓ premières coordonnées, et un degré au plus $\beta-1$ sur toutes les suivantes. De plus, l'annulation de tous les coefficients a_{0j} pour tout $j>\partial$ entraine un coefficient (constant) nul pour toutes les coordonnées polynomiales entre $\partial+1$ et ℓ.

Comme nous avons à l'esprit d'appliquer l'algorithme de Giorgi, Jeannerod et Villard, nous allons nous restreindre à l'étude des relations modulo un monôme de haut degré, plutôt que de conserver des séries formelles. En effet, nous avons clairement $\vec{u}(X)\cdot{}^tG(X)=\vec{0}$ modulo X^β pour tout entier β, avec les trois contraintes sur le vecteur de polynômes $\vec{u}(X)$ qui restent identiques. Nous analysons la valeur de β qui permet d'affirmer qu'une solution de l'équation (6.9) modulo X^β, avec les mêmes contraintes sur le vecteur $\vec{u}$, peut être transformée en retour en une solution de l'équation (6.6) pour tout entier κ dans $[\![0,\beta-1]\!]$.

Supposons que nous ayons un vecteur $\vec{u}={}^t(u^{(1)},\cdots,u^{(2\ell)})$ solution de l'équation (6.9) modulo X^β avec :

— $\forall i\in[\![1,\ell]\!]$, $\deg u^{(i)}(X)\le\beta$,
— $\forall i\in[\![\ell+1,2\ell]\!]$, $\deg u^{(i)}(X)\le\beta-1$,
— $\forall i\in[\![\partial+1,\ell]\!]$, $u^{(i)}(0)=0$.

Puisque $\vec{u}$ consiste en 2ℓ polynômes nous pouvons le couper virtuellement en deux parties distinctes de même longueur et considérer dans un premier temps ses ℓ premiers termes polynomiaux, qui sont de degré au plus β. Il

existe une correspondance canonique entre ce vecteur $\vec{v}$ de polynômes et un polynôme $\phi(\vec{v}) = P(X)$ de degré au plus β dont les coefficients sont des vecteurs de $\mathbb{K}^\ell$. En posant $P(X) = \sum_{i=0}^{\beta} \vec{z}_i X^i$ avec $\vec{z}_i \in \mathbb{K}^\ell$ nous pouvons définir pour tout entier i dans l'intervalle $[\![0, \beta]\!]$ et tout entier j dans $[\![1, \ell]\!]$:

$$a_{ij} = \text{la } j\text{-ème coordonnée du vecteur } \vec{z}_{\beta-i}.$$

Puisque modulo le monôme X^β le produit $P(X)S(X)$ est un vecteur de polynômes de degré au plus $\beta - 1$ nous déduisons que pour tout entier κ tel que $0 \leq \kappa \leq b - \beta - 1$ le coefficient de $P(X)S(X)$ associé au monôme $X^{\beta+\kappa}$ est le vecteur nul. Combiné avec $P(X) = \sum_{i=0}^{\beta} \vec{a_i} X^{\beta-i}$ et $S(X) = \sum_{i\in\mathbb{N}} ({}^t W A^i V) X^i$ il en découle l'égalité vectorielle $\sum_{i=0}^{\beta} ({}^t W A^{i+\kappa} V) \cdot \vec{a_i} = \vec{0}$. Aussi, en multipliant la matrice V par le vecteur $\vec{a_i}$ nous obtenons $\sum_{j=1}^{\ell} \sum_{i=0}^{\beta} {}^t W A^{i+\kappa} (a_{ij} \vec{v}_j) = \vec{0}$. Finalement, en regardant chacune des lignes ${}^t\vec{w}_k$ avec k dans l'intervalle $[\![1, \ell]\!]$ et pour tout entier κ dans l'intervalle $[\![0, b - \beta - 1]\!]$ nous savons déduire des coefficients a_{ij} qui sont solutions de l'équation (6.6). Par conséquent, pour obtenir l'équation (6.6) pour tout κ dans l'intervalle $[\![0, \beta - 1]\!]$ il suffit de poser :

$$b = 2\beta.$$

Feuille de route des transformations

Reprenons les différentes étapes de notre processus. Nous avons transformé le problème d'expliciter le noyau de M en celui de trouver toutes les solutions de l'équation (6.9) modulo $X^{2\beta}$, de degré au plus β sur leur ℓ premières coordonnées, de degré au plus $\beta - 1$ sur les suivantes, et dotées d'un coefficient constant nul pour toutes les coordonnées entre $\partial + 1$ et ℓ. Sous les deux conditions (6.4) et (6.7), l'analyse précédente explicite directement une bijection entre l'ensemble des solutions des deux problèmes, comme l'illustre la figure 6.2.

6.3.4 Application de l'algorithme de Giorgi, Jeannerod et Villard.

Grâce à Giorgi, Jeannerod et Villard [GJV03] nous pouvons calculer une σ-base des vecteurs solutions de l'équation (6.9) modulo $X^{2\beta}$ en temps $\tilde{O}(\ell^\omega \beta)$. Cependant, il est nécessaire de travailler a posteriori sur cette σ-base pour retrouver une base du noyau de M. Plus précisément, nous souhaitons obtenir une description explicite de tous les vecteurs solutions de l'équation (6.9) qui sont de degré au plus β sur leurs ℓ premières coordonnées, de degré au plus $\beta - 1$ sur les ℓ dernières, et tels que les coordonnées indicées entre

$\partial+1$ et ℓ présentent un coefficient constant qui soit nul. Nous montrons dans un premier temps comment obtenir tous les vecteurs solutions qui sont de degré au plus β sur les 2ℓ coordonnées. Au final, une étape de filtrage permet d'assurer que la borne plus forte sur le degré des dernières coordonnées est bien respectée, et que les $\ell-\partial$ coefficients constants intéressés sont bel et bien nuls.

Solutions de degrés bornés

Notons dans un premier temps $\vec{b_1},\cdots,\vec{b_t}$ les t vecteurs[2] de la σ-base de degré au plus β. Si $\vec{u}$ désigne n'importe quel vecteur solution de l'équation (6.9) de degré au plus β, alors, du caractère minimal de la σ-base, nous savons que $\vec{u}$ peut s'écrire comme une combinaison linéaire $\sum_{i=1}^{t} c_i\vec{b}_i$ où les c_i sont des polynômes de $\mathbb{K}[X]$ tels que $\deg c_i + \deg b_i \leq \deg\vec{u}$ pour tout $i\in[\![1,t]\!]$. Ainsi, l'ensemble des solutions de l'équation (6.9) de degré au plus β est engendré par la famille :

$$\mathcal{E} := \bigcup_{i=1}^{t}\{\vec{b}_i, X\vec{b}_i, X^2\vec{b}_i,\cdots,X^{\beta-\deg\vec{b}_i}\vec{b}_i\}.$$

Cette famille est libre et, puisque génératrice, il s'agit donc d'une base du sous-espace des solutions de l'équation (6.9). En effet, les t vecteurs $\vec{b_i}$ appartiennent à une même σ-base et, par conséquent, sont linéairement indépendants. En outre, la multiplication par X induit une structure sur la matrice représentant $\mathcal{E}$, qui se retrouve alors être diagonale par blocs. En raison de cette structure, tous les vecteurs de $\mathcal{E}$ sont eux-aussi linéairement indépendants.

Filtrage

Pour obtenir une base du noyau de M, nous devons maintenant opérer une étape de filtrage, le but étant de s'assurer que l'on ne garde que les vecteurs de $\mathcal{E}$ de degrés inférieurs à $\beta-1$ sur les ℓ dernières coordonnées, et de coefficients constants nuls en positions $\partial+1$ à ℓ. De manière amusante, la première propriété tient pour tout vecteur de $\mathcal{E}$ hormis le dernier multiple de chaque $\vec{b_i}$, c'est-à-dire hormis $X^{\beta-\deg\vec{b}_i}\vec{b_i}$. De même, la seconde propriété est

2. Au pire, il y a 2ℓ tels vecteurs. Remarquons qu'en pratique, il est agréable d'éxecuter l'algorithme pour trouver une σ-base sur des séries formelles avec une précision légèrement plus grande que 2β, afin d'avoir moins de vecteurs à ce stade de l'algorithme (souvent de l'ordre de ℓ).

déjà satisfaite par tous les multiples de $X^j\vec{b_i}$ pour $j \neq 0$. Ainsi, pour tout vecteur $\vec{b_i}$, tous les multiples à l'exception (éventuelle) du premier et du dernier vérifient ces deux conditions supplémentaires. En outre, certaines combinaisons linéaires de ces premiers et derniers multiples peuvent satisfaire toutes ces conditions, tandis que d'autres en violeront au moins une. Il est cependant facile de construire les combinaisons qui fonctionnent : en effet, ces conditions sont linéaires et n'impliquent que les coefficients apparaissant sur $2\ell - \partial$ positions. Aussi, pour identifier ces combinaisons, il suffit d'extraire les coefficients pertinents des multiples engendrés qui ne satisfont pas les conditions, et de les assembler en une matrice de dimension $2\ell - \partial$ par, au plus [3], $2t$. Le noyau de la matrice ainsi décrite donne les combinaisons désirées. Il peut être explicité asymptotiquement en $O(\ell^\omega)$ opérations. En pratique, il s'agira plutôt de $O(\ell^3)$ opérations, particulièrement pour les petites valeurs de ℓ. Dorénavant, nous appelons $\vec{b'_i}$ les t' combinaisons ainsi produites.

Nous déduisons de tout ce qui précède que la base de solutions de l'équation (6.9) qui satisfait de plus les trois conditions est donnée par :

$$\mathcal{U} := \{\vec{b'_1}, \cdots, \vec{b'_{t'}}\} \cup \bigcup_{i=1}^{t} \{X\vec{b}_i, X^2\vec{b}_i, \cdots, X^{\beta-1-\deg \vec{b}_i}\vec{b}_i\}.$$

Il est possible de représenter cette base sous une forme compacte, en écrivant simplement $t + t'$ vecteurs, avec $t' \leq 2t$. Ceci apporte précisément la base des solutions de l'équation encadrée dans la figure 6.2.

Enfin, l'algorithme 3 résume en pseudo-code les étapes principales par lesquelles nous passons pour retrouver le noyau d'une matrice presque creuse M, qui a été au préalable transformée en une matrice composée de la concaténation d'une matrice carrée A et de la partie dense M_∂ de M.

6.3.5 Rôle des blocs V et W et vérification automatique

Vérification de la condition (6.4) **sur le bloc** V.

Un avantage de la méthode que nous proposons réside dans le fait que celle-ci permet dans un même temps de vérifier la validité de l'hypothèse faite sur V, c'est-à-dire de la condition (6.4). En effet, en revenant à la famille $\mathcal{F}$ nous pouvons observer qu'elle consiste en $\ell\beta$ vecteurs de $\mathbb{K}^N$. La matrice correspondant à la famille $\mathcal{F}$ est de rang plein si et seulement si la dimension de son noyau est précisément $\ell\beta - N$. Par ailleurs un élément du noyau est

3. En effet, les vecteurs $\vec{b_i}$ de degré exactement β apparaissent une unique fois dans la matrice, tandis que les autres apparaissent deux fois.

exactement une famille de coefficients (a_{ij}) tels que $\sum_{j=1}^{\ell}\sum_{i=0}^{\beta-1} a_{ij}A^i v_j = 0$. La différence avec l'équation (6.5) se situe dans les termes de la somme, qui

Algorithm 3 Algorithme presque creux pour la résolution de $(A|M_\partial)\vec{x} = \vec{0}$

Entrée : Une matrice A de taille $N \times N$ et une matrice $M_\partial = (\vec{\delta}_1|\cdots|\vec{\delta}_\partial)$ de taille $N \times \partial$
Sortie : Une base de $\mathrm{Ker}(A|M_\partial)$.

Calcul d'une suite de matrices

$\vec{r}_{\partial+1} \leftarrow\in \mathbb{K}^N, \cdots, \vec{r}_\ell \leftarrow\in \mathbb{K}^N$ et $\vec{w}_1 \leftarrow\in \mathbb{K}^N, \cdots, \vec{w}_\ell \leftarrow\in \mathbb{K}^N$
$\vec{v}_1 \leftarrow \vec{\delta}_1, \cdots, \vec{v}_\partial \leftarrow \vec{\delta}_\partial$
$\vec{v}_{\partial+1} \leftarrow \vec{r}_{\partial+1}, \cdots, \vec{v}_\ell \leftarrow \vec{r}_\ell$
$\beta \leftarrow \lceil N/\ell \rceil$
for chacun des ℓ processeurs indicés par j **do**
  $u_0 \leftarrow v_j$
  for $i = 0, \cdots, 2\beta$ **do**
    for $k = 1, \cdots, \ell$ **do**
      $\lambda_{i,j,k} \leftarrow \vec{w}_k \cdot \vec{u}_i$
    end for
    $\vec{u}_{i+1} \leftarrow A\vec{u}_i$
  end for
end for
for $i = 0, \cdots, 2\beta$ **do**
  $M_i \leftarrow (\lambda_{i,j,k})$ la matrice de taille $\ell \times \ell$ contenant tous les produits de la forme ${}^t\vec{w}A^i\vec{v}$
end for

Application de l'algorithme de Giorgi, Jeannerod, et Villard

$S \leftarrow \sum_{i=0}^{2\beta-1} M_i X^i$.
Retrouver une σ–base de la matrice ${}^t(S|-\mathrm{Id})$ modulo $X^{2\beta}$.
$\vec{b}_1, \cdots, \vec{b}_t \leftarrow$ les vecteurs de cette σ–base de degré inférieur à β.
$\vec{b'}_1, \cdots, \vec{b'}_{t'} \leftarrow$ une base des combinaisons linéaires de $\vec{b}_1$, $\cdots$, $\vec{b}_t$, $X^{B-\deg b_1}\vec{b}_1$, $\cdots$, $X^{\beta-\deg b_1}\vec{b}_t$ telle que les ℓ dernières coordonnées soient de degré inférieur à $\beta-1$ et les coordonnées entre $\partial+1$ et ℓ soient divisible par X.
$U \leftarrow [\vec{b'}_1, \cdots, \vec{b'}_{t'}, X\vec{b}_1, \cdots, X^{\beta-1-\deg b_1}\vec{b}_1, \cdots, X\vec{b}_t, \cdots, X^{\beta-1-\deg b_t}\vec{b}_t]$

Résolution

Sol $\leftarrow$ [ ]
for $\vec{u} \in U$ **do**
  for $i = 0, \ldots, \beta$ **do**
    for $j = 1, \ldots, \ell$ **do**
      $a_{ij} \leftarrow$ le coefficient associé au monôme $X^{\beta-i}$ dans le polynôme qui est le j-ème coefficient de $\vec{u}$.
    end for
  end for
  $\vec{x} \leftarrow {}^t({}^t(\sum_{j=1}^{\ell}\sum_{i=0}^{\beta-1} a_{(i+1)j}A^i\vec{v}_j)|a_{01}|\cdots|a_{0\partial})$
  Add $\vec{x}$ to Sol
end for
return Sol

s'achève ici à $\beta-1$ et non à β, ainsi que dans les coefficients a_{0j} pour lesquels rien n'est indiqué. En suivant le chemin donné par la figure 6.2, nous pouvons en déduire une bijection entre le noyau de cette matrice et l'ensemble des solutions de l'équation (6.9) qui sont de degré $\beta-1$ sur leurs ℓ premières coordonnées, et $B-2$ sur les ℓ dernières. Puisque nous avons déjà calculé un ensemble de solutions plus large de l'équation (6.9), nous pouvons vérifier si la dimension de l'ensemble réduit est bien $\ell\beta - N$. Si ce n'est pas le cas, les éléments du noyau de M qui ont été obtenus restent valident, mais la base du noyau peut être incomplète.

V pour la complétion, W pour la correction

Si le choix du bloc V détermine si la base du noyau sera complète ou non, comme nous venons de le voir, celui du bloc W indique en revanche si toutes les solutions sont correctes ou non. Nous ne prouvons pas ici de méthode automatique pour vérifier la condition (6.7) au cours de l'algorithme. Cependant, il faut garder à l'esprit que ceci ne constitue par un souci important lors de l'application de l'algorithme. En effet, il est toujours aisé de vérifier si une solution donnée à la fin de l'exécution renvoie bel et bien un élément du noyau de la matrice cherchée – qui peut ne pas être le cas si la condition sur W n'est pas satisfaite, mais alors cette erreur sera rapidement détectée.

6.3.6 Quelques exigences concernant les paramètres

L'algorithme décrit, il nous faut maintenant discuter du choix des valeurs pour les paramètres β (qui borne le degré des polynômes) et ℓ (qui indique le nombre de suites à calculer) en fonction des paramètres d'entrée qui sont fixes, c'est-à-dire en fonction de N, s et ∂. Par construction, nous savons déjà que $\ell \geq \partial$. Cependant, pour que les conditions (6.4) et (6.7) puissent être vérifiées, des restrictions additionnelles viennent s'ajouter. En particulier, la condition (6.7) exige une famille constituée de $\ell\beta$ vecteurs de rang N. Aussi, il est nécessaire d'avoir l'inégalité :

$$\beta \geq \left\lceil \frac{N}{\ell} \right\rceil.$$

D'autres prérequis au fonctionnement de l'algorithme se trouvent plus implicites. Par exemple, en regardant de nouveau la condition (6.7), nous remarquons que tous les vecteurs de $\left\{ {}^t\vec{w}_j A^i \middle| \begin{smallmatrix} i=1,\cdots,\beta-1 \\ j=1,\cdots,\ell \end{smallmatrix} \right\}$ appartiennent à l'image de la matrice A. Conséquence de quoi, la dimension de l'espace vectoriel de la condition (6.7) est majorée par $\mathrm{rg}(A) + \ell$. En outre, grâce à la pré-transformation détaillée au paragraphe 6.3.2, nous savons que le rang de A

est au plus s. De ceci découle l'exigence nouvelle :

$$\ell \geq \max(N - s, \partial).$$

Que pouvons-nous dire lorsque les vecteurs de W sont choisis aléatoirement – comme c'était le cas jusqu'ici ? Pour un corps suffisamment gros, la dimension de l'espace vectoriel de la condition (6.7), pour des vecteurs $\vec{w}_i$ aléatoires, est $\mathrm{rg}(A) + \ell$ avec probabilité proche de 1.

Les exigences associées à la condition (6.4) ne donnent pas de contraintes arithmétiques plus fortes sur les paramètres ℓ et β. En revanche, la famille de vecteurs qui apparait dans la condition (6.4) contient des vecteurs préalablement fixés (ceux qui sont dérivés des colonnes denses de M) ; par conséquent, nous ne pouvons pas affirmer que la condition associée tient pour tout choix aléatoire du reste des vecteurs de V. La vérification qui permet d'être opérée par l'algorithme, et détaillée au paragraphe 6.3.5, rend heureusement cet inconvénient mineur.

6.3.7 Analyse de complexité

Le coût total de notre méthode se scinde en deux parties. D'une part, nous avons la complexité des produits matrice-vecteur dont le coût séquentiel s'élève à $O(\lambda N^2)$, qui inclut la préparation des suites de matrices $\ell \times \ell$ et le calcul final de la base du noyau. Ceci peut être facilement distribué sur plusieurs processeurs, en particulier lorsque le nombre ℓ de suites est égal au nombre c de processeurs, ou à un multiple de celui-ci. Cette opération réduit le temps d'horloge à $O(\lambda N^2/c)$. Nous renvoyons à [Kal95] pour réduire encore le coût de la dernière phase d'évaluation. Par ailleurs, puisque $\beta \approx N/\ell$, l'étape qui retrouve les coefficients a_{ij} a une complexité en $\tilde{O}(\ell^{\omega-1}N)$, donnée par l'utilisation de l'algorithme de Giorgi, Jeannerod et Villard. L'étape de filtrage à la fin de cette exécution coûte $O(\ell^w)$ et peut donc être négligée – puisque nous avons très clairement $\ell \leq N$.

Pour minimiser le coût de l'algorithme de Giorgi, Jeannerod et Villard, nous ajustons ℓ pour qu'il soit égal au plus petit multiple de c plus grand que ∂. Dans ce cas, le coût séquentiel total de l'exécution de l'algorithme devient :

$$O(\lambda N^2) + \tilde{O}(\max(c, \partial)^{\omega-1}N).$$

Ceci est à comparer avec la complexité précédente en $O((\lambda+\partial)N^2)+\tilde{O}(c^{\omega-1}N)$ que l'on pouvait obtenir en combinant l'algorithme de Block Wiedemann avec la variante de Giorgi, Jeannerod et Villard pour résoudre le même problème. Nous insistons sur le fait que le temps d'horloge décroit du même coup de $O((\lambda + \partial)N^2/c) + \tilde{O}(c^{\omega-1}N)$ à $O(\lambda N^2/c) + \tilde{O}(\max(c, \partial)^{\omega-1}N)$.

Si $\partial \leq c$ alors la complexité de la variante que nous proposons est en $O(\lambda N^2) + \tilde{O}(c^{\omega-1}N)$, ce qui est exactement la complexité obtenue en combinant l'algorithme de Block Wiedemann avec la variante de Giorgi, Jeannerod et Villard pour résoudre cette fois un problème similaire sur une matrice (entièrement !) creuse de même taille. Dit autrement, *lorsque nous parallélisons sur c processeurs, il est possible de traiter jusqu'à c colonnes denses sans aucun frais supplémentaire.*

6.3.8 Quelle limite de densité pour nos matrices presque creuses ?

Nous aimerions clarifier le terme *presque* dans cette expression que nous employons depuis le début de ce chapitre. Jusqu'à quel point une matrice à la fois creuse et dense par colonnes peut prétendre à ce qualificatif ? Dit autrement, combien de colonnes denses pouvons-nous nous autoriser sans sur-coût ? Nous savons déjà que notre algorithme presque creux se comporte mieux pour ces types de matrices que l'adaptation directe d'une méthode (simplement) creuse. Lorsque le nombre ∂ de colonnes denses croit, il devient en revanche plus pertinent de comparer notre algorithme avec les méthodes classiques utilisées dans le cadre de matrices quelconques (donc denses), c'est-à-dire de comparer notre complexité à $O(N^\omega)$. Dans ce cas de figure, nous supposons donc que le nombre de processeurs à disposition devient plus petit que celui de colonnes denses, et nous remplaçons donc $\max(c, \partial)$ par ∂ dans les formules de complexités.

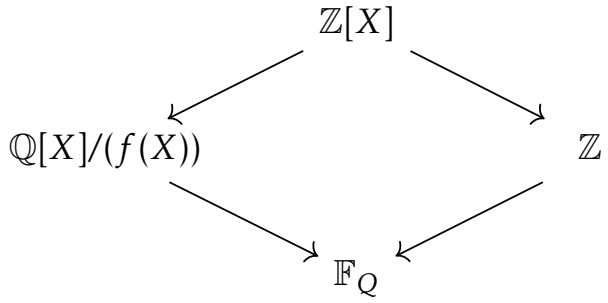
Supposons que $\partial \leq N^{1-\epsilon}$ avec $\epsilon > 0$, alors notre complexité devient $\tilde{O}(\partial^{\omega-1}N) = O(N^{\omega-\epsilon(\omega-1)}(\log N)^\alpha)$ pour un certain $\alpha > 0$, ce qui est asymptotiquement plus bas que $O(N^\omega)$. Cependant, lorsque la matrice est presque tout à fait dense, c'est-à-dire lorsque $\partial = \Omega(N)$, notre technique devient moins efficace, d'un facteur logarithmique, que les méthodes classiques d'algèbre linéaire dense.

6.4 Application au calcul de logarithme discret dans les corps de moyenne et grande caractéristiques

En pratique, ce résultat s'applique précisément au problème du logarithme discret, lorsqu'il est fait usage du crible par corps de nombres (NFS). En effet, les calculs de logarithmes dans les corps finis $\mathbb{F}_Q$ de moyenne et grande caractéristiques font naturellement apparaitre des matrices presque creuses.

6.4.1 Applications de Schirokauer et colonnes lourdes

En deux mots, nous rappelons que NFS construit dans une phrase préliminaire un diagramme de la forme :



Nous ne parlons pas ici de la généralisation avec un corps de nombres de chaque côté du diagramme, mais simplement, pour la simplicité de ce paragraphe, du cas où l'un des côtés est rationnel. Dans une seconde phase NFS crée des relations multiplicatives entre images, dans $\mathbb{F}_Q$, de produits d'idéaux de petites normes dans le corps de nombres $\mathbb{Q}[X]/(f(X))$, et de produits de petits entiers. Ces relations sont ensuite transformées en des relations linéaires entre des logarithmes virtuels d'idéaux, et des logarithmes d'entiers modulo l'ordre multiplicatif de $\mathbb{F}_Q^*$. La descente de ces relations nécessite en réalité de surmonter des obstructions techniques dont les méthodes de contournement demeurent fastidieuses. En pratique, chaque relation est complétée d'un petit nombre d'inconnues supplémentaires dont les coefficients sont calculés grâce aux applications dites de Schirokauer [Sch93]. En un mot : ces applications représentent la contribution des unités du corps de nombres dans les équations. Chacune de ces applications introduit une colonne dense dans le système d'équations linéaires associé. Le nombre total de colonnes ainsi créées est majoré par le degré de f – ou par la somme des degrés dans le cas de figure où deux corps de nombres apparaissent dans le diagramme. Asymptotiquement, nous savons que dans le domaine d'application de NFS, le degré des polynômes qui définissent les corps de nombres est d'au plus $O((\log Q/\log\log Q)^{2/3})$. Ceci reste négligeable par rapport à la taille du système linéaire produit, qui est lui de l'ordre de $L_Q(1/3) = \exp(O((\log Q)^{1/3}(\log\log Q)^{2/3}))$.

Dans le cadre d'un crible multiple cette remarque reste valable, car la matrice des relations comporte les colonnes denses issues de chacun des corps de nombres du diagramme.

6.4.2 Un cas d'application optimal

La troisième étape consiste alors en la résolution du système linéaire précédemment créé, et c'est ici que notre algorithme intervient précisément. Mieux, il s'agit d'un cas optimal pour l'application de notre méthode. En effet, le nombre de colonnes denses est suffisamment petit pour être plus petit que le nombre de processeurs que l'on peut s'attendre à utiliser pour un tel calcul. Prenons un exemple concret pour illustrer notre propos. Dans le record précédent de calcul de logarithme annoncé en Juin 2014 sur un corps premier $\mathbb{F}_p$, où p est un nombre premier de 180 décimales [4], [BGI+14] utilisent un corps de nombres de degré 5. Le nombre d'applications de Schirokauer – et donc de colonnes denses – est de 4 tandis que les auteurs du record disposent de 12 processeurs. Nous nous situons bien dans le cas de figure où le coût de traitement des colonnes denses est ainsi invisible. Par ailleurs, ceci est à comparer avec les $7,28$ millions de colonnes et de lignes de la matrice des relations, qui comporte en moyenne 150 coefficients non nuls par lignes. Ces colonnes denses donnent ainsi à la matrice cette structure presque creuse que nous avons étudiée.

Conclusion de notre algorithme presque creux dans le cadre du problème du logarithme discret.

Notre adaptation de l'algorithme de Block Wiedemann chasse entièrement la difficulté qui existait lors du traitement de ces colonnes denses, dans le contexte du logarithme discret. Nous soulignons qu'il s'agissait bel et bien d'un souci pratique avéré, en témoignent les tentatives d'approches observées et portant aussi sur la volonté de décroitre le coût associé au traitement de ces colonnes particulières. Par exemple, dans [BGGM14], la construction du diagramme commutatif est remplacée par une méthode sophistiquée basée sur les automorphismes pour réduire le nombre d'applications nécessaires lors du calcul. En ce qui concerne les colonnes denses, cette approche est maintenant périmée.

De plus, comme nous avons généralement plus de processeurs en pratique, il est possible de considérer les colonnes correspondants à des très petits entiers ou à des idéaux de très petites normes (qui apparaissent donc souvent) comme faisant partie de la partie dense de la matrice, pour réduire encore un peu, en pratique, le coût de l'algèbre linéaire.

4. Encore une fois, témoignage de l'activité du domaine, ce précédent record a été battu par un calcul de logarithme discret sur un corps premier de 768 bits durant la rédaction de ce manuscrit, comme en témoigne le tableau 3.1.

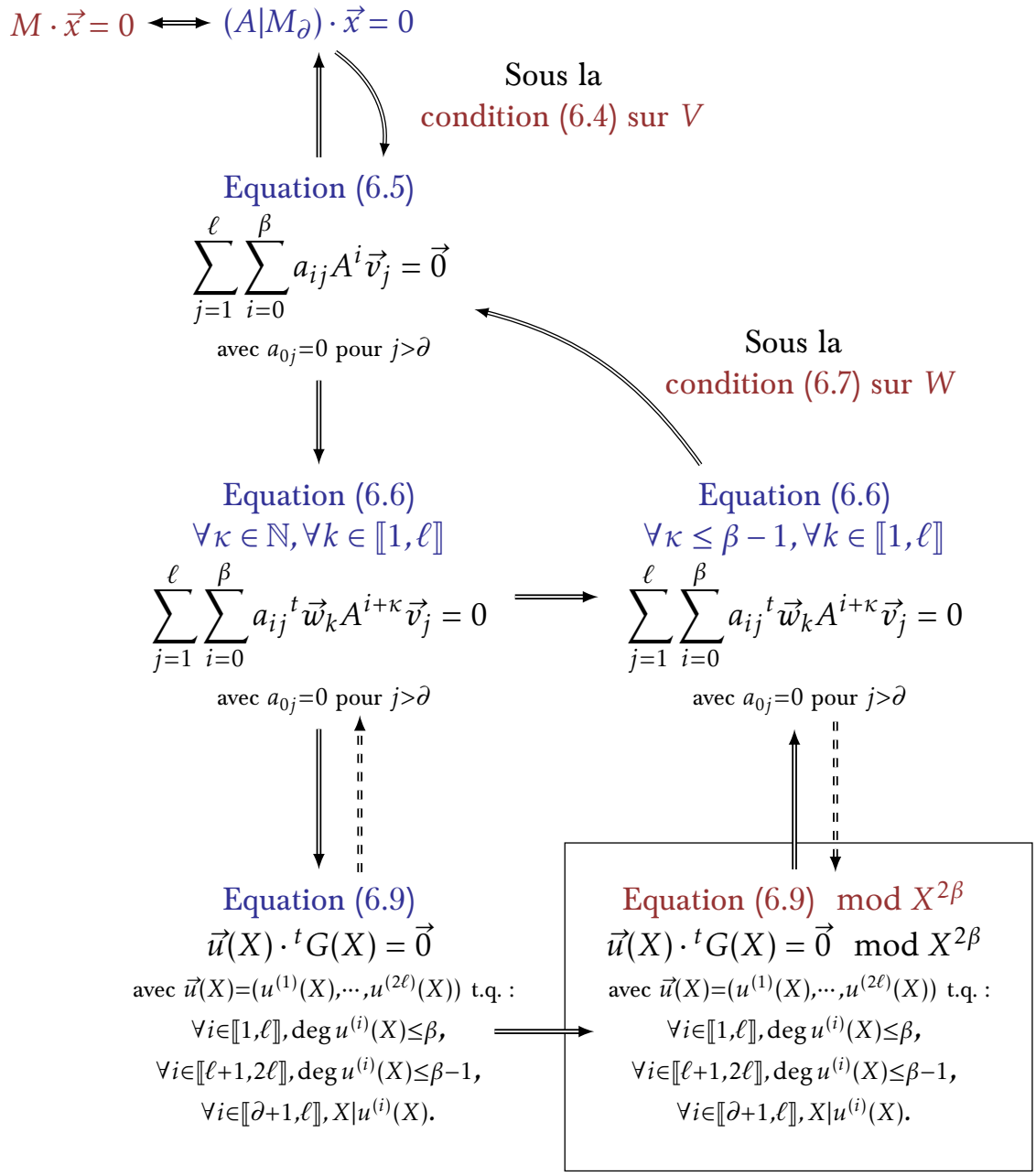


Figure 6.2 – *Comment le calcul du noyau d'une matrice presque creuse M est réduit au calcul du noyau d'une matrice G de séries formelles.* Les équivalences et les implications entre les différents problèmes sont à lire selon les indications qui suivent : $A \Rightarrow B$ signifie qu'une solution de l'équation A peut être transformée en une solution de l'équation β. Les inconnues sont représentées par x, a_{ij} ou $\vec{u}$ tandis que les autres variables $M, \partial, M_\partial, A, \ell, \beta, V = \{\vec{v}_1, \cdots, \vec{v}_\ell\}, W = \{\vec{w}_1, \cdots, \vec{w}_\ell\}$ et G sont supposées connues. Nous avertissons le lecteur que, bien que ce soit vrai, nous ne démontrons pas l'implication Equation (6.9) $\Rightarrow$ Equation (6.6) dans ce manuscrit, puisque toutes les autres implications sont déjà suffisantes pour conclure comme nous le souhaitons. Une remarque similaire peut être faite sur la réduction : Equation (6.6) pour tout $\kappa \leq \beta - 1$ $\Rightarrow$ Equation (6.9) $\mod X^{2\beta}$.

Et demain ?

À ce jour, le statu du problème du logarithme discret évolue encore rapidement, aussi, tenter de prédire les avancées futures dans ce domaine reste un pari complexe. Nous ne donnons ici qu'un rappel des problèmes ouverts, à lire comme une liste des progrès qui verront peut être le jour dans les années à venir, mais sans certitude aucune de ce qui sera découvert, de l'échelle de temps qui entrera en jeu, ni de l'ordre éventuel de telles avancées.

La menace la plus dangereuse et la plus concrète pour le problème du logarithme discret en toute généralité, et qui concerne tout aussi bien la factorisation des entiers, correspond à la possibilité de l'émergence d'ordinateurs quantiques capables de calculs de grandes envergures. Si de telles machines voyaient le jour, l'algorithme de Shor briserait totalement ces deux problèmes difficiles. Cependant, la création et la disponibilité même dans un futur lointain de telles machines restent un sujet délicat voire opaque, pour lequel beaucoup de chercheurs refusent de s'enthousiasmer aujourd'hui.

A propos du problème du logarithme discret dans les corps finis, plusieurs voies sont ouvertes à de nouvelles découvertes. En petite caractéristique tout d'abord, les variations autour de l'algorithme quasi-polynomial pourraient être améliorées de trois façon, soit que l'on abaisse la valeur de l'exposant dans la partie polynomiale de l'algorithme, que l'on découvre un algorithme entièrement polynomial, ou que l'on construise un algorithme rigoureux, qui s'affranchirait des hypothèses heuristiques résiduelles.

Si la caractéristique est suffisamment grande, ces méthodes ne peuvent s'appliquer directement. Elles s'appuient par ailleurs sur des propriétés spécifiques de polynômes qui ne semblent pas aisément s'adapter aux entiers : c'est par exemple le cas de la relation systématique $\prod_{\alpha \in \mathbb{F}_q}(X-\alpha) = X^q - X$ qui n'a pas d'homologue connu pour la factorisation systématique des nombres. Sans prédire de juste chemin pour y parvenir, la complexité en $L(1/3)$ ne pa-

rait pourtant plus être une barrière naturelle, et nous pourrions espérer des progrès pour le crible par corps de nombres en ce sens. Du même fait il existe alors un espoir d'améliorer la complexité de la factorisation.

Le défit le plus incertain réside probablement dans la recherche d'un algorithme sous-exponentiel qui s'appliquerait aux courbes elliptiques générales définies sur des corps de grandes caractéristiques. La question est notamment discutée dans [Mas14]. Cela étant, la découverte de nouveaux algorithmes de calcul d'indice qui ne recouvreraient que le cas de courbes particulières concrétise déjà un problème fascinant et tout à fait ambitieux de notre domaine de recherche.

Bibliographie

[ACMCC+16] Gora Adj, Isaac Canales-Martinez, Nareli Cruz-Cortés, Alfred Menezes, Thomaz Olivieira, Francisco Rodriguez-Henriquez, and Luis Rivera-Zamarripa. Discrete logarithm computation in $\mathbb{F}_{3^{6*509}}$. *Announcement to the NMBRTHRY list*, July 2016.

[Adl79] Leonard M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography (abstract). In *FOCS*, pages 55–60, 1979.

[AFK89] Martín Abadi, Joan Feigenbaum, and Joe Kilian. On hiding information from an oracle. *J. Comput. Syst. Sci.*, 39(1) :21–50, 1989.

[AH99] Leonard M. Adleman and Ming-Deh A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inf. Comput.*, 151(1-2) :5–16, 1999.

[AMOR14] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodríguez-Henríquez. Computing discrete logarithms in $\mathbb{F}_{3^{6\cdot137}}$ and $\mathbb{F}_{3^{6\cdot163}}$ using Magma. In *Arithmetic of Finite Fields : WAIFI'2014*, pages 3–22, 2014.

[AMORH13] Gora Adj, Alfred Menezes, Thomaz Oliveira, and Francisco Rodriguez-Henriquez. Weakness of $\mathbb{F}_{3^{6\cdot1429}}$ and $\mathbb{F}_{2^{4\cdot3041}}$ for discrete logarithm cryptography. Cryptology ePrint Archive, Report 2013/737, 2013.

[ANS14] ANSSI. Agence nationale de la sécurité des systèmes d'information : Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques. Référentiel Général de Sécurité version 2.0, Annexe B1, 2014.

[BBD+14] Razvan Barbulescu, Cyril Bouvier, Jérémie Detrey, Pierrick Gaudry, Hamza Jeljeli, Emmanuel Thomé, Marion Videau, and Paul Zimmermann. Discrete logarithm in GF(2809) with FFS. In

Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings, pages 221–238, 2014.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.

[BD94] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 275–286, 1994.

[Ber68] Elwyn R. Berlekamp. Nonbinary BCH decoding (abstr.). *IEEE Transactions on Information Theory*, 1968.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3) :586–615, 2003.

[BGGM14] Razvan Barbulescu, Pierrick Gaudry, Aurore Guillevic, and François Morain. Improvements to the number field sieve for non-prime finite fields. *INRIA Hal Archive, Report 01052449*, 2014.

[BGGM15] Razvan Barbulescu, Pierrick Gaudry, Aurore Guillevic, and François Morain. Improving NFS for the discrete logarithm problem in non-prime finite fields. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 129–155, 2015.

[BGI+14] Cyril Bouvier, Pierrick Gaudry, Laurent Imbert, Hamza Jeljeli, and Emmanuel Thomé. Discrete logarithms in GF(p) – 180 digits. *Announcement to the NMBRTHRY list, item 003161*, June 2014.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 1–16, 2014.

[BGK15] Razvan Barbulescu, Pierrick Gaudry, and Thorsten Kleinjung. The tower number field sieve. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 31–55, 2015.

[BL94] Bernhard Beckermann and George Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 1994.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4) :297–319, 2004.

[BLS11] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. On the correct use of the negation map in the Pollard Rho method. In *Public Key Cryptography*, pages 128–146, 2011.

[BMV84] Ian F. Blake, Ronald C. Mullin, and Scott A. Vanstone. Computing logarithms in GF(2^n). In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 73–82, 1984.

[BP14] Razvan Barbulescu and Cécile Pierrot. The multiple number field sieve for medium and high characteristic finite fields. *LMS Journal of Computation and Mathematics*, 17(A) :230–246, 2014.

[BS84] László Babai and Endre Szemerédi. On the complexity of matrix group problems I. In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 229–240, 1984.

[CEP83] E.R Canfield, Paul Erdös, and Carl Pomerance. On a problem of Oppenheim concerning factorisatio numerorum. In *Journal of Number Theory*, volume 17, pages 1–28, 1983.

[CGH00] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *CoRR*, cs.CR/0010019, 2000.

[CHK12] Jung Hee Cheon, Jin Hong, and Minkyu Kim. Accelerating Pollard's Rho algorithm on finite fields. *J. Cryptology*, 25(2) :195–242, 2012.

[Cop84] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30(4) :587–593, 1984.

[Cop93] Don Coppersmith. Modifications to the number field sieve. *JC*, 6(3) :169–180, 1993.

[Cop94] Don Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Mathematics of Computation*, 1994.

[COS86] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1(1) :1–15, 1986.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 1990.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 100–109, 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[DK13] Claus Diem and Sebastian Kochinke. Computing discrete logarithms with special linear systems. preprint, 2013.

[DOW92] Whitfield Diffie, Paul C. Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2) :107–125, 1992.

[EGT11] Andreas Enge, Pierrick Gaudry, and Emmanuel Thomé. An $L(1/3)$ discrete logarithm algorithm for low degree curves. *J. Cryptology*, 24(1) :24–41, 2011.

[FJM14] Pierre-Alain Fouque, Antoine Joux, and Chrysanthi Mavromati. Multi-user collisions : Applications to discrete logarithm, even-mansour and PRINCE. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 420–438, 2014.

[FPPR12] Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Guénaël Renault. Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 27–44, 2012.

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62 :865–874, 1994.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[Gal14] François Le Gall. Powers of tensors and fast matrix multiplication. *International Symposium on Symbolic and Algebraic Computation, ISSAC*, 2014.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4) :469–472, 1985.

[GG16] Steven D. Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Des. Codes Cryptography*, 78(1) :51–72, 2016.

[GGMZ13] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. On the function field sieve and the impact of higher splitting probabilities - application to discrete logarithms in and. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 109–128, 2013.

[GGV16] Pierrick Gaudry, Laurent Grémy, and Marion Videau. Collecting relations for the number field sieve in GF(p^6). *IACR Cryptology ePrint Archive*, page 124, 2016.

[GHS02] Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1) :19–46, 2002.

[GJV03] Pascal Giorgi, Claude-Pierre Jeannerod, and Gilles Villard. On the complexity of polynomial matrix computations. *Symbolic and Algebraic Computation, International Symposium ISSAC*, 2003.

[GKZ14a] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves - (or how to solve discrete logarithms in $f_{2^4 1223}$ and $f_{2^{12} 367}$). In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 126–145, 2014.

[GKZ14b] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the powers of 2. Cryptology ePrint Archive, Report 2014/300, 2014.

[GKZ15] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the discrete logarithm problem in finite fields of fixed characteristic. Cryptology ePrint Archive, Report 2015/685, 2015.

[GL14] Pascal Giorgi and Romain Lebreton. Online order basis algorithm and its impact on the block Wiedemann algorithm. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 202–209, 2014.

[Gor93] Daniel M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6(1) :124–138, 1993.

[GTTD07] Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257) :475–492, 2007.

[Gui15] Aurore Guillevic. Computing individual discrete logarithms faster in GF(p^n) with the NFS-DL algorithm. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 149–173, 2015.

[Gui16] Aurore Guillevic. Faster individual discrete logarithms in non-prime finite fields with the NFS and FFS algorithms. Cryptology ePrint Archive, Report 2016/684, 2016.

[HN13] Ming-Deh Huang and Anand Kumar Narayanan. Finding primitive elements in finite fields of small characteristic. *CoRR*, abs/1304.1206, 2013.

[HR82] Martin E. Hellman and Justin M. Reyneri. Fast computation of discrete logarithms in GF(q). In *Advances in Cryptology : Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 3–13, 1982.

[JK16] Jinhyuck Jeong and Taechan Kim. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. *IACR Cryptology ePrint Archive*, 2016.

[JL02] Antoine Joux and Reynald Lercier. The function field sieve is quite special. In *Algorithmic Number Theory, 5th International*

Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings, pages 431–445, 2002.

[JL03] Antoine Joux and Reynald Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comput.*, 72(242) :953–967, 2003.

[JL06] Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 254–270, 2006.

[JLSV06] Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren. The number field sieve in the medium prime case. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 326–344, 2006.

[JOP14] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. In Çetin Kaya Koç, editor, *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer International Publishing, 2014.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4) :263–276, 2004.

[Jou13a] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 177–193, 2013.

[Jou13b] Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. In *Selected Areas in Cryptography, SAC 2013*, pages 355–379, 2013.

[JP13] Antoine Joux and Cécile Pierrot. The special number field sieve in $\mathbb{F}_{p^n}$, Application to pairing-friendly constructions. Pairing Conference, 2013.

[JP14] Antoine Joux and Cécile Pierrot. Improving the polynomial time precomputation of Frobenius representation discrete logarithm algorithms – Simplified setting for small characteristic finite fields. In *Advances in Cryptology - ASIACRYPT 2014*

- 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, pages 378–397, 2014.

[JP16a] Antoine Joux and Cécile Pierrot. Nearly sparse linear algebra, and applications to discrete logarithms computations. *Review Volume « Contemporary Developments in Finite Fields and Applications », World Scientific Publishing Company*, 2016.

[JP16b] Antoine Joux and Cécile Pierrot. Technical history of discrete logarithms in small characteristic finite fields - the road from subexponential to quasi-polynomial complexity. *Des. Codes Cryptography*, 78(1) :73–85, 2016.

[JV12] Antoine Joux and Vanessa Vitse. Cover and decomposition index calculus on elliptic curves made practical - application to a previously unreachable curve over $\mathbb{F}_{p^6}$. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 9–26, 2012.

[Kal95] Erich Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 1995.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve : A new complexity for medium prime case. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 543–571, 2016.

[Kra22] Maurice Kraitchik. *Théorie des nombres.* Gauthier, Villars, 1922.

[KS91] Erich Kaltofen and David Saunders. On Wiedemann's method of solving sparse linear systems. *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, 1991.

[KS01] Fabian Kuhn and René Struik. Random walks revisited : Extensions of Pollard's Rho algorithm for computing multiple discrete logarithms. In *Selected Areas in Cryptography*, pages 212–229, 2001.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite fields.* Encyclopaedia of mathematics and its applications. Cambridge University Press, New York, 1997.

[LO90] Brian A. LaMacchia and Andrew M. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in Cryptology*

- CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings, pages 109–133, 1990.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 1969.

[Mas14] Maike Massierer. Some experiments investigating a possible $L(1/4)$ algorithm for the discrete logarithm problem in algebraic curves. Cryptology ePrint Archive, Report 2014/996, 2014.

[Mat03] Dimitry V. Matyukhin. On asymptotic complexity of computing discrete logarithms over GF(p). *Discrete Mathematics and Applications*, 13(1) :27–50, 2003.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5) :1639–1646, 1993.

[Nec94] Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2) :165–172, 1994.

[Odl85] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology : EUROCRYPT'84*, pages 224–314, 1985.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[PGF98] Daniel Panario, Xavier Gourdon, and Philippe Flajolet. An analytic approach to smooth polynomials over finite fields. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 226–236, 1998.

[PH78] Stephen C. Pohlig and Martin E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1) :106–110, 1978.

[Pie15] Cécile Pierrot. The multiple number field sieve with conjugation and generalized Joux-Lercier methods. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia,*

Bulgaria, April 26-30, 2015, Proceedings, Part I, pages 156–170, 2015.

[Pol78] John Pollard. Monte Carlo methods for index computations mod p. In *Mathematics of Computation*, pages 918–924, 1978.

[Pom87] Carl Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete algorithms and complexity*, pages 119–143, 1987.

[PQ12] Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a Weil descent. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 451–466, 2012.

[PW16] Cécile Pierrot and Benjamin Wesolowski. Malleability of the blockchain's entropy. *IACR Cryptology ePrint Archive*, 2016 :370, 2016.

[QD89] Jean-Jacques Quisquater and Jean-Paul Delescaille. How easy is collision search. New results and applications to DES. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 408–413, 1989.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, February 1978.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 239–252, 1989.

[Sch93] Oliver Schirokauer. Discrete logarithm and local units. *Philosophical Transactions of the Royal Society of London*, pages 409–423, 1993.

[Sch00] Oliver Schirokauer. Using number fields to compute logarithms in finite fields. *Math. Comput.*, 69(231) :1267–1283, 2000.

[Sem04] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. *IACR Cryptology ePrint Archive*, 2004 :31, 2004.

[Sho97a] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5) :1484–1509, 1997.

[Sho97b] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[SS16a] Palash Sarkar and Shashank Singh. A general polynomial selection method and new asymptotic complexities for the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2016/485, 2016.

[SS16b] Palash Sarkar and Shashank Singh. A generalisation of the conjugation method for polynomial selection for the extended tower number field sieve algorithm. Cryptology ePrint Archive, Report 2016/537, 2016.

[SS16c] Palash Sarkar and Shashank Singh. New complexity trade-offs for the (multiple) number field sieve algorithm in non-prime fields. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 429–458, 2016.

[SS16d] Palash Sarkar and Shashank Singh. Tower number field sieve variant of a recent polynomial selection method. *IACR Cryptology ePrint Archive*, 2016.

[SSHT14] Naoyuki Shinohara, Takeshi Shimoyama, Takuya Hayashi, and Tsuyoshi Takagi. Key length estimation of pairing-based cryptosystems using eta pairing over $GF(3^n)$. *IEICE Transactions*, 97-A(1) :236–244, 2014.

[Tes00] Edlyn Teske. On random walks for Pollard's Rho method. *Mathematics of Computation*, 70 :809–825, 2000.

[Tho02] Emmanuel Thomé. Subquadratic computation of vector generating polynomials and improvement of the block wiedemann algorithm. *J. Symb. Comput.*, 33(5) :757–775, 2002.

[vOW99] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12(1) :1–28, 1999.

[vzGP01] Joachim von zur Gathen and Daniel Panario. Factoring polynomials over finite fields : A survey. *Journal of Symbolic Computation*, 31 :3 – 17, 2001.

[Wie86] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1) :54–62, 1986.