Algorithmes pour la factorisation d'entiers et le calcul de logarithme discret

Cyril Bouvier

▶ **To cite this version:**

Cyril Bouvier. Algorithmes pour la factorisation d'entiers et le calcul de logarithme discret. Cryptographie et sécurité [cs.CR]. Université de Lorraine, 2015. Français. NNT : 2015LORR0053 . tel-01751648v2

HAL Id: tel-01751648
https://theses.hal.science/tel-01751648v2

Submitted on 26 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Algorithmes pour la factorisation d'entiers et le calcul de logarithme discret

THÈSE

présentée et soutenue publiquement le 22 juin 2015

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Informatique)

par

Cyril Bouvier

Composition du jury

Rapporteurs :	Guillaume Hanrot	Professeur de l'ENS de Lyon
	Reynald Lercier	Ingénieur de l'armement à la DGA et chercheur associé de l'Université de Rennes 1
Examinateurs :	Jean-Marc Couveignes	Professeur de l'Université de Bordeaux
	Nadia Heninger	Assistant Professor de l'Université de Pennsylvanie
	Pierre-Étienne Moreau	Professeur de l'Université de Lorraine
Directeur de thèse :	Paul Zimmermann	Directeur de Recherche à Inria

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Remerciements

Je tiens tout d'abord à remercier Guillaume Hanrot et Reynald Lercier d'avoir accepté la lourde tâche de relire ce manuscrit. Je remercie également Jean-Marc Couveignes, Nadia Heninger et Pierre-Étienne Moreau d'avoir accepté de faire partie de mon jury de thèse.

Je remercie mon directeur de thèse, Paul Zimmermann. Il m'a accompagné lors de mes premiers pas dans le monde de la recherche pendant mon stage de L3 (déjà sur la factorisation !). Il a aussi su me guider dans mes recherches durant cette thèse et s'est montré très disponible. Ce manuscrit a énormement profité de ses nombreuses remarques et corrections.

Pour l'accueil, les échanges et l'ambiance, aussi bien au travail qu'aux pauses café, je remercie aussi tous les membres de l'équipe CARAMEL, en particulier, les autres doctorant que j'ai croisés au cours de ma thèse : Nicolas Estibals, qui finissait sa thèse lorsque que je commençais la mienne et qui m'a aidé à trouver des enseignements, Răzvan Bărbulescu et Hamza Jeljeli, mes anciens co-bureaux qui ont commencé leur thèse au même moment que moi, ainsi que Svyatoslav Covanov, Laurent Grémy et Hugo Labrande.

Je remercie mes coauteurs avec qui j'ai eu le plaisir de collaborer, Shi Bai, Joppe Bos, Thorsten Kleinjung, Alexander Kruppa et Peter Montgomery, ceux avec qui j'ai eu l'occasion de travailler sur CADO-NFS, Laurent Imbert et François Morain, ainsi que les enseignants qui m'ont confié leurs étudiants pendant mon monitorat, Francis Alexandre, Moufida Maimour et Alexandre Parodi.

Je tiens également à remercier mes parents, mes frères et sœur et ma famille pour leur soutien. Je remercie Emmanuel, Lucas, Margot, Paul, Romain, Tzzm et Ugo pour leur amitié. Et merci à tous les bénévoles de «la Feinte de l'Ours» pour ces moments de détente autour d'un jeu.

Je termine ces remerciements par celle qui a partagé cette aventure avec moi et qui m'a supporté durant cette thèse. Merci Alice pour ta patience et pour tous ces moments partagés depuis toutes ces années.

Table des matières

Introduction **1**

1 Courbes elliptiques : propriétés galoisiennes et familles adaptées à ECM **5**

1.1 Courbes elliptiques 5

1.1.1 Définitions et propriétés 5

1.1.2 Courbes de Montgomery et courbes d'Edwards 10

1.2 L'algorithme ECM 12

1.2.1 Description de l'algorithme ECM 13

1.2.2 Complexité de l'algorithme ECM 15

1.2.3 Familles de courbes elliptiques utilisées pour ECM 16

1.3 Propriétés galoisiennes et probabilités de divisibilité 17

1.3.1 Propriétés galoisiennes des points de torsion des courbes elliptiques 17

1.3.2 Calcul explicite de $\mathbb{Q}(E[m])$ et $\rho_m(\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}))$ pour les puissances de nombre premier 20

1.3.3 Divisibilité par une puissance de nombre premier 23

1.4 Applications à certaines familles de courbes elliptiques 27

1.4.1 Meilleures courbes d'Edwards avec torsion $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ en utilisant les polynômes de division 28

1.4.2 Meilleures courbes de Suyama par changement direct du groupe de Galois 31

1.4.3 Comparaisons 33

1.5 Conclusion et perspectives 34

2 L'algorithme NFS pour la factorisation d'entiers **35**

2.1 Description générale de l'algorithme NFS 35

2.1.1 Les différentes étapes de l'algorithme NFS 35

2.1.2 Complexité, implémentations et records 38

2.2 Sélection polynomiale 38

2.2.1 Variante SNFS 39

2.3 Collecte des relations : l'étape de crible 39

2.4 Filtrage . . . 41
2.5 Algèbre linéaire . . . 41
2.6 Calcul des racines carrées . . . 42
2.6.1 Obstructions et caractères . . . 42
2.6.2 Calcul de racine carrée . . . 43

3 La sélection polynomiale pour NFS **44**
3.1 Généralités sur la sélection polynomiale . . . 44
3.1.1 Description du problème de sélection polynomiale . . . 44
3.1.2 Évaluer la qualité des polynômes . . . 45
3.1.3 La sélection polynomiale linéaire . . . 47
3.2 Génération de polynômes dans le cas linéaire . . . 48
3.2.1 Le premier algorithme de Kleinjung . . . 48
3.2.2 Le second algorithme de Kleinjung . . . 50
3.2.3 Utilisation de spécial-q . . . 52
3.3 Amélioration de l'algorithme d'optimisation de la taille . . . 54
3.3.1 Descente locale . . . 54
3.3.2 Utilisation de translations avant la descente locale . . . 56
3.3.3 Utilisation de l'algorithme LLL avant la descente locale . . . 56
3.3.4 Minimiser simultanément les coefficients de degré $d-2$ et $d-3$ . . . 57
3.3.5 Résultats expérimentaux . . . 58
3.4 La sélection polynomiale non linéaire . . . 61
3.4.1 Progressions géométriques . . . 62
3.4.2 Progressions géométriques de longueur 5 et polynômes de degré 3 . . . 63
3.5 Conclusion et perspectives . . . 65

4 Les algorithmes NFS-DL et FFS pour le calcul de logarithme discret **68**
4.1 Le logarithme discret dans les corps finis . . . 68
4.2 L'algorithme NFS-DL pour le calcul de logarithme discret dans les corps premiers 70
4.2.1 Description générale de l'algorithme NFS-DL . . . 70
4.2.2 Les différentes étapes de l'algorithme NFS-DL . . . 73
4.3 Calcul de logarithme discret dans un corps premier de 180 chiffres . . . 74
4.4 L'algorithme FFS pour le calcul de logarithme discret en petite caractéristique . 76
4.5 Calcul de logarithme discret dans $\mathbb{F}_{2^{809}}$ avec FFS . . . 78

5 L'étape de filtrage des algorithmes de crible : NFS, NFS-DL et FFS **83**
5.1 Les objectifs de l'étape de filtrage . . . 83
5.1.1 Le cas de la factorisation . . . 84

5.1.2 Le cas du calcul de logarithme discret . . . 84
5.1.3 Points communs et différences . . . 84
5.2 Description de l'étape de filtrage . . . 85
5.2.1 La suppression des singletons . . . 86
5.2.2 La suppression des cliques . . . 87
5.2.3 Début d'élimination de Gauss : la combinaison des relations . . . 89
5.3 Les fonctions de poids lors de la suppression des cliques . . . 92
5.3.1 Les fonctions de poids étudiées . . . 93
5.3.2 Les fonctions de poids utilisées dans les différentes implémentations disponibles . . . 94
5.3.3 Description des expériences . . . 94
5.4 Comparaison des fonctions de poids avec NFS . . . 95
5.4.1 Expériences : RSA-155, B200 et RSA-704 . . . 95
5.4.2 Résultats . . . 95
5.5 Comparaison des fonctions de poids avec NFS-DL . . . 97
5.5.1 Expériences : p155 et p180 . . . 97
5.5.2 Résultats . . . 98
5.6 Comparaison des fonctions de poids avec FFS . . . 99
5.6.1 Expériences : $\mathbb{F}_{2^{619}}$, $\mathbb{F}_{2^{809}}$ et $\mathbb{F}_{2^{1039}}$ . . . 99
5.6.2 Résultats . . . 99
5.7 Influence de l'excès initial sur l'efficacité du filtrage . . . 101
5.8 Conclusion et perspectives . . . 103

Bibliographie **112**

Introduction

La cryptographie est la discipline de la cryptologie qui s'attache à l'étude de la sécurité, c'est-à-dire de la confidentialité, l'authenticité et l'intégrité, de messages échangés entre deux ou plusieurs entités. La cryptographie est divisée en deux branches : la cryptographie à *clé privée* (ou *symétrique)* et la cryptographie à *clé publique* (ou *asymétrique).* Dans la cryptographie à clé privée, la sécurité des messages est assurée par un secret partagé par toutes les entités impliquées. Dans la cryptographie à clé publique, il existe deux clés (une privée et une publique) et la sécurité des messages repose sur la difficulté de trouver la clé privée à partir de la clé publique. Pour cela, des fonctions *à sens unique* sont utilisées, c'est-à-dire des fonctions facilement calculables mais dont l'inverse est très difficile à calculer. La sécurité de la majorité des cryptosystèmes à clé publique actuels est basée sur la difficulté de deux problèmes : la *factorisation d'entiers* et le *calcul de logarithme discret*. L'étude d'algorithmes permettant de résoudre ces deux problèmes est le sujet de cette thèse.

Factorisation d'entiers

Le problème de la factorisation d'entiers est, étant donné un entier N non nul, de trouver tous les facteurs premiers de N, c'est-à-dire tous les nombres premiers p divisant N. La difficulté du problème de la factorisation d'entiers est utilisée dans le cryptosystème RSA, décrit en 1978 par Rivest, Shamir et Adleman [129]. La sécurité du cryptosystème RSA repose sur le fait que, en général, il est très difficile, étant donné un entier N produit de deux nombres premiers p et q de taille similaire, de calculer les facteurs premiers p et q.

L'algorithme de factorisation le plus simple est l'algorithme par divisions successives, qui consiste à tester tous les nombres premiers inférieurs à la racine carrée de l'entier à factoriser. Les algorithmes non triviaux de factorisation se divisent en deux catégories : ceux dont le but est de trouver le plus petit facteur premier, dont la complexité dépend essentiellement de la taille de ce plus petit facteur, et ceux dont le but est de trouver la factorisation complète, dont la complexité dépend de la taille de l'entier à factoriser. Parmi les algorithmes de la première catégorie, les trois algorithmes suivants ont une complexité sous-exponentielle en la taille du plus petit facteur premier : l'algorithme P-1, décrit en 1974 par Pollard [122], l'algorithme P+1, décrit en 1982 par Williams [146], et l'algorithme ECM (*Elliptic Curve Method*), décrit en 1987 par H. Lenstra [97]. Parmi les algorithmes de la seconde catégorie, il existe des algorithmes dont la complexité est exponentielle en la taille de l'entier à factoriser, par exemple l'algorithme SQUFOF (*Square Forms Factorization*) décrit en 1975 par Shanks [137], et des algorithmes dont la complexité est sous-exponentielle en la taille de l'entier à factoriser : l'algorithme CFRAC (*Continued Fraction*), décrit en 1931 par Lehmer et Powers [91] et développé en 1975 par Morrison et Brillhart [109], l'algorithme de Dixon, décrit en 1981 [52], l'algorithme QS (*Quadratic Sieve*), décrit en 1985 par Pomerance [126], l'algorithme MPQS (*Multiple Polynomial Quadratic Sieve*), décrit en 1987 par Silverman [139], et l'algorithme NFS (*Number Field Sieve*), décrit en 1993 par Pollard [124]. Tous

les algorithmes de la seconde catégorie listés ci-dessus se basent sur la même idée, qui remonte à Fermat : si N est l'entier à factoriser et si deux entiers x et y, tels que $x^2 \equiv y^2 \pmod{N}$ et $x \not\equiv \pm y \pmod{N}$, sont connus alors un facteur de N, différent de 1 et N, peut être trouvé en calculant $\gcd(x+y, N)$ ou $\gcd(x-y, N)$. Ces algorithmes vont donc s'attacher à construire des entiers x et y tels que $x^2 \equiv y^2 \pmod{N}$ en espérant que $\gcd(x \pm y, N)$ soit différent de 1 et N. Si N a au moins deux facteurs premiers impairs et si les entiers x et y sont considérés comme uniformément aléatoires, alors la probabilité que $\gcd(x \pm y, N)$ soit différent de 1 et N est supérieure à $1/2$. La majorité des algorithmes listés ci-dessus sont décrits en détail dans le livre de Crandall et Pomerance [45].

Dans cette thèse, nous nous intéresserons en particulier aux algorithmes ECM et NFS.

Calcul de logarithme discret

Le problème du calcul de logarithme discret est, étant donnés un groupe G cyclique, fini et engendré par g, et un élément $h \in G$, de trouver l'entier $a \in [0, \#G[$ tel que $h = g^a$. L'entier a est appelé le logarithme de h dans la base g et est noté $\log_g(h)$ ou simplement $\log(h)$ lorsqu'il n'y a pas d'ambiguïté sur la base.

La difficulté du problème de calcul de logarithme discret est à la base de la sécurité du protocole d'échange de clés Diffie–Hellman [51], qui a été décrit en 1976 et qui est le premier exemple de cryptographie à clé publique, et du cryptosystème de ElGamal [55], décrit en 1985.

La difficulté du problème de calcul de logarithme discret dépend du groupe utilisé. En pratique, les groupes utilisés dans les cryptosystèmes dont la sécurité repose sur la difficulté du calcul de logarithme discret sont soit des sous-groupes du groupe multiplicatif d'un corps fini, soit des groupes de points de courbes définies sur un corps fini (en particulier, les groupes de points de courbes elliptiques). Dans cette thèse, pour abréger, nous parlerons de calcul de logarithme discret dans un corps fini pour parler du calcul de logarithme discret dans un sous-groupe du groupe multiplicatif d'un corps fini.

Pour attaquer le problème du logarithme discret, l'un des algorithmes les plus importants est l'algorithme de Pohlig–Hellman [121] décrit en 1978 qui montre que le calcul de logarithme discret dans un groupe peut se ramener au calcul de logarithme discret dans ses sous-groupes d'ordre premier et que le coût total du calcul est dominé par le coût du calcul dans le plus grand sous-groupe d'ordre premier. Il existe ensuite des algorithmes génériques qui peuvent être utilisés pour tous les groupes, comme l'algorithme Baby-Step-Giant-Step qui est l'application du compromis temps–mémoire au problème du logarithme discret, et l'algorithme rho de Pollard [123], décrit en 1978. La majorité des algorithmes de calcul de logarithme discret récents se basent sur la méthode de *calcul d'indices* décrite en 1922 par Kraitchik [85]. Cette méthode ne fonctionne pas pour tous les groupes mais sert de base aux algorithmes de calcul de logarithme discret dans les corps finis, comme l'algorithme NFS-DL, décrit en 1993 par Gordon [64], ou l'algorithme FFS, décrit en 1994 par Adleman [2], ainsi qu'à certains algorithmes pour le calcul de logarithme discret dans les groupes de points de courbes, comme l'algorithme pour les courbes de genre élevé décrit en 2011 par Enge, Gaudry et Thomé [57].

Dans cette thèse, nous nous intéresserons aux algorithmes NFS-DL et FFS pour le calcul de logarithme discret dans les corps finis.

Complexité et friabilité

La complexité des différents algorithmes de factorisation et de calcul de logarithme discret est souvent exprimée à l'aide de la notation L définie par

$$\mathrm{L}_x(\alpha, c) = \exp\left((c + \mathrm{o}(1))(\log x)^{\alpha}(\log\log x)^{1-\alpha}\right),$$

où c est un réel positif non nul et α un réel appartenant à l'intervalle $[0, 1]$. Le cas où α est nul correspond à une complexité polynomiale (en $\log(x)$), le cas où α est égal à 1 correspond à une complexité exponentielle et le cas où $\alpha \in]0, 1[$ correspond à une complexité sous-exponentielle.

Une autre notion très importante, qui apparaît à la fois dans les algorithmes de factorisation et de calcul de logarithme discret, est la *friabilité*. Il existe différentes notions de friabilité et, dans cette thèse, nous utiliserons les définitions suivantes : si B est un entier positif non nul, un entier x sera dit B-friable si et seulement si tout nombre premier divisant x est inférieur ou égal à B et un entier x sera dit B-ultrafriable si et seulement si toute puissance de nombre premier divisant x est inférieure ou égale à B. Par exemple, l'entier $12 = 2^2 \times 3$ est 3-friable mais n'est pas 3-ultrafriable. De plus, si B' est un entier strictement supérieur à B alors un entier x sera dit (B, B')-ultrafriable si et seulement si x est B-ultrafriable ou s'il existe un nombre premier $p \in]B, B']$ tel que x/p est B-ultrafriable.

Plan de la thèse

Dans le premier chapitre, après avoir rappelé quelques propriétés des courbes elliptiques, nous présenterons l'algorithme de factorisation d'entier ECM et nous proposerons une méthode pour analyser les courbes elliptiques utilisées dans cet algorithme en étudiant les propriétés galoisiennes des polynômes de division. Ensuite dans le chapitre 2, nous donnerons une description générale de l'algorithme de factorisation d'entier NFS ainsi que des détails sur les différentes étapes le composant, puis dans le chapitre 3, nous nous intéresserons en particulier à l'étape de sélection polynomiale de l'algorithme NFS et nous proposerons des améliorations d'algorithmes existants. Dans le chapitre 4, nous décrirons les algorithmes NFS-DL et FFS pour le calcul de logarithme discret dans les corps finis et nous présenterons deux calculs de logarithme discret effectués durant cette thèse, l'un dans un corps fini dont le cardinal est un nombre premier de 180 chiffres décimaux avec NFS-DL et l'autre dans $\mathbb{F}_{2^{809}}$ avec FFS. Enfin, dans le chapitre 5, nous étudierons une étape commune à l'algorithme NFS pour la factorisation et aux algorithmes NFS-DL et FFS pour le calcul de logarithme discret : l'étape de filtrage. Nous décrirons cette étape en détail et nous présenterons une amélioration dont nous validerons l'impact en utilisant des données provenant de plusieurs calculs de factorisation et de logarithme discret.

Publications et contributions logicielles

[10] R. Barbulescu, J. W. Bos, C. Bouvier, T. Kleinjung et P. L. Montgomery. « Finding ECM-Friendly Curves through a Study of Galois Properties ». In : *ANTS X: Proceedings of the Tenth Algorithmic Number Theory Symposium*. Sous la dir. d'E. W. Howe et K. S. Kedlaya. T. 1. Open Book Series. Berkeley : Mathematical Sciences Publishers, 2013, p. 63–86. doi : 10.2140/obs.2013.1.63. url : https://hal.inria.fr/hal-00671948

[25] C. Bouvier. *The filtering step of discrete logarithm and integer factorization algorithms*. Preprint, 22 pages. 2013. url : http://hal.inria.fr/hal-00734654

[12] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau et P. Zimmermann. « Discrete logarithm in GF(2^{809}) with FFS ». In : *Public-Key Cryptography – PKC 2014*. Sous la dir. de H. Krawczyk. T. 8383. Lecture Notes in Computer Science. Springer-Verlag, 2014, p. 221–238. ISBN : 978-3-642-54630-3. DOI : 10.1007/978-3-642-54631-0_13. URL : http://hal.inria.fr/hal-00818124

[27] C. Bouvier et P. Zimmermann. « Division-Free Binary-to-Decimal Conversion ». In : *IEEE Transactions on Computers* 63.8 (2014), p. 1895–1901. ISSN : 0018-9340. DOI : 10.1109/TC.2014.2315621. URL : https://hal.inria.fr/hal-00864293

[5] S. Bai, C. Bouvier, A. Kruppa et P. Zimmermann. « Better polynomials for GNFS ». In : *Mathematics of Computation* (2015). DOI : 10.1090/mcom3048. URL : https://hal.inria.fr/hal-01089507

Parmi ces publications, les résultats de l'article [27] n'apparaissent pas dans cette thèse car le sujet de l'article est trop éloigné. De plus, durant cette thèse, des contributions ont été faites à deux logiciels scientifiques : `cado-nfs` et GMP-ECM. Par exemple, les améliorations de l'étape de sélection polynomiale décrites dans le chapitre 3 et l'étape de filtrage pour les algorithmes NFS, NFS-DL et FFS décrite dans le chapitre 5 ont été implémentées dans `cado-nfs`.

[33] The CADO-NFS Development Team. *CADO-NFS, An Implementation of the Number Field Sieve Algorithm*. URL : http://cado-nfs.gforge.inria.fr/

[61] P. Zimmermann et al. *GMP-ECM (Elliptic Curve Method for Integer Factorization)*. URL : https://gforge.inria.fr/projects/ecm/

Chapitre 1

Courbes elliptiques : propriétés galoisiennes et familles adaptées à ECM

L'algorithme ECM est très utilisé pour la factorisation d'entiers lorsqu'il s'agit de trouver des facteurs de tailles moyennes ou comme sous-routine dans l'étape de crible de l'algorithme NFS. Dans la section 1.1, nous introduirons les courbes elliptiques, ensuite dans la section 1.2 nous présenterons l'algorithme ECM, dans la section 1.3 nous étudierons les propriétés galoisiennes des courbes elliptiques et enfin dans la section 1.4 nous utiliserons ces résultats pour étudier des familles de courbes elliptiques adaptées à l'algorithme ECM.

La majorité des résultats de ce chapitre sont tirés de l'article [10], écrit en collaboration avec R. Barbulescu, J. Bos, T. Kleinjung et P. Montgomery.

1.1 Courbes elliptiques

Les courbes elliptiques sont à la base de l'algorithme ECM. Dans cette section, nous allons définir les courbes elliptiques, donner quelques-unes de leurs propriétés et ensuite étudier deux familles de courbes elliptiques utilisées dans le cadre de l'algorithme ECM.

1.1.1 Définitions et propriétés

Les définitions et propriétés que nous présenterons dans cette section ne seront pas toujours les plus générales possible, mais seulement ce qui sera nécessaire à la compréhension de l'algorithme ECM et des théorèmes de la section 1.3. Une présentation précise et détaillée des courbes elliptiques peut être trouvée dans le livre de Silverman [138].

Avant de définir les courbes elliptiques, il nous faut définir les espaces projectifs, et en particulier le plan projectif, qui est l'ensemble dans lequel les courbes seront définies.

Définition 1.1.1 (Espace projectif)**.** Soit K un corps. L'espace projectif de dimension n sur K, noté $\mathbb{P}^n(K)$, est l'ensemble $K^{n+1}\backslash\{\mathbf{0}\}$ quotienté par la relation d'équivalence $\sim$ définie par : $(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$ si et seulement s'il existe un élément non nul λ de K tel que $x_i = \lambda y_i$ pour $0 \leqslant i \leqslant n$. La classe d'équivalence de $(x_0, \ldots, x_n) \in K^{n+1}\backslash\{\mathbf{0}\}$ est notée $(x_0 : \ldots : x_n)$. L'espace projectif de dimension 2 est appelé plan projectif.

Avant de définir ce qu'est une courbe dans le plan projectif, posons la notation suivante : si K est un corps, la clôture algébrique de K est notée $\overline{K}$.

Définition 1.1.2 (Courbe)**.** Soient K un corps et $(X : Y : Z)$ les coordonnées homogènes du plan projectif. Une courbe algébrique projective (ou simplement courbe dans la suite) définie sur K est l'ensemble des zéros dans $\mathbb{P}^2(K)$ d'un polynôme homogène, irréductible et non nul de $K[X, Y, Z]$. Dans la suite, nous utiliserons la notation C/K pour parler d'une courbe C définie sur un corps K.

Une courbe est définie comme les racines d'un polynôme homogène dans le plan projectif, mais, souvent, nous nous intéresserons aux racines du polynôme définies sur une extension algébrique du corps de définition, et en particulier sur la clôture algébrique.

Définition 1.1.3 (Points rationnels)**.** Si L est une extension algébrique de K et C/K est une courbe définie par le polynôme homogène F, alors l'ensemble des points L-rationnels, noté $C(L)$, est défini par

$$C(L) = \left\{ (x : y : z) \in \mathbb{P}^2(L) \;\middle|\; F(x, y, z) = 0 \right\}.$$

L'ensemble des points K-rationnels sera simplement appelé l'ensemble des points rationnels et noté C au lieu de $C(K)$.

Pour pouvoir définir ce qu'est une courbe elliptique, il est nécessaire de définir les notions de courbes non singulières et de genre.

Définition 1.1.4 (Courbe non singulière et genre)**.** Soient K un corps et C/K une courbe définie par le polynôme homogène F. La courbe C est non singulière (ou lisse) si et seulement si F, $\frac{\partial F}{\partial X}$, $\frac{\partial F}{\partial Y}$ et $\frac{\partial F}{\partial Z}$ n'admettent pas de zéro commun sur $\overline{K}$.
Le genre g d'une courbe non singulière est défini par

$$g = \frac{(d-1)(d-2)}{2},$$

où d est le degré total du polynôme homogène F.

Toutes les notions nécessaires à la définition des courbes elliptiques ont été présentées.

Définition 1.1.5 (Courbe elliptique)**.** Soit K un corps. Une courbe E/K est une courbe elliptique si et seulement si elle est non singulière, de genre 1 et possède au moins un point rationnel.

Par exemple, si a et b sont deux éléments d'un corps K, alors la courbe définie par

$$ZY^2 = X^3 + aXZ^2 + bZ^3. \tag{1.1}$$

est une courbe elliptique si et seulement si $4a^3 + 27b^2$ est non nul. En effet, une courbe définie par une équation de la forme de l'équation (1.1) admet toujours le point $(0 : 1 : 0)$ comme point rationnel, elle est non singulière si et seulement si $4a^3 + 27b^2$ est non nul et, dans le cas où elle est non singulière, elle est de genre 1. Une courbe définie par une équation de la forme de l'équation (1.1) est dite sous forme de Weierstrass courte.

Proposition 1.1.6 (Forme de Weierstrass courte [138, section III.1])**.** *Soit K un corps de caractéristique différente de 2 et 3. Toute courbe elliptique sur K est isomorphe à une courbe elliptique sous forme de Weierstrass courte.*

La définition précise de morphisme entre courbes n'est pas donnée ici (voir [138, section I.3]), cette proposition est citée seulement pour justifier que, dans les preuves, il nous suffira de considérer uniquement les courbes sous forme de Weierstrass courte, dès que la caractéristique est

différente de 2 et 3. Pour des raisons d'efficacité, l'algorithme ECM utilise généralement des courbes elliptiques qui ne sont pas sous forme de Weierstrass courte, mais elles seront toujours équivalentes à des courbes sous forme de Weierstrass courte dès que le nombre que nous voudrons factoriser ne sera divisible ni par 2 ni par 3.

Les courbes elliptiques peuvent aussi être définies de façon affine, c'est-à-dire, si K est le corps de définition, sur K^2, l'espace vectoriel de dimension 2, au lieu du plan projectif $\mathbb{P}^2(K)$. Pour passer de la version projective à la version affine, il suffit de diviser le polynôme qui définit la courbe elliptique par Z^3 et d'effectuer le changement de variables suivant : $x = X/Z$ et $y = Y/Z$. Cela permet d'obtenir la version affine de la courbe elliptique définie par un polynôme de $K[x, y]$. Un point $(X : Y : Z)$, avec $Z \neq 0$, sur la courbe projective correspond au point $(X/Z, Y/Z)$ sur la courbe affine, et inversement le point (x, y) sur la courbe affine correspond au point $(x : y : 1)$ sur la courbe projective. Le point $\mathcal{O} = (0 : 1 : 0)$ est le seul point de la courbe pour lequel $Z = 0$ et c'est donc le seul point projectif qui n'a pas d'équivalent affine — c'est pour cette raison qu'il est souvent appelé *point à l'infini* — et est ajouté à l'ensemble des points affines de la courbe. Dans la suite nous utiliserons soit la version affine soit la version projective, suivant ce qui est le plus adapté.

Nous allons maintenant définir les notions autour des courbes elliptiques que nous utiliserons dans la suite. Pour l'algorithme ECM, la propriété principale des courbes elliptiques est que l'ensemble des points rationnels d'une courbe elliptique forme un groupe. En effet, toute courbe elliptique sous forme de Weierstrass courte peut être munie d'une loi de groupe dont le neutre est le point à l'infini $\mathcal{O} = (0 : 1 : 0)$ [138, section III.2]. La loi de groupe peut être illustrée géométriquement par la méthode des droites et des tangentes, comme sur la figure 1.1.

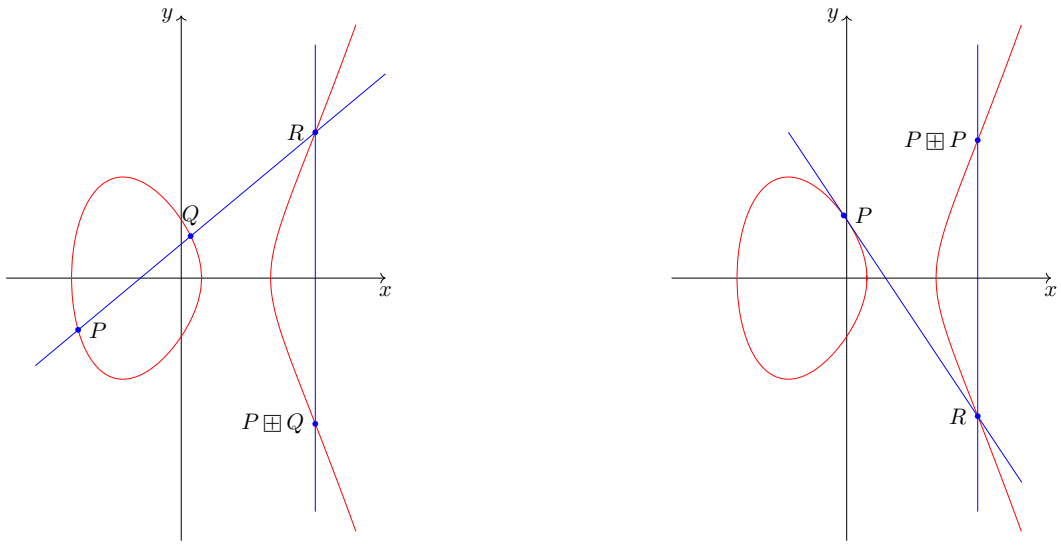

FIGURE 1.1 – Interprétation géométrique de la loi de groupe pour la courbe $E/\mathbb{R}$: $y^2 = x^3 - 3x + 1$. La somme de deux points P et Q est le symétrique par rapport à l'axe des x du point d'intersection R entre la courbe et la droite passant par P et Q. Le double d'un point P est le symétrique par rapport à l'axe des x du point d'intersection R entre la tangente au point P et la courbe.

L'addition de deux points sur une courbe elliptique sera noté $\boxplus$. L'opposé d'un point P sera noté $\boxminus P$. Pour une courbe sous forme de Weierstrass courte, $\boxminus(X : Y : Z) = (X : -Y : Z)$. Pour une courbe elliptique en version projective, la loi de groupe peut être exprimée à l'aide de polynômes, c'est-à-dire que si P et Q sont deux points sur une courbe elliptique, les coordonnées

du point $P \boxplus Q$ sont des polynômes en les coordonnées des points P et Q. Pour une courbe elliptique en version affine, la loi de groupe est donnée par des fractions rationnelles. Si E/K est une courbe elliptique et L une extension algébrique du corps K, alors la loi de groupe sur E étendue à $E(L)$ est encore une loi de groupe. En particulier, la somme de deux points L-rationnels est encore L-rationnel. Les formules de la loi de groupe pour les différentes formes de courbes elliptiques, en particulier pour les courbes sous forme de Weierstrass courte, peuvent être trouvées dans [22].

La multiplication d'un point P par un entier $m \in \mathbb{Z}$, appelée *multiplication scalaire*, est notée mP et est définie par :

$$mP = \begin{cases} \mathcal{O} & \text{si } m = 0 \\ \underbrace{P \boxplus \cdots \boxplus P}_{m \text{ fois}} & \text{si } m > 0 \\ \boxminus(-m)P & \text{si } m < 0 \end{cases}.$$

Pour tout entier m, la multiplication par m définit un endomorphisme de la courbe elliptique.

Définition 1.1.7 (Multiplication complexe)**.** Soient K un corps et E/K une courbe elliptique. Si l'ensemble des endomorphismes de la courbe contient au moins un endomorphisme qui ne provient pas de la multiplication scalaire par un entier m, alors nous dirons que la courbe E est avec multiplication complexe. Dans le cas contraire, nous dirons que la courbe E est sans multiplication complexe.

L'étude des points d'ordre fini pour la loi de groupe associée à une courbe elliptique est une part importante de l'étude des courbes elliptiques.

Définition 1.1.8 (Torsion)**.** Soient K un corps, E/K une courbe elliptique, L une extension algébrique de K et m un entier strictement positif. Les points L-rationnels de m-torsion de E, notés $E(L)[m]$, sont définis par

$$E(L)[m] = \{\, P \in E(L) \mid mP = \mathcal{O} \,\}.$$

L'ensemble des points L-rationnels de torsion de E est l'ensemble

$$E(L)_{\text{tors}} = \bigcup_{m=1}^{\infty} E(L)[m].$$

Les ensembles $E(K)[m]$ et $E(K)_{\text{tors}}$ seront souvent simplement notés $E[m]$ et E_{tors}.

La théorie des groupes montre que $E(L)[m]$ et $E(L)_{\text{tors}}$ sont des sous-groupe de $E(L)$. Si, en plus, L' est une extension algébrique de L, alors $E(L)[m]$ et $E(L)_{\text{tors}}$ sont, respectivement, des sous-groupes de $E(L')[m]$ et $E(L')_{\text{tors}}$. Un point d'une courbe elliptique qui n'appartient pas à $E(L)_{\text{tors}}$ sera dit d'*ordre infini* sur L.

Nous nous servirons des trois propriétés importantes suivantes sur la torsion des courbes elliptiques.

Proposition 1.1.9 ([138, corollaire III.6.4])**.** *Soient K un corps, E/K une courbe elliptique et m un entier strictement positif. Si K est de caractéristique nulle ou si la caractéristique de K ne divise pas m, alors*

$$E(\overline{K})[m] = \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

Cela implique que, si les conditions de la proposition sont vérifiées et si L est une extension algébrique de K, alors, comme $E(L)[m]$ est un sous-groupe de $E(\overline{K})[m]$, $E(L)[m]$ est un sous-groupe de $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Les deux théorèmes suivants concernent l'ensemble des points $\mathbb{Q}$-rationnels des courbes elliptiques définies sur $\mathbb{Q}$.

Théorème 1.1.10 (Mordell–Weil)**.** *Soit $E/\mathbb{Q}$ une courbe elliptique. Alors il existe un entier $r \geqslant 0$ tel que le groupe $E(\mathbb{Q})$ est isomorphe à $\mathbb{Z}^r \times E(\mathbb{Q})_{tors}$.*

Théorème 1.1.11 (Théorème de Mazur)**.** *Soit $E/\mathbb{Q}$ une courbe elliptique, alors le groupe de torsion $E(\mathbb{Q})_{tors}$ est l'un des 15 groupes suivants :*

— $\mathbb{Z}/n\mathbb{Z}$, *pour* $n \in \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12 \}$

— $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2n\mathbb{Z}$, *avec* $n \in \{ 1, 2, 3, 4 \}$.

Pour une courbe elliptique définie sur $\mathbb{Q}$, l'entier r du théorème de Mordell–Weil est appelé le *rang* de la courbe.

Définition 1.1.12. Soient $m \geqslant 2$ un entier, K un corps et E/K une courbe elliptique. La plus petite extension algébrique L de K telle que tous les points de m-torsion de E sont L-rationnels, c'est-à-dire telle que $E(L)[m] = E(\overline{K})[m] = \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$, est notée $K(E[m])$.

Dans la suite, nous utiliserons beaucoup le fait que pour un entier $m \geqslant 2$ et une courbe elliptique $E/\mathbb{Q}$, $\mathbb{Q}(E[m])$ est une extension galoisienne de $\mathbb{Q}$. Nous n'en donnerons pas la preuve mais la section 1.3.2 décrit un algorithme naïf permettant de calculer $\mathbb{Q}(E[m])$ qui donne une bonne intuition de la preuve. Les outils principaux de cet algorithme sont les polynômes de division que nous allons maintenant définir.

Définition 1.1.13 (Polynôme de division)**.** Soient $m \geqslant 2$ un entier et $E/\mathbb{Q}$ une courbe elliptique sous forme de Weierstrass courte en version affine. Le polynôme de division P_m est le polynôme unitaire tel que $x \in \overline{K}$ est une racine de P_m si et seulement s'il existe $y \in \overline{K}$ tel que (x, y) est un point $\overline{K}$-rationnel de m-torsion de la courbe E. De plus, le polynôme P_m^{new} est défini comme le polynôme unitaire tel que $x \in \overline{K}$ est une racine de P_m^{new} si et seulement s'il existe $y \in \overline{K}$ tel que (x, y) est un point $\overline{K}$-rationnel d'ordre exactement m de la courbe E.

Il peut être montré que les polynômes P_m et P_m^{new} sont des polynômes à coefficients dans $\mathbb{Q}$ et que le degré de P_m est $(m^2 + 2 - 3\eta)/2$, où $\eta = m \bmod 2$.

Nous allons maintenant nous intéresser à la réduction de courbes elliptiques définies sur $\mathbb{Q}$ modulo des nombres premiers. Pour cela, nous avons besoin de la définition suivante.

Définition 1.1.14 (Discriminant)**.** Étant donnés un corps K de caractéristique différente de 2 et 3 et une courbe sous forme de Weierstrass courte dont les coefficients sont les éléments a et b de K, le discriminant de la courbe, noté Δ, est défini par $\Delta = -16(4a^3 + 27b^2)$.

Le discriminant d'une courbe elliptique est toujours non nul. Dans la suite, pour un nombre premier p et un nombre rationnel q, la valuation de q en p, noté $v_p(q)$, est définie comme la différence entre l'exposant de la plus grande puissance de p divisant le numérateur de q et l'exposant de la plus grande puissance de p divisant le dénominateur de q.

Définition 1.1.15 (Bonne réduction et courbe réduite)**.** Soient $p \geqslant 5$ un nombre premier et $E/\mathbb{Q}$ une courbe elliptique. La courbe E a bonne réduction en p si et seulement si les coefficients a et b de l'équation de Weierstrass courte ont une valuation en p positive ou nulle et si le discriminant est non nul modulo p. Dans ce cas, la courbe réduite est la courbe définie par l'équation de Weierstrass courte où les coefficients ont été réduits modulo p et l'ensemble des points $\mathbb{F}_p$-rationnels de la courbe réduite est noté $E(\mathbb{F}_p)$. Dans le cas contraire, nous dirons que la courbe E a mauvaise réduction en p.

Pour une courbe elliptique donnée, il n'existe qu'un nombre fini de nombres premiers tels que la courbe a mauvaise réduction. Dans le cas où une courbe E a bonne réduction modulo un nombre premier p, il est possible de définir une application de $E(\mathbb{Q})$ dans $E(\mathbb{F}_p)$ par :

$$(x : y : z) \in E(\mathbb{Q}) \mapsto (x \bmod p : y \bmod p : z \bmod p) \in E(\mathbb{F}_p),$$

où les points du plan projectif sont représentés par des triplets $(x : y : z)$ tels que les dénominateurs de x, y et z ne soient pas divisibles par p (cela est toujours possible). Cette application est appelée la réduction. La réduction d'un point P de E sera notée $\widetilde{P}$. De plus, la réduction est un homomorphisme de groupe, c'est-à-dire que la loi de groupe sur $E(\mathbb{F}_p)$ est définie par les équations de la loi de groupe sur E réduites modulo p.

Proposition 1.1.16 ([138, proposition VII.3.1])**.** *Soient $p \geqslant 5$ un nombre premier, $m \geqslant 2$ un entier et $E/\mathbb{Q}$ une courbe elliptique. Si E a bonne réduction en p, alors l'application de réduction restreinte à $E[m]$ est une injection dans $E(\mathbb{F}_p)[m]$.*

La proposition précédente montre que pour construire une courbe sur $\mathbb{F}_p$, où p est un nombre premier, avec des points de m-torsion, pour des petits entiers m, une méthode est de construire une courbe définie sur $\mathbb{Q}$ avec des points de m-torsion $\mathbb{Q}$-rationnels et de considérer la courbe réduite modulo p.

Enfin, terminons par l'un des théorèmes les plus importants concernant les courbes elliptiques définies sur les corps finis.

Théorème 1.1.17 (Théorème de Hasse)**.** *Soient q une puissance d'un nombre premier et $E/\mathbb{F}_q$ une courbe elliptique, alors*

$$|\#E(\mathbb{F}_q) - (q+1)| \leqslant 2\sqrt{q}.$$

1.1.2 Courbes de Montgomery et courbes d'Edwards

Il existe plusieurs familles de courbes elliptiques qui sont adaptées à différents usages. Dans le cas de l'algorithme ECM, les courbes les plus utilisées sont les courbes de Montgomery et les courbes d'Edwards.

Les courbes de Montgomery ont été définies pour la première fois dans [103].

Définition 1.1.18 (Courbes de Montgomery)**.** Soient K un corps de caractéristique différente de 2 et A et B deux éléments de K tels que $A \neq \pm 2$ et $B \neq 0$. La courbe définie par

$$BY^2Z = X^3 + AX^2Z + XZ^2 \tag{1.2}$$

est une courbe elliptique sur K. Une courbe de Montgomery est une courbe elliptique définie par une équation de la forme de l'équation (1.2).

Le discriminant d'une courbe de Montgomery est $16(A-2)(A+2)/B^6$. Le point à l'infini d'une courbe de Montgomery est le point $(0 : 1 : 0)$. En outre, le point $(0 : 0 : 1)$ est toujours rationnel et d'ordre 2. Comme pour les courbes sous forme de Weierstrass courte, l'opposé du point $(X : Y : Z)$ est le point $(X : -Y : Z)$. De plus, pour tout premier $p \geqslant 3$, l'ordre du groupe des points d'une courbe de Montgomery définie sur $\mathbb{F}_p$ est toujours divisible par 4. L'étude de la 2^k-torsion des courbes de Montgomery sera faite à la fin de cette section. La loi de groupe pour les courbes de Montgomery peut être trouvée dans [22, 103].

Généralisant un travail de Gauss et Euler, Edwards présente une loi de groupe pour les courbes définies par $x^2 + y^2 = c^2(1 + x^2y^2)$ en coordonnées affines (x, y) sur un corps de caractéristique

différente de 2 [53]. Il démontre aussi que, sur un corps algébriquement clos, toute courbe elliptique est équivalente à une courbe d'Edwards. Plusieurs variations sur le travail d'Edwards ont ensuite été publiées. En particulier, les courbes définies par $ax^2 + y^2 = 1 + dx^2y^2$ en coordonnées affines (x, y) sur un corps de caractéristique différente de 2 sont une variante tordue («twisted») des courbes définies par Edwards [17]. Ensuite une version complétée («completed») a été donnée dans [19]. C'est cette définition que nous allons utiliser, et ce que nous appellerons dans la suite les courbes d'Edwards sont en fait les courbes d'Edwards tordues complétées («completed twisted Edwards curves»).

Définition 1.1.19 (Courbes d'Edwards)**.** Soient K un corps de caractéristique différente de 2 et a et d deux éléments distincts et non nuls de K. La courbe définie par

$$aX^2T^2 + Y^2Z^2 = Z^2T^2 + dX^2Y^2 \tag{1.3}$$

est une courbe sur $\mathbb{P}^1(K) \times \mathbb{P}^1(K)$ en les coordonnées homogènes $((X : Z), (Y : T))$ isomorphe à une courbe elliptique sur K. Une courbe d'Edwards est une courbe elliptique définie par une équation de la forme de l'équation (1.3).

Le point à l'infini d'une courbe d'Edwards est le point $((0 : 1), (1 : 1))$. De plus, le point $((0 : 1), (-1 : 1))$ est toujours rationnel et d'ordre 2. L'opposé du point $((X : Z), (Y : T))$ est le point $((-X : Z), (Y : T))$. Pour passer à la version affine d'une courbe d'Edwards, il faut diviser l'équation définissant la courbe par T^2Z^2 et effectuer le changement de variables suivant : $x = X/Z$ et $y = Y/T$. L'équivalent affine du point projectif $((X : Z), (Y : T))$, lorsque $ZT \neq 0$, est $(X/Z, Y/T)$. Lorsque $Z = 0$ (resp. $T = 0$), l'équivalent affine du point projectif $((X : Z), (Y : T))$ sera noté $(\infty, Y/T)$ (resp. $(X/Z, \infty)$). L'équivalent projectif du point affine (x, y) est $((x : 1), (y : 1))$ et l'équivalent projectif du point affine (∞, y) (resp. (x, ∞)) est $((1 : 0), (y : 1))$ (resp. $((x : 1), (1 : 0))$. La loi de groupe des courbes d'Edwards peut être trouvée dans [19, 22].

Avant de faire l'étude de la 2^k-torsion des courbes d'Edwards, nous donnons une propriété importante liant les courbes de Montgomery et les courbes d'Edwards.

Théorème 1.1.20 ([17, Théorème 3.2])**.** *Soit K un corps de caractéristique différente de 2. Si E/K est une courbe de Montgomery, alors E est birationnellement équivalente sur K à la courbe d'Edwards avec les coefficients $d = (A-2)/B$ et $a = (A+2)/B$. Réciproquement, si E/K est une courbe d'Edwards, alors E est birationnellement équivalente sur K à la courbe de Montgomery avec les coefficients $A = 2(a + d)/(a - d)$ et $B = 4/(a - d)$.*

Deux courbes birationnellement équivalentes ont la même structure de groupe (les groupes des points K-rationnels des deux courbes sont isomorphes), mais diffèrent par la représentation des points et les formules des lois de groupe. Toutes les propriétés sur la structure du groupe de points d'une courbe d'Edwards (en particulier les propriétés de torsion) peuvent être traduites pour les courbes de Montgomery, et inversement.

Étude de la 2^k-torsion pour les courbes de Montgomery et d'Edwards

La propriété principale concernant la torsion des courbes de Montgomery a été montrée par Suyama [141] : soient $E/\mathbb{Q}$ une courbe de Montgomery et $p \geqslant 5$ un nombre premier de bonne réduction, alors $4 \mid \#E(\mathbb{F}_p)$. L'argument principal est que soit le discriminant est un résidu quadratique modulo p et alors tous les points de 2-torsion sont $\mathbb{F}_p$-rationnels, soit le discriminant est un non-résidu quadratique modulo p mais alors $\frac{A-2}{B}$ ou $\frac{A+2}{B}$ est un résidu quadratique modulo

p et cela permet de définir un point d'ordre 4. Le théorème 1.1.20 permet d'affirmer que c'est aussi vrai pour les courbes d'Edwards. Le théorème suivant donne plus de détails sur la 2^k-torsion des courbes de Montgomery et d'Edwards.

Théorème 1.1.21. *Soient $E/\mathbb{Q}$ une courbe d'Edwards (resp. une courbe de Montgomery) dont les coefficients sont a et d (resp. A et B) et p un premier de bonne réduction.*

1. *Si $p \equiv 3 \pmod 4$ et a/d (resp. A^2-4) est un résidu quadratique modulo p, alors*
$$E(\mathbb{F}_p)[4] = \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}.$$
2. *Si $p \equiv 1 \pmod 4$, a (resp. $(A+2)/B$) est un résidu quadratique modulo p (en particulier, si $a = \pm 1$) et a/d (resp. A^2-4) est un résidu quadratique modulo p, alors*
$$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z} \subset E(\mathbb{F}_p)[4].$$
3. *Si $p \equiv 1 \pmod 4$, a/d (resp. A^2-4) est un non-résidu quadratique modulo p et $a-d$ (resp. B) est un résidu quadratique modulo p, alors*
$$\mathbb{Z}/8\mathbb{Z} \subset E(\mathbb{F}_p)[8].$$

Démonstration. Le théorème 1.1.20 nous permet de démontrer le théorème uniquement dans le cas des courbes d'Edwards. Pour les courbes d'Edwards, une étude de la 2-, 4- et 8-torsion sur $\overline{\mathbb{Q}}$ est présentée dans la figure 1.2.

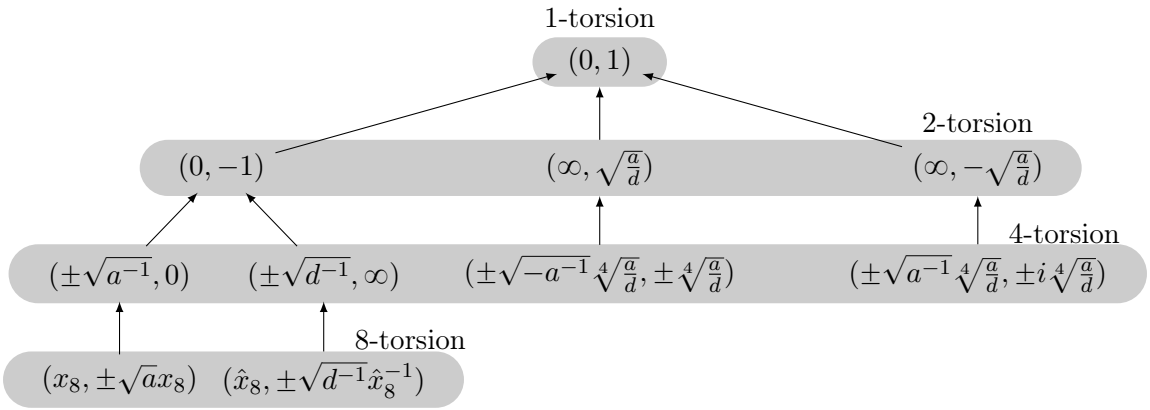

FIGURE 1.2 – Les coordonnées affines de tous les points de 1-, 2- et 4-torsion et de quelques points de 8-torsion pour les courbes d'Edwards. Pour les points de 8-torsion, x_8 et $\hat{x}_8$ vérifient $adx_8^4 - 2ax_8^2 + 1 = 0$ et $ad\hat{x}_8^4 - 2d\hat{x}_8^2 + 1 = 0$.

La figure 1.2 permet de démontrer toutes les affirmations. □

1.2 L'algorithme ECM

L'algorithme ECM a été décrit par H. W. Lenstra Jr en 1987 [97]. Nous allons d'abord donner une description de l'algorithme ECM (section 1.2.1), ensuite nous en étudierons rapidement la complexité (section 1.2.2) et finalement nous nous intéresserons aux familles de courbes elliptiques utilisées dans le cadre de l'algorithme ECM (section 1.2.3).

1.2.1 Description de l'algorithme ECM

L'algorithme ECM peut être vu comme une extension des algorithmes P-1 de Pollard [122] et P+1 de Williams [146]. L'algorithme P-1 (resp. P+1) parvient à trouver un facteur premier p d'un entier N si $p-1$ (resp. $p+1$) est le produit de petits facteurs premiers. L'algorithme ECM parvient à trouver un facteur premier p si le cardinal d'une courbe elliptique réduite modulo p (qui est de l'ordre de grandeur de p d'après le théorème de Hasse) est le produit de petits facteurs premiers. Le fait de pouvoir faire varier la courbe elliptique utilisée, et donc le cardinal de la courbe réduite, permet d'augmenter les chances de trouver un facteur. Comme les algorithmes P-1 et P+1, l'algorithme ECM est divisé en deux étapes. L'opération principale des deux étapes est le calcul de multiplications scalaires sur la courbe.

Dans la suite, nous noterons N l'entier pour lequel nous voulons trouver un facteur.

L'étape 1

Dans l'étape 1 d'ECM, nous voudrions calculer sur la courbe réduite modulo p, où p est un facteur de N, d'une courbe définie sur $\mathbb{Q}$. Mais p étant le facteur recherché, il n'est pas connu. Nous allons donc utiliser la loi de groupe de la courbe définie sur $\mathbb{Q}$, en considérant les formules modulo N, comme si N était un nombre premier. Cela revient à calculer en même temps sur toutes les courbes réduites modulo p, pour tous les facteurs premiers p de N. Une description de l'étape 1 d'ECM est donnée dans l'algorithme 1.1.

Algorithme 1.1 Étape 1 d'ECM

Entrée : L'entier N à factoriser, un entier $B_1 > 0$, une courbe elliptique $E/\mathbb{Q}$ sous forme de Weierstrass courte ou de Montgomery, telle que N et $6\Delta(E)$ sont premiers entre eux, et un point $P \in E(\mathbb{Q})$ d'ordre infini

Sortie : Un facteur de N ou ÉCHEC

1: Calculer s, le produit de toutes les puissances de nombres premiers inférieures à B_1 :

$$s = \prod_{\substack{\pi \leqslant B_1 \\ \pi \text{ premier}}} \pi^{\lfloor \log(B_1)/\log(\pi) \rfloor}$$

2: Calculer $(X : Y : Z) = sP$ en utilisant la loi de groupe sur E modulo N
3: $g = \text{pgcd}(Z, N)$
4: **si** $g > 1$ **alors**
5: **renvoyer** g
6: **sinon**
7: **renvoyer** ÉCHEC

Pour adapter ce pseudo-code aux courbes d'Edwards, il faut remplacer le point $(X : Y : Z)$ par le point $((X : Z), (Y : T))$ et $g = \text{pgcd}(Z, N)$ par $g = \text{pgcd}(X, N)$. Dans la suite de l'explication de l'algorithme ECM, nous considérerons uniquement les courbes sous forme de Weierstrass courte ou de Montgomery.

Le fait d'imposer au discriminant de la courbe d'être premier avec N permet de s'assurer que tous les facteurs premiers de N sont des nombres premiers de bonne réduction pour la courbe E. Si le pgcd est différent de 1, alors avec une grande probabilité, un facteur non trivial de N est trouvé. Nous demandons aussi que N ne soit pas divisible par 2 ou 3 pour s'assurer que nous ne serons jamais en caractéristique 2 ou 3. Cette exigence n'est pas contraignante car diviser l'entier

N par les plus grandes puissances de 2 et 3 le divisant est très simple.

L'étape 1 renvoie un entier divisible par le facteur premier p de N si l'ordre du point $\widetilde{P} \in E(\mathbb{F}_p)$ est B_1-ultrafriable. En effet, si l'ordre de $\widetilde{P}$ est B_1-ultrafriable, alors le point $\widetilde{sP} = s\widetilde{P} \in E(\mathbb{F}_p)$ est égal au point à l'infini $(0 : 1 : 0)$. Donc la coordonnée Z du point sP est égale à 0 modulo p et donc nous aurons $p \mid g = \text{pgcd}(Z, N)$. L'étape 1 nécessite que le point P ne soit pas un point de torsion sur $\mathbb{Q}$. En effet, si c'était le cas, alors, à cause de la proposition 1.1.16, pour tous les facteurs premiers de N, le point $\widetilde{P}$ de $E(\mathbb{F}_p)$ aurait le même ordre et l'étape 1 renverrait toujours N ou ÉCHEC. L'étape 1 peut aussi renvoyer N si B_1 est tel que l'ordre de $\widetilde{P}$ est B_1-ultrafriable pour tous les facteurs premiers p de N. Il faut dans ce cas diminuer la valeur de B_1 pour séparer les facteurs. Le seul cas où cela ne fonctionne pas est lorsque N est une puissance de nombre premier. Mais heureusement, il est facile de détecter si un entier est une puissance d'un nombre premier. L'ordre du point $\widetilde{P}$ dans $E(\mathbb{F}_p)$ divisant le cardinal de $E(\mathbb{F}_p)$, si $\#E(\mathbb{F}_p)$ est B_1-ultrafriable alors, quel que soit le point P choisi, son ordre sera B_1-ultrafriable. C'est donc les propriétés de friabilité de $\#E(\mathbb{F}_p)$ qui seront étudiées.

Le calcul d'un point d'ordre infini sur une courbe elliptique définie sur $\mathbb{Q}$ étant très coûteux, il est nécessaire de le fournir à l'algorithme de l'étape 1. Pour utiliser l'algorithme ECM, il faut donc savoir produire des courbes elliptiques pour lesquelles un point d'ordre infini sur $\mathbb{Q}$ est connu. En pratique, ce dont nous avons besoin est un triplet $(X, Y, Z) \in (\mathbb{Z}/N\mathbb{Z})^3$ qui vérifie l'équation de la courbe modulo N. Mais savoir calculer ce triplet requiert de savoir prendre une racine carrée modulo N, ce qui est un problème aussi dur que la factorisation de N. Donc la seule façon de faire est de prendre un point d'ordre infini sur $\mathbb{Q}$ et de le réduire modulo N.

Lorsque N est suffisamment grand, le temps de calcul de s est négligeable par rapport au temps de calcul de la multiplication scalaire. De plus le calcul de s n'est pas toujours nécessaire et peut être précalculé si la même valeur de B_1 est utilisée plusieurs fois. Il existe plusieurs méthodes pour calculer la multiplication scalaire sP, appelées méthodes d'exponentiation rapide : exponentiation par doublements et additions, chaînes d'additions [29], exponentiation par fenêtres glissantes [143], Montgomery Ladder [103], NAF [68], *etc* [20, 39]. En particulier, pour les courbes de Montgomery, il existe une formule, appelée addition–soustraction, qui permet d'additionner plus rapidement deux points lorsque la différence entre ces points est connue. Pour tirer parti de cela, il faut utiliser les chaînes de Lucas, et en particulier l'algorithme PRAC [102] de Montgomery. Le choix de l'algorithme de calcul de la multiplication scalaire est donc primordial car il détermine le nombre d'opérations modulaires qui seront effectuées. Le choix optimal peut être différent d'une forme de courbe à l'autre.

Il est possible d'utiliser des courbes en version affine pour l'étape 1 d'ECM. Mais dans ce cas les formules pour la loi de groupe ne sont plus polynomiales mais sont des fractions rationnelles. Des inversions modulo N doivent donc être calculées, mais N n'étant pas premier cela n'est pas toujours possible. Par contre si une inversion modulo N est impossible, cela signifie que le nombre qui doit être inversé n'est pas premier avec N et donc un facteur de N a été trouvé. Le calcul du pgcd final est donc remplacé par des calculs de pgcd à chaque fois qu'une inversion modulo N doit être effectuée. Une façon de réduire le coût important des inversions modulaires nécessaires à l'utilisation de courbes en version affine est d'utiliser plusieurs courbes en parallèle et d'utiliser l'algorithme d'inversion par lots de Montgomery [103].

L'étape 2

Comme pour les algorithmes P-1 et P+1, il existe une étape 2 pour ECM, décrite dans l'algorithme 1.2, qui est utilisée lorsque l'étape 1 renvoie ÉCHEC, pour augmenter les chances de trouver un facteur.

Algorithme 1.2 Étape 2 d'ECM

Entrée : Les mêmes entrées que pour l'étape 1 et un entier $B_2 > B_1$
Sortie : Un facteur de N ou ÉCHEC

1: Soit $Q = sP$ calculé par l'étape 1
2: **pour** tous les premiers p dans $]B_1, B_2]$ **faire**
3: Calculer $(x_p : y_p : z_p) = pQ$ en utilisant la loi de groupe sur E modulo N
4: $g_p = \text{pgcd}(z_p, N)$
5: **si** $g_p > 1$ **alors**
6: **renvoyer** g_p
7: **renvoyer** ÉCHEC

L'étape 2 renverra un facteur p de N si l'ordre du point $\widetilde{P}$ dans $E(\mathbb{F}_p)$ est (B_1, B_2)-ultrafriable, en particulier si $\#E(\mathbb{F}_p)$ est (B_1, B_2)-ultrafriable. L'étape 2 telle qu'elle est décrite ici peut être largement optimisée pour diminuer le nombre de multiplications scalaires et le nombre de pgcds calculés [103, 104, 148].

1.2.2 Complexité de l'algorithme ECM

Pour l'analyse de la complexité de l'algorithme ECM, nous pouvons considérer uniquement l'étape 1, car l'étape 2, bien que très importante en pratique, ne permet de gagner qu'un facteur logarithmique dans la complexité théorique. Nous considérerons aussi que tant que l'étape 1 retourne ÉCHEC, une nouvelle courbe elliptique est générée et l'étape 1 est relancée, jusqu'à ce qu'un facteur soit trouvé.

Remarquons tout d'abord que la probabilité que l'ordre du point de départ de l'étape 1 soit B_1-ultrafriable dans $E(\mathbb{F}_p)$ est supérieure ou égale à la probabilité que le cardinal de $E(\mathbb{F}_p)$ soit B_1-ultrafriable. Ensuite, d'après le théorème de Hasse, le cardinal d'une courbe elliptique sur $\mathbb{F}_p$, où p est un nombre premier, est compris entre $p + 1 - 2\sqrt{p}$ et $p + 1 + 2\sqrt{p}$. Comme il n'est pas facile d'estimer la probabilité de friabilité d'un entier dans cet intervalle, l'heuristique suivante est utilisée : le cardinal d'une courbe elliptique se comporte, du point de vue de la friabilité, comme un entier dans l'intervalle $[1, p]$. Nous verrons dans la section 1.3 comment étudier plus précisément quelques propriétés de divisibilité du cardinal d'une courbe elliptique. Cependant, cette heuristique est faite pour pouvoir utiliser le théorème de Canfield–Erdős–Pomerance [34]. Ce théorème nous dit que, en notant $\Psi(x, y)$ le nombre d'entiers positifs inférieurs à x sans facteur premier plus grand que y, pour tout $\epsilon > 0$, $\Psi(x, x^{1/u}) = xu^{-u(1+o(1))}$ pour $u \to \infty$ uniformément pour $x \geqslant u^{u(1+\epsilon)}$. En combinant ces deux observations, il peut être montré que pour trouver un facteur premier p d'un entier N, il faut choisir la taille de B_1 en $\mathrm{L}_p\left(\frac{1}{2}, \frac{1}{\sqrt{2}}\right)$, et que le nombre de courbes à tester est en $\mathrm{L}_p\left(\frac{1}{2}, \frac{1}{\sqrt{2}}\right)$. Comme la complexité de l'étape 1 pour une courbe est $\mathrm{O}(B_1 M(\log(N)))$, où $M(\log(N))$ est la coût d'une multiplication modulo N, la complexité totale pour trouver un facteur p de N est

$$\mathrm{O}\left(\mathrm{L}_p\left(\frac{1}{2}, \sqrt{2}\right) M(\log(N))\right) = \mathrm{O}\left(\exp\left(\left(\sqrt{2} + \mathrm{o}(1)\right)(\log p)^{\frac{1}{2}} (\log\log p)^{\frac{1}{2}}\right) M(\log(N))\right).$$

Une étude plus précise de la complexité de l'étape 1 est donnée dans [97].

En pratique, comme a priori ni le facteur p ni sa taille ne sont connus, de plus en plus de courbes elliptiques sont essayées, avec des B_1 de plus en plus grands jusqu'à ce qu'un facteur soit trouvé.

1.2.3 Familles de courbes elliptiques utilisées pour ECM

Les courbes les plus utilisées dans les implémentations d'ECM sont les courbes de Montgomery et d'Edwards. Par exemple, le logiciel GMP-ECM [61] utilise les courbes de Montgomery (en se basant sur GMP [60] pour l'arithmétique modulaire) et le logiciel EECM-MPFQ [54] utilise les courbes d'Edwards (en se basant sur $\texttt{mp}\mathbb{F}_q$ [110] pour l'arithmétique modulaire). Ces courbes sont utilisées car elles possèdent une arithmétique rapide et des sous-familles avec de bonnes propriétés de torsion sont connues.

Pour pouvoir utiliser plusieurs courbes elliptiques dans l'algorithme ECM, nous voulons une famille infinie de courbes elliptiques pour lesquelles un point d'ordre infini est connu. En pratique, nous chercherons des familles pour lesquelles il existe une paramétrisation rationnelle (c'est-à-dire, à chaque rationnel, sauf un nombre fini d'entre eux, est associé une courbe elliptique et un point d'ordre infini) ou une paramétrisation elliptique (c'est-à-dire, à chaque point d'une courbe elliptique, sauf un nombre fini d'entre eux, est associé une courbe elliptique et un point d'ordre infini). Une paramétrisation rationnelle correspond à une famille paramétrée par une courbe de genre 0 et une paramétrisation elliptique correspond à une famille paramétrée par une courbe de genre 1. Le théorème de Faltings, donné ci-dessous, montre qu'il n'est pas nécessaire de rechercher des familles paramétrées par des courbes de genre supérieur.

Théorème 1.2.1 (Faltings)**.** *Une courbe définie sur $\mathbb{Q}$ de genre supérieur ou égal à 2 ne possède qu'un nombre fini de points $\mathbb{Q}$-rationnels.*

Pour être adaptée à ECM, une courbe elliptique doit posséder des bonnes propriétés de torsion pour augmenter la probabilité de friabilité du cardinal de la courbe modulo un nombre premier. Par exemple, les courbes de Montgomery et d'Edwards ont l'avantage, par rapport aux courbes sous forme de Weierstrass courte générique, d'imposer un facteur 4 dans le cardinal de la courbe modulo tous les nombres premiers. Cela augmente les probabilités de friabilité.

Montgomery donne dans sa thèse [104] plusieurs sous-familles de courbes de Montgomery possédant de bonnes propriétés de torsion : une avec $\mathbb{Z}/12\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$ et une autre avec $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$ (le plus grand groupe de torsion possible d'après le théorème de Mazur). De plus, dans [103], il décrit la sous-famille donnée par l'équation $B = 4A + 10$ pour laquelle le point $(2 : 1 : 1)$ est toujours d'ordre infini. Cette famille ne possède pas de meilleures propriétés de torsion mais une meilleure arithmétique, une multiplication modulo N pouvant être remplacée par une multiplication par 2 dans la formule d'addition–soustraction. Enfin, Suyama [141] a décrit une famille de courbes de Montgomery pour lesquelles il existe sur $\mathbb{Q}$ un point d'ordre 3 et un point d'ordre infini. Une courbe de Suyama est une courbe de Montgomery dont les coefficients $A, B \in \mathbb{Q}$ sont solutions du système d'équations suivant :

$$\begin{cases} P_3(x_3) = 0, \quad By_3^2 = x_3^3 + Ax_3^2 + x_3 & \text{(point de 3-torsion)} \\ k = \dfrac{y_3}{y_\infty}, \quad k^2 = \dfrac{x_3^3 + Ax_3^2 + x_3}{x_\infty^3 + Ax_\infty^2 + x_\infty} & \text{(point d'ordre infini)} \\ x_\infty = x_3^3. & \text{(équation de Suyama)} \end{cases} \tag{1.4}$$

où x_3, y_3, k, x_∞ et y_∞ sont des rationnels. Les solutions de (1.4) peuvent être paramétrées par un rationnel, souvent appelé σ (cf. [141, 148]). Pour tout $\sigma \in \mathbb{Q} \setminus \{ 0, \pm 1, \pm 3, \pm 5, \pm 5/3 \}$, la courbe de Suyama associée a un rang positif sur $\mathbb{Q}$ et un point d'ordre 3 défini sur $\mathbb{Q}$. Une courbe de Suyama étant une courbe de Montgomery, elle possède au moins un point de 2-torsion sur $\mathbb{Q}$ et donc son groupe de torsion sur $\mathbb{Q}$ contient $\mathbb{Z}/6\mathbb{Z}$. De plus, le cardinal d'une courbe de Suyama sur $\mathbb{F}_p$ est divisible par 12, pour tous les nombres premiers p.

Dans [19], les sous-familles des courbes d'Edwards correspondant aux sous-familles décrites dans la thèse de Montgomery, aux courbes de Suyama et aux courbes d'Atkin–Morain [4] sont présentées. D'autres sous-familles pour les courbes d'Edwards sont présentées dans [18].

1.3 Propriétés galoisiennes et probabilités de divisibilité

Dans cette section, nous allons étudier certaines propriétés galoisiennes des courbes elliptiques dans le but de donner une méthode pour calculer la probabilité que l'ordre du groupe d'une courbe elliptique réduite modulo un nombre premier aléatoire soit divisible par une certaine puissance d'un nombre premier.

1.3.1 Propriétés galoisiennes des points de torsion des courbes elliptiques

L'outil principal utilisé dans cette section est le théorème de Chebotarev. Quelques définitions sur les corps de nombres sont nécessaires afin de pouvoir présenter ce théorème.

Définition 1.3.1. Soient K une extension galoisienne finie de $\mathbb{Q}$ et p un nombre premier. Si $\mathfrak{p}$ est un idéal premier au-dessus de p alors le corps résiduel de $\mathfrak{p}$ est noté $k_\mathfrak{p}$, l'automorphisme de Frobenius du corps $k_\mathfrak{p}$ est noté $\phi_\mathfrak{p}$, le groupe de décomposition de $\mathfrak{p}$, défini comme le sous-groupe de $\mathrm{Gal}(K/\mathbb{Q})$ qui stabilise $\mathfrak{p}$, est noté $\mathrm{Dec}(\mathfrak{p})$, et le morphisme canonique de $\mathrm{Dec}(\mathfrak{p})$ dans $\mathrm{Gal}(k_\mathfrak{p}/\mathbb{F}_p)$ est noté $\alpha^{(\mathfrak{p})}$. Finalement, $\mathrm{Frobenius}(p)$ est défini par

$$\mathrm{Frobenius}(p) = \bigcup_{\mathfrak{p}|p} (\alpha^{(\mathfrak{p})})^{-1}(\phi_\mathfrak{p}).$$

Dans la suite nous parlerons de probabilité sur l'ensemble des nombres premiers. De façon plus précise, nous dirons qu'un ensemble S de nombres premiers admet *une densité naturelle égale à* δ, que nous noterons $\mathrm{Prob}(S) = \delta$, si la limite

$$\lim_{x\to\infty} \frac{\#(S \bigcap \mathcal{P}(x))}{\#\mathcal{P}(x)}$$

existe et est égale à δ, où $\mathcal{P}(x)$ est l'ensemble des nombres premiers inférieurs ou égaux à x. Si $\mathcal{E}(p)$ est une propriété qui est définie pour tous les nombres premiers sauf pour un nombre fini (donc de densité nulle), quand nous écrirons $\mathrm{Prob}(\mathcal{E}(p))$, nous exclurons implicitement les premiers pour lesquels $\mathcal{E}(p)$ ne peut pas être définie. Par exemple, quand nous calculerons des probabilités portant, pour une courbe elliptique E, sur $\#E(\mathbb{F}_p)$ nous exclurons implicitement tous les premiers de mauvaise réduction.

Théorème 1.3.2 (Chebotarev, [116])**.** *Soient K une extension galoisienne finie de $\mathbb{Q}$ et H une classe de conjugaison de* $\mathrm{Gal}(K/\mathbb{Q})$*. Alors*

$$\mathrm{Prob}(\mathrm{Frobenius}(p) = H) = \frac{\#H}{\#\,\mathrm{Gal}(K/\mathbb{Q})}.$$

Avant de pouvoir appliquer le théorème de Chebotarev au cas des courbes elliptiques, nous devons définir quelques notations. Tout d'abord, considérons, pour un corps K, une courbe elliptique E/K et un entier $m \geqslant 2$, l'application suivante :

$$\begin{array}{rccc} \iota_m^K\colon & \mathrm{Gal}(K(E[m])/K) & \to & \mathrm{Aut}(E(\overline{K})[m]) \\ & \sigma & \mapsto & P \in E(\overline{K})[m] \mapsto \sigma(P) \end{array}$$

où $\sigma(P)$ dénote l'application de σ à toutes les coordonnées du point P. Le fait que $\sigma(P)$ appartient à $E(\overline{K})[m]$ pour un $P \in E(\overline{K})[m]$, et donc que $\iota_m^K(\sigma) \in \mathrm{Aut}(E(\overline{K})[m])$, se démontre en utilisant le fait que les coordonnées des points de m-torsion peuvent être décrites comme les racines des polynômes de division et que la loi de groupe s'écrit grâce à des fonctions rationnelles. De plus, il est facile de montrer que ι_m^K est un morphisme de groupe injectif. Par ailleurs, lorsque la proposition 1.1.9 s'applique, choisir des générateurs pour $E(\overline{K})[m]$ induit un isomorphisme ψ_m^K entre $\mathrm{Aut}(E(\overline{K})[m])$ et $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$. Il existe donc un morphisme injectif ρ_m^K de $\mathrm{Gal}(K(E[m])/K)$ dans $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ défini par $\psi_m^K \circ \iota_m^K$. Dans le cas où $K = \mathbb{Q}$, $\iota_m^{\mathbb{Q}}$ et ρ_m^K seront simplement notés ι_m et ρ_m respectivement. Dans le cas où $K = \mathbb{F}_p$, avec p un nombre premier, $\iota_m^{\mathbb{F}_p}$ sera simplement noté $\iota_m^{(p)}$. Enfin, si p est un nombre premier de bonne réduction pour une courbe elliptique $E/\mathbb{Q}$ tel que $p \nmid m$, alors il existe un isomorphisme canonique $r_m^{(p)}$ entre $\mathrm{Aut}(E(\overline{\mathbb{Q}})[m])$ et $\mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])$ (cf. proposition 1.1.9 et [138, proposition VII.3.1]).

Remarque 1.3.3. L'existence du morphisme injectif ρ_m montre que le cardinal du groupe de Galois $\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$ est borné par le cardinal de $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$. Or pour tout nombre premier π, le cardinal de $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$ est égal à $(\pi-1)^2(\pi+1)\pi$, et pour tout entier k positif non nul, $\#\,\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z}) = \pi^4 \#\,\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$.

Pour tout $g \in \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$, nous noterons $\mathrm{Fix}(g)$ l'ensemble des vecteurs de $(\mathbb{Z}/m\mathbb{Z})^2$ fixés par g, c'est-à-dire $\mathrm{Fix}(g) = \left\{ v \in (\mathbb{Z}/m\mathbb{Z})^2 \;\middle|\; g(v) = v \right\}$. Si C est une classe de conjugaison d'éléments de $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$, nous noterons $\mathrm{Fix}(C)$ la classe d'isomorphismes du groupe $\mathrm{Fix}(g)$, pour g quelconque dans C ; cette classe d'isomorphismes ne dépend pas du choix de g. Nous utiliserons des notations analogues pour les groupes $\mathrm{Aut}(E(\overline{\mathbb{Q}})[m])$ et $\mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])$.

Théorème 1.3.4. *Soient $m \geqslant 2$ un entier, $E/\mathbb{Q}$ une courbe elliptique pour laquelle nous noterons $K = \mathbb{Q}(E[m])$ et T un sous-groupe de $\mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$. Alors,*

1. $\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T) = \dfrac{\#\left\{ g \in \rho_m(\mathrm{Gal}(K/\mathbb{Q})) \mid \mathrm{Fix}(g) \simeq T \right\}}{\#\,\mathrm{Gal}(K/\mathbb{Q})}$.
2. *De plus, soient a et n deux entiers premiers entre eux et strictement positifs tels que $a \leqslant n$, ζ_n une racine primitive $n^{\text{ième}}$ de l'unité et $G_a = \left\{ \sigma \in \mathrm{Gal}(K(\zeta_n)/\mathbb{Q}) \mid \sigma(\zeta_n) = \zeta_n^a \right\}$. Alors :*

$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T \mid p \equiv a \bmod n) = \frac{\#\left\{ \sigma \in G_a \;\middle|\; \mathrm{Fix}(\rho_m(\sigma_{|K})) \simeq T \right\}}{\#G_a}.$$

Démonstration. Soit p un nombre premier tel que p ne divise pas m et E a bonne réduction en p. Soit $\mathfrak{p}$ un idéal premier de K au-dessus de p. Dans la suite de la preuve, nous noterons $H = \left\{ \sigma \in \mathrm{Gal}(K/\mathbb{Q}) \mid \mathrm{Fix}(\iota_m(\sigma)) \simeq T \right\}$. Tout d'abord, remarquons que si ϕ_p dénote le Frobenius dans $\mathrm{Gal}(\mathbb{F}_p(E[m])/\mathbb{F}_p)$, alors $E(\mathbb{F}_p)[m] = \mathrm{Fix}(\iota_m^{(p)}(\phi_p))$. Étant donné que le diagramme suivant est commutatif

$$\begin{array}{ccccc}
\mathrm{Dec}(\mathfrak{p}) & \hookrightarrow & \mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}) & \xrightarrow{\iota_m} & \mathrm{Aut}(E(\overline{\mathbb{Q}})[m]) \\
\downarrow{\scriptstyle \alpha^{(p)}} & & & & \downarrow{\scriptstyle r_m^{(p)}} \\
\mathrm{Gal}(k_{\mathfrak{p}}/\mathbb{F}_p) & \xrightarrow{\sim} & \mathrm{Gal}(\mathbb{F}_p(E[m])/\mathbb{F}_p) & \xrightarrow{\iota_m^{(p)}} & \mathrm{Aut}(E(\overline{\mathbb{F}_p})[m])
\end{array}$$

et que $\mathrm{Frobenius}(p) \subset \mathrm{Gal}(K/\mathbb{Q})$ est la classe de conjugaison générée par $(\alpha^{(\mathfrak{p})})^{-1}(\phi_{\mathfrak{p}})$, nous avons $E(\mathbb{F}_p)[m] \simeq \mathrm{Fix}(\iota_m(\mathrm{Frobenius}(p)))$.

Si nous décomposons H en une union disjointe de ℓ classes de conjugaison $C_1, \ldots, C_\ell$, alors, $\mathrm{Fix}(\iota_m(\mathrm{Frobenius}(p))) \simeq T$ si et seulement si $\mathrm{Frobenius}(p)$ est l'un des C_i. En appliquant le théorème de Chebotarev, nous obtenons :

$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T) = \sum_{i=1}^{\ell} \mathrm{Prob}(\mathrm{Frobenius}(p) = C_i) = \sum_{i=1}^{\ell} \frac{\#C_i}{\#\,\mathrm{Gal}(K/\mathbb{Q})} = \frac{\#H}{\#\,\mathrm{Gal}(K/\mathbb{Q})}.$$

Ceci prouve l'affirmation 1.

En utilisant des arguments similaires, prouver l'affirmation 2 peut se ramener à évaluer

$$\frac{\mathrm{Prob}(\mathrm{Frobenius}(p) \in \{ C_1, \ldots, C_\ell \}, p \equiv a \bmod n)}{\mathrm{Prob}(p \equiv a \bmod n)}.$$

Soient p un nombre premier et $\mathfrak{p}$ un idéal premier, comme dans la première partie de la preuve, et $\mathfrak{P}$ un idéal premier de $K(\zeta_n)$ au-dessus de $\mathfrak{p}$. De plus, soient $\tilde{C}_1, \ldots, \tilde{C}_{\tilde{\ell}}$ les classes de conjugaison de $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ qui sont dans les pré-images de $C_1, \ldots, C_\ell$ et dont les éléments σ satisfont $\sigma(\zeta_n) = \zeta_n^a$. Comme $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ envoie ζ_n sur une des racines primitives $n^{\text{ièmes}}$ de l'unité, nous savons que pour $\sigma \in (\alpha^{(\mathfrak{P})})^{-1}(\phi_{\mathfrak{P}})$, il existe un entier b tel que $\sigma(\zeta_n) = \zeta_n^b$. En combinant cela avec le fait que $\sigma(x) \equiv x^p \bmod \mathfrak{P}$, nous obtenons que $\zeta_n^b \equiv \zeta_n^p \bmod \mathfrak{P}$. En excluant le nombre fini de premiers divisant les normes de $\zeta_n^c - 1$ pour $c = 1, \ldots, n-1$, nous obtenons $b \equiv p \bmod n$. Comme $\mathrm{Frobenius}(K(\zeta_n), p)$, la classe de conjugaison de Frobenius pour $K(\zeta_n)$, est la pré-image de $\mathrm{Frobenius}(p)$, les arguments ci-dessus permettent de montrer l'égalité suivante :

$$\mathrm{Prob}\left(\mathrm{Frobenius}(p) \in \{ C_1, \ldots, C_\ell \}, p \equiv a \bmod n\right) = \mathrm{Prob}\left(\mathrm{Frobenius}(K(\zeta_n), p) \in \left\{ \tilde{C}_1, \ldots, \tilde{C}_{\tilde{\ell}} \right\}\right).$$

Des considérations similaires pour le dénominateur $\mathrm{Prob}(p \equiv a \bmod n)$ permettent de compléter la preuve de l'affirmation 2. □

Remarque 1.3.5. En utilisant les notations du théorème précédent, si $[K(\zeta_n) : \mathbb{Q}(\zeta_n)] = [K : \mathbb{Q}]$, alors

$$\mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T \mid p \equiv a \bmod n) = \mathrm{Prob}(E(\mathbb{F}_p)[m] \simeq T)$$

pour a premier avec n. En effet, d'après la théorie de Galois,

$$\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})/\,\mathrm{Gal}(K(\zeta_n)/K) \simeq \mathrm{Gal}(K/\mathbb{Q})$$

via l'application qui à $\overline{\sigma}$ associe la restriction $\sigma_{|K}$. Comme $[K(\zeta_n) : \mathbb{Q}(\zeta_n)] = [K : \mathbb{Q}]$, alors $[K(\zeta_n) : K] = \varphi(n)$ et donc chaque élément σ de $\mathrm{Gal}(K/\mathbb{Q})$ se prolonge d'une seule façon en un élément de $\mathrm{Gal}(K(\zeta_n)/\mathbb{Q})$ qui vérifie $\sigma(\zeta_n) = \zeta_n^a$. Notons que pour $n \in \{ 3, 4 \}$, la condition est équivalente à $\zeta_n \notin K$.

Les familles de courbes elliptiques décrites par Brier et Clavier [30], qui ont été construites spécifiquement pour factoriser des entiers N tels que le $n^{\text{ième}}$ polynôme cyclotomique ait des racines modulo tous les facteurs premiers de N, modifient $[K(\zeta_n) : \mathbb{Q}(\zeta_n)]$ en imposant un grand sous-groupe de torsion sur $\mathbb{Q}(\zeta_n)$.

Un cas particulier important du théorème précédent est donné dans le corollaire suivant.

Corollaire 1.3.6. *Soient $E/\mathbb{Q}$ une courbe elliptique et π un nombre premier, alors,*

$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z}) = \frac{\#\{\, g \in \rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})) \mid \det(g - \mathrm{Id}) = 0, g \neq \mathrm{Id} \,\}}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})},$$
$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z}) = \frac{1}{\#\,\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})}.$$

Exemple 1.3.7. Dans cet exemple, nous allons nous intéresser au cas où l'application ρ_π est surjective, c'est-à-dire au cas où $\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})$ est isomorphe à $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$. L'objectif est de compter le nombre de matrices de $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$ qui ont 1 comme valeur propre et qui sont différentes de l'identité. Ces matrices sont conjuguées soit à une matrice de la forme $\left(\begin{smallmatrix} 1 & 0 \\ 0 & a \end{smallmatrix}\right)$, avec $a \in (\mathbb{Z}/\pi\mathbb{Z})\backslash\{\,0,1\,\}$, soit à $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$. Il existe $\pi-2$ matrices de la forme $\left(\begin{smallmatrix} 1 & 0 \\ 0 & a \end{smallmatrix}\right)$, avec $a \in (\mathbb{Z}/\pi\mathbb{Z})\backslash\{\,0,1\,\}$, et chacune de ces matrices a $\pi(\pi+1)$ matrices conjuguées. La matrice $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$ a π^2-1 matrices conjuguées. Il existe donc $(\pi+1)(\pi^2-\pi-1)$ matrices dans $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$ qui ont 1 comme valeur propre et sont différentes de l'identité. D'où, si $E/\mathbb{Q}$ est une courbe elliptique et π un nombre premier tel que l'application ρ_π est surjective, alors, le corollaire 1.3.6 nous donne

$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z}) = \frac{(\pi+1)(\pi^2-\pi-1)}{\pi(\pi+1)(\pi-1)^2} = \frac{\pi^2-\pi-1}{\pi(\pi-1)^2},$$
$$\mathrm{Prob}(E(\mathbb{F}_p)[\pi] \simeq \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z}) = \frac{1}{\pi(\pi+1)(\pi-1)^2}.$$

Exemple 1.3.8. Nous allons calculer les probabilités du corollaire 1.3.6 pour les courbes elliptiques $E_1/\mathbb{Q}\colon y^2 = x^3 + 5x + 7$ et $E_2/\mathbb{Q}\colon y^2 = x^3 - 11x + 14$ et les premiers $\pi = 3$ et $\pi = 5$. La courbe E_1 illustre le cas générique, tandis que la courbe E_2 possède un groupe de Galois spécial. Avec Sage [130], il peut être vérifié que $[\mathbb{Q}(E_1[3]) : \mathbb{Q}] = 48$ et $\#\,\mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z}) = 48$. Nous pouvons donc conclure que $\rho_3(\mathrm{Gal}(\mathbb{Q}(E_1[3])/\mathbb{Q})) = \mathrm{GL}_2(\mathbb{Z}/3\mathbb{Z})$. L'exemple 1.3.7 nous donne les probabilités suivantes :

$$\mathrm{Prob}(E_1(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z}) = \frac{20}{48} \quad \text{et} \quad \mathrm{Prob}(E_1(\mathbb{F}_p)[3] \simeq \mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}) = \frac{1}{48}.$$

Les autres probabilités ont été calculées en utilisant la même méthode, et sont comparées dans la table 1.1 avec les valeurs expérimentales.

La différence relative entre les valeurs théoriques et expérimentales n'est jamais supérieure à 0,4 %. De plus, il est intéressant de remarquer que réduire le groupe de Galois n'augmente pas nécessairement la probabilité, comme le montre le cas $\pi = 3$.

1.3.2 Calcul explicite de $\mathbb{Q}(E[m])$ et $\rho_m(\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}))$ pour les puissances de nombre premier

Nous allons maintenant montrer comment $\mathbb{Q}(E[m])$ et $\rho_m(\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q}))$ peuvent être calculés lorsque m est une puissance d'un nombre premier et ensuite nous donnerons des résultats de Serre qui montrent que dans la majorité des cas, l'application ρ est surjective. L'algorithme 1.3 permet de calculer, pour une courbe elliptique $E/\mathbb{Q}$, l'extension $\mathbb{Q}(E[\pi])$, pour un nombre premier π plus grand ou égal à 3. Cette méthode permet aussi de montrer que $\mathbb{Q}(E[\pi])$ est une extension galoisienne. L'outil principal de cet algorithme est le polynôme de division. Remarquons que les polynômes de division dépendent de la forme de la courbe elliptique (Weierstrass, Montgomery, Edwards, *etc*) mais que le groupe de Galois $\mathrm{Gal}(\mathbb{Q}(E[m])/\mathbb{Q})$ n'en dépend pas et peut être calculé en utilisant les polynômes de division de la définition 1.1.13.

π	T	d_1	$\mathrm{P_{th}}(E_1,\pi,T)$ $\mathrm{P_{exp}}(E_1,\pi,T)$	d_2	$\mathrm{P_{th}}(E_2,\pi,T)$ $\mathrm{P_{exp}}(E_2,\pi,T)$
3	$\mathbb{Z}/3\mathbb{Z}\times\mathbb{Z}/3\mathbb{Z}$	48	$\frac{1}{48}\approx 0.02083$ 0.02082	16	$\frac{1}{16}=0.06250$ 0.06245
3	$\mathbb{Z}/3\mathbb{Z}$	48	$\frac{20}{48}\approx 0.4167$ 0.4165	16	$\frac{4}{16}=0.2500$ 0.2501
5	$\mathbb{Z}/5\mathbb{Z}\times\mathbb{Z}/5\mathbb{Z}$	480	$\frac{1}{480}\approx 0.002083$ 0.002091	32	$\frac{1}{32}=0.03125$ 0.03123
5	$\mathbb{Z}/5\mathbb{Z}$	480	$\frac{114}{480}\approx 0.2375$ 0.2373	32	$\frac{10}{32}=0.3125$ 0.3125

TABLE 1.1 – Valeurs théoriques et expérimentales pour $\mathrm{P}(E,\pi,T) := \mathrm{Prob}(E(\mathbb{F}_p)[\pi]\simeq T)$ pour les courbes elliptiques E_1 et E_2 de l'exemple 1.3.8, pour plusieurs nombres premiers π et sous-groupes T. Les valeurs théoriques ont été obtenues en utilisant le corollaire 1.3.6, et les valeurs expérimentales ont été calculées en utilisant tous les nombres premiers inférieurs à 2^{25}. Les colonnes d_1 et d_2 donnent le degré des corps de nombres $\mathbb{Q}(E_1[\pi])$ et $\mathbb{Q}(E_2[\pi])$ respectivement.

Algorithme 1.3 Calcul de $\mathbb{Q}(E[\pi])$

Entrée : Une courbe elliptique $E/\mathbb{Q}$: $y^2=x^3+ax+b$ sous forme de Weierstrass courte affine et un nombre premier $\pi\geqslant 3$

Sortie : L'extension $\mathbb{Q}(E[\pi])$

1: Calculer une première extension de $\mathbb{Q}$ *via* un facteur irréductible de degré plus grand que 1 du polynôme de division P_π pour obtenir un corps de nombre F_1 où P_π a une racine α_1.
2: Soit $f_2(y)=y^2-(\alpha_1^3+a\alpha_1+b)\in F_1[y]$.
3: Calculer l'extension F_2 de F_1 *via* f_2. Dans F_2, f_2 a une racine β_1 et $M_1=(\alpha_1,\beta_1)$ est un point de π-torsion de $E(F_2)$, donc $\mathbb{Z}/\pi\mathbb{Z}\subset E(F_2)[\pi]$. Dans F_2, P_π a $\frac{\pi-1}{2}$ racines triviales correspondant aux coordonnées x des multiples de M_1. Soit $P_\pi^{(F_2)}\in F_2[x]$ le polynôme P_π divisé par toutes ces racines triviales.
4: Calculer F_3 l'extension de F_2 *via* un facteur irréductible de $P_\pi^{(F_2)}$. Soit α_2 la nouvelle racine de $P_\pi^{(F_2)}$ dans F_3.
5: $f_4(y)=y^2-(\alpha_2^3+a\alpha_2+b)\in F_3[y]$.
6: **renvoyer** F_4, l'extension de F_3 *via* f_4.

Le cas des puissances de nombre premier π^k, avec $k \geqslant 2$, peut être traité récursivement. Une fois que $\mathbb{Q}(E[\pi^{k-1}])$ a été calculé, $\mathbb{Q}(E[\pi^k])$ peut être calculé en utilisant les mêmes étapes que l'algorithme 1.3 mais en utilisant $\mathbb{Q}(E[\pi^{k-1}])$ au lieu de $\mathbb{Q}$ comme corps de base, $P_{\pi^k}^{\text{new}}$ au lieu de P_π et en considérant, à l'étape 3, les coordonnées x des points $\{ P + M_1 \mid P \in E[\pi^{k-1}] \}$ comme racines triviales. L'algorithme 1.3 peut aussi être adapté pour le cas $\pi = 2$ et pour les puissances de 2.

L'algorithme 1.3 nous permet de calculer $\mathbb{Q}(E[\pi^k])$ comme une tour d'extensions. Ensuite, il est facile d'obtenir le degré de l'extension $\mathbb{Q}(E[\pi^k])/\mathbb{Q}$ et un élément primitif. Identifier $\rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))$ (à conjugaison près) est facile quand il n'y a qu'un sous-groupe (à conjugaison près) de $\mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ avec la bonne taille. Lorsque ce n'est pas le cas, nous fixons des générateurs de $E(\overline{\mathbb{Q}})[\pi^k]$ pour vérifier pour chaque $g \in \mathrm{GL}_2(\mathbb{Z}/m\mathbb{Z})$ si g donne un automorphisme sur $\mathbb{Q}(E[\pi^k])$. En pratique, la partie coûteuse de cet algorithme est la factorisation des polynômes avec coefficients dans des corps de nombres.

Un algorithme beaucoup plus rapide pour calculer $\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})$ a été proposé par Sutherland [140]. Lors de nos travaux sur l'article [10] sur lequel se base ce chapitre, cet algorithme n'était pas connu des auteurs mais il nous a été très utile lors de la révision de l'article, car il nous a permis d'accélérer les calculs de nos exemples.

En pratique, nous avons observé qu'en général $P_\pi, f_2, P_\pi^{(F_2)}$ et f_4 sont irréductibles. Lorsque c'est le cas, comme $\deg(P_\pi) = \frac{\pi^2-1}{2}$, le degré de l'extension F_4 est

$$\frac{\pi^2-1}{2} \times 2 \times \frac{\pi^2-\pi}{2} \times 2 = (\pi-1)^2(\pi+1)\pi.$$

D'après la remarque 1.3.3, $\#\,\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z}) = (\pi-1)^2(\pi+1)\pi$, donc en général, nous nous attendons à avoir

$$\rho_\pi(\mathrm{Gal}(\mathbb{Q}(E[\pi])/\mathbb{Q})) = \mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z}).$$

Nous avons aussi observé qu'en général le degré de l'extension $\mathbb{Q}(E[\pi^k])/\mathbb{Q}(E[\pi^{k-1}])$ est π^4.

Le théorème ci-dessous montre que les observations ci-dessus sont presque toujours vraies. C'est une réécriture des assertions (1) et (6) de l'introduction de [135].

Théorème 1.3.9 (Serre)**.** *Soit $E/\mathbb{Q}$ une courbe sans multiplication complexe.*

1. *Pour tous les premiers π, la suite des indices*

$$[\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}) : \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))], \quad k \geqslant 1$$

   est croissante et bornée par une constante dépendant de E et π.

2. *Pour tous les premiers π, sauf un nombre fini dépendant de E, et pour tout $k \geqslant 1$,*

$$\rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})) = \mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}).$$

Ce théorème nous permet de prouver que l'exposant de Serre défini ci-dessous existe.

Définition 1.3.10 (Exposant de Serre)**.** Soient $E/\mathbb{Q}$ une courbe elliptique sans multiplication complexe, π un nombre premier et $k \geqslant 1$ un entier. Définissons $I(E, \pi, k)$ par

$$I(E, \pi, k) = [\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z}) : \rho_{\pi^k}(\mathrm{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))].$$

De plus, appelons l'exposant de Serre l'entier

$$n(E, \pi) = \min\{ n \in \mathbb{N}^* \mid \forall k \geqslant n, I(E, \pi, k+1) = I(E, \pi, k) \}.$$

Dans [136], Serre a montré que dans certains cas, il peut être prouvé que $I(E, \pi, k) = 1$ pour tout entier $k \geqslant 1$. En effet, Serre a prouvé que si E est une courbe elliptique sans multiplication complexe, si $\pi \geqslant 5$ est un nombre premier et si l'application ρ_π est surjective (c'est-à-dire, si $I(E, \pi, 1) = 1$) alors l'application ρ_{π^k} est surjective, pour tout entier $k \geqslant 1$ (c'est-à-dire, $I(E, \pi, k) = 1$, pour tout entier $k \geqslant 1$). Pour avoir le même résultat pour $\pi = 2$ (resp. $\pi = 3$), il doit être supposé que les applications ρ_2, ρ_4 et ρ_8 sont surjectives (resp. ρ_3 et ρ_9 sont surjectives).

Serre a aussi conjecturé que seulement un nombre fini de nombres premiers, ne dépendant pas de la courbe elliptique, peut apparaître dans le second point du théorème 1.3.9. La conjecture actuelle est que pour toutes courbes elliptiques sans multiplication complexe et tous nombres premiers $\pi > 37$, l'application ρ_π est surjective (et donc ρ_{π^k} aussi, pour tout entier $k \geqslant 1$, d'après le paragraphe précédent). Dans [149], Zywina décrit un algorithme qui permet de calculer les nombres premiers π pour lesquels ρ_π n'est pas surjective. De plus, il montre que la conjecture est vérifiée pour toutes les courbes elliptiques sans multiplication complexe de la base de données de Cremona [46] utilisée dans `MAGMA` [99] (ce qui correspond à toutes les courbes elliptiques dont le conducteur [1] est au plus 350 000).

Remarque 1.3.11. Une application des résultats de Serre est donnée ici. Les expériences montrent que si $E/\mathbb{Q}$ est une courbe elliptique sans multiplication complexe, alors $E(\mathbb{F}_p)$ est proche d'un groupe cyclique pour presque tous les nombres premiers p, indépendamment du rang de E sur $\mathbb{Q}$. Pour une borne B donnée, calculer

$$\mathrm{Prob}(\exists \pi > B \mid \mathbb{Z}/\pi\mathbb{Z} \times \mathbb{Z}/\pi\mathbb{Z} \subset E(\mathbb{F}_p)) \tag{1.5}$$

dépasse le propos de ce chapitre. Cependant, si π est un nombre premier tel que ρ_π est surjective, en particulier pour $\pi > 37$ si la conjecture de Serre est vraie, alors le corollaire 1.3.6 montre que

$$\mathrm{Prob}(\mathbb{Z}/\pi \times \mathbb{Z}/\pi \subset E(\mathbb{F}_p)) = \frac{1}{\pi(\pi+1)(\pi-1)^2}.$$

Cela suggère que la probabilité de l'équation (1.5) devrait être en $\mathrm{O}(1/B^3)$.

1.3.3 Divisibilité par une puissance de nombre premier

C'est un fait bien connu qu'étant donné un nombre premier π, le cardinal d'une courbe elliptique sur $\mathbb{F}_p$ possède une probabilité plus grande d'être divisible par π qu'un entier aléatoire de la même taille que p [97, prop. 1.14]. Dans cette section, nous allons nous intéresser à un problème analogue, où au lieu de fixer p et de faire varier E, nous allons fixer une courbe elliptique $E/\mathbb{Q}$ et faire varier p.

Pour simplifier les expressions des calculs de cette section, nous allons définir deux notations. Si π est un nombre premier et i, j et k sont des entiers positifs tels que $i \leqslant j$ et k est non nul, alors nous noterons

$$p_{\pi,k}(i, j) = \mathrm{Prob}(E(\mathbb{F}_p)[\pi^k] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}).$$

Si, de plus, ℓ et h sont des entiers positifs tels que $\ell \leqslant h$, alors nous noterons aussi, lorsque cela est défini,

$$p_{\pi,k}(\ell, h \mid i, j) = \mathrm{Prob}(E(\mathbb{F}_p)[\pi^{k+1}] \simeq \mathbb{Z}/\pi^\ell\mathbb{Z} \times \mathbb{Z}/\pi^h\mathbb{Z} \mid E(\mathbb{F}_p)[\pi^k] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}).$$

Si le contexte le permet, l'indice π peut être omis dans les notations précédentes.

1. Le conducteur d'une courbe est un entier divisible par les mêmes nombres premiers que le discriminant de la courbe.

Remarque 1.3.12. Comme pour tout entier $m \geqslant 2$ et tout nombre premier p ne divisant pas m, nous avons $E(\mathbb{F}_p)[m] \subset \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$, alors $p_{\pi,k}(i,j) = 0$ si $j > k$. Dans le cas où $j < k$, si $p_{\pi,k}(\ell, h \mid i,j)$ est défini, alors il est égal à 1 si $(\ell,h) = (i,j)$ et à 0 si $(\ell,h) \neq (i,j)$. Finalement, pour $j = k$, il n'y a que trois probabilités conditionnelles qui peuvent être non nulles : $p_{\pi,k}(i,k \mid i,k)$, $p_{\pi,k}(i,k+1 \mid i,k)$, et $p_{\pi,k}(k+1,k+1 \mid k,k)$.

Le théorème suivant donne la valeur de ces trois probabilités conditionnelles.

Théorème 1.3.13. *Soient $E/\mathbb{Q}$ une courbe elliptique et π un nombre premier. Si $k \geqslant 1$ est un entier tel que $I(E,\pi,k+1) = I(E,\pi,k)$ (par exemple, si E est sans multiplication complexe et $k \geqslant n(E,\pi)$), alors*

$$p_{\pi,k}(k+1,k+1 \mid k,k) = \frac{1}{\pi^4};$$
$$p_{\pi,k}(k,k+1 \mid k,k) = \frac{(\pi-1)(\pi+1)^2}{\pi^4};$$
$$p_{\pi,k}(i,k+1 \mid i,k) = \frac{1}{\pi} \quad \textit{pour } 0 \leqslant i < k.$$

Démonstration. Posons $M = (\mathbb{Z}/\pi^k\mathbb{Z})^2$. Pour tout $g \in \mathrm{GL}_2(\pi M)$, considérons l'ensemble

$$\mathrm{Lift}(g) = \left\{ h \in \mathrm{GL}_2(M) \mid h_{|\pi M} = g \right\} = \left\{ g + \pi^{k-1} \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \;\middle|\; a,b,c,d \in \mathbb{Z}/\pi\mathbb{Z} \right\}$$

dont la taille est π^4. Comme $I(E,\pi,k+1) = I(E,\pi,k)$, nous avons

$$\frac{\#\operatorname{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q})}{\#\operatorname{Gal}(\mathbb{Q}(E[\pi^{k+1}])/\mathbb{Q})} = \frac{\#\,\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})}{\#\,\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z})},$$

qui est égal à $\frac{1}{\pi^4}$ (remarque 1.3.3). Donc pour tout $g \in \rho_{\pi^k}(\operatorname{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))$, nous avons $\mathrm{Lift}(g) \subset \rho_{\pi^{k+1}}(\operatorname{Gal}(\mathbb{Q}(E[\pi^{k+1}])/\mathbb{Q}))$. Grâce au théorème 1.3.4, il ne nous reste qu'à compter pour chaque g le nombre d'éléments de $\mathrm{Lift}(g)$ qui fixent un sous-groupe donné.

Pour $g = \mathrm{Id} \in \rho_{\pi^k}(\operatorname{Gal}(\mathbb{Q}(E[\pi^k])/\mathbb{Q}))$, il n'y a qu'un seul élément de $\mathrm{Lift}(g)$ qui fixent $\left(\mathbb{Z}/\pi^{k+1}\mathbb{Z}\right)^2$, donc $p_{\pi,k}(k+1,k+1 \mid k,k) = \frac{1}{\pi^4}$.

Il y a exactement $\pi^4 - 1 - \#\,\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$ éléments de $\mathrm{Lift}(\mathrm{Id})$ qui fixent la π^k-torsion, un point d'ordre π^{k+1} mais pas toute la π^{k+1}-torsion. Donc $p_{\pi,k}(k,k+1 \mid k,k) = \frac{(\pi-1)(\pi+1)^2}{\pi^4}$.

Chaque élément de $\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$, différent de l'identité, qui fixe une ligne, peut être relevé d'exactement π^3 façons différentes en un élément de $\mathrm{GL}_2(\mathbb{Z}/\pi^{k+1}\mathbb{Z})$ qui fixe une ligne de $(\mathbb{Z}/\pi^{k+1}\mathbb{Z})^2$. Donc $p_{\pi,k}(i,k+1 \mid i,k) = \frac{\pi^3}{\pi^4} = \frac{1}{\pi}$. □

Le théorème suivant utilise les informations de $\operatorname{Gal}(\mathbb{Q}(E[\pi^{n(E,\pi)}])/\mathbb{Q})$ pour un nombre premier π, pour calculer les probabilités de divisibilité par une puissance de π. Il permet aussi de calculer la valuation moyenne de $\#E(\mathbb{F}_p)$ en π. La valuation moyenne, noté $\bar{v}_\pi$, est définie par

$$\bar{v}_\pi = \sum_{k \geqslant 1} k \operatorname{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k),$$

où v_π est la valuation en π. Nous ne prétendons pas que $\bar{v}_\pi$ soit égale à

$$\lim_{x \to \infty} \frac{1}{\mathcal{P}(x)} \sum_{p \leqslant x} v_\pi(\#E(\mathbb{F}_p))$$

même si nous nous attendons à ce que ce soit vrai.

Définissons quelques notations avant d'énoncer le théorème. Si π est un nombre premier, nous noterons

$$\gamma_n(h) = \pi^n \sum_{\ell=0}^{h} \pi^\ell p_n(\ell, n),$$

$$\delta(k) = \begin{cases} p_{i+1}(i+1, i+1) & \text{si } k = 2i+1 \\ 0 & \text{sinon} \end{cases} \quad \text{et}$$

$$S_k(h) = \pi^k \left(\sum_{\ell=h}^{\lfloor \frac{k}{2} \rfloor} p_{k-\ell}(\ell, k-\ell) + \delta(k) \right).$$

Théorème 1.3.14. *Soient π un nombre premier, $E/\mathbb{Q}$ une courbe elliptique et n un entier positif tel que $\forall k \geqslant n$, $I(E, \pi, k) = I(E, \pi, n)$ (par exemple, une courbe sans multiplication complexe et $n \geqslant n(E, \pi)$). Alors, pour tout entier $k \geqslant 1$,*

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \frac{1}{\pi^k} \begin{cases} S_k(0) & \text{pour } 1 \leqslant k \leqslant n, \\ \gamma_n(k-n-1) + S_k(k-n) & \text{pour } n < k < 2n, \\ \gamma_n(n) + p_n(n,n)\pi^{2n-1} - \frac{\pi^{4n-1} p_n(n,n)}{\pi^k} & \text{pour } k \geqslant 2n. \end{cases}$$

De plus, la valuation moyenne $\bar{v}_\pi$ est finie et

$$\bar{v}_\pi = 2 \sum_{\ell=1}^{n-1} p_\ell(\ell, \ell) + \frac{\pi}{\pi - 1} \sum_{\ell=0}^{n-1} p_n(\ell, n) + \sum_{\ell=0}^{n-2} \sum_{i=\ell+1}^{n-1} p_i(\ell, i) + \frac{\pi(2\pi+1)}{(\pi-1)(\pi+1)} p_n(n, n).$$

Démonstration. Soit $k \geqslant 1$ un entier. La figure 1.3, dans laquelle les probabilités conditionnelles sont notées $c_1 = \frac{1}{\pi^4}$, $c_2 = \frac{(\pi-1)(\pi+1)^2}{\pi^4}$, et $c_3 = \frac{1}{\pi}$, permet de voir que

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \sum_{\ell=0}^{\lfloor \frac{k}{2} \rfloor} p_{k-\ell}(\ell, k-\ell) + \delta(k). \tag{1.6}$$

En outre, pour $j > n$ et $\ell < n$, la probabilité $p_j(\ell, j)$ est le produit des probabilités conditionnelles de l'unique chemin de (ℓ, j) à (ℓ, n) dans le graphe de la figure 1.3, multiplié par la probabilité $p_n(\ell, n)$. De plus, pour $j > n$ et $\ell \geqslant n$, la probabilité $p_j(\ell, j)$ est le produit des probabilités conditionnelles de l'unique chemin de (ℓ, j) à (n, n) dans le graphe de la figure 1.3, multiplié par la probabilité $p_n(n, n)$.

Trois cas doivent être traités séparément : $1 \leqslant k \leqslant n$, $n < k < 2n$ et $k \geqslant 2n$. Pour $1 \leqslant k \leqslant n$, le résultat découle directement de l'équation (1.6). Détaillons le cas $k \geqslant 2n$, avec $k = 2i$ pour un entier i :

$$\begin{aligned} \mathrm{Prob}(\pi^{2i} \mid \#E(\mathbb{F}_p)) &= \sum_{\ell=0}^{i} p_{2i-\ell}(\ell, 2i-\ell) + \delta(2i) = \sum_{\ell=0}^{i} p_{2i-\ell}(\ell, 2i-\ell) \\ &= \sum_{\ell=0}^{n-1} p_{2i-\ell}(\ell, 2i-\ell) + \sum_{\ell=n}^{i-1} p_{2i-\ell}(\ell, 2i-\ell) + p_i(i, i) \\ &= \sum_{\ell=0}^{n-1} c_3^{2i-\ell-n} p_n(\ell, n) + \sum_{\ell=n}^{i-1} c_3^{2i-2\ell-1} c_2 c_1^{\ell-n} p_n(n, n) + c_1^{i-n} p_n(n, n). \end{aligned}$$

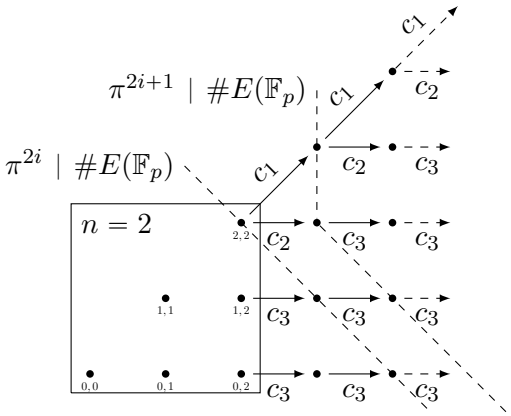

FIGURE 1.3 – Chaque nœud de coordonnées (i, j) dans le graphe représente l'événement $\big(E(\mathbb{F}_p)[\pi^j] \simeq \mathbb{Z}/\pi^i\mathbb{Z} \times \mathbb{Z}/\pi^j\mathbb{Z}\big)$. Les flèches représentent les probabilités conditionnelles du théorème 1.3.13.

La formule voulue s'obtient en remplaçant c_1, c_2 et c_3 par leur formule respective. Les cas $k \geqslant 2n$ impair et $n < k < 2n$ se traitent de la même façon.

Pour prouver la formule de la valuation moyenne, remarquons tout d'abord que la probabilité $\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$ est $\mathrm{O}(\frac{1}{\pi^k})$ lorsque $k \to \infty$. En effet, pour k assez grand ($k \geqslant 2n$), la formule que nous venons de prouver pour $\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$ est bornée par $\frac{C_n}{\pi^k}$, où C_n est une constante dépendant seulement de n. La somme définissant $\bar{v}_\pi$ est donc absolument convergente et nous pouvons donc réarranger les termes,

$$\bar{v}_\pi = \sum_{k \geqslant 1} k \,\mathrm{Prob}(v_\pi(\#E(\mathbb{F}_p)) = k) = \sum_{k \geqslant 1} \mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)).$$

En substituant dans cette dernière expression les formules que nous venons de prouver pour les probabilités $\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p))$, nous retrouvons la formule donnée dans l'énoncé du théorème pour la valuation moyenne $\bar{v}_\pi$. □

Exemple 1.3.15. Pour cet exemple, nous allons considérer une courbe elliptique E sans multiplication complexe et un nombre premier π tels que $n(E, \pi) = 1$ et $I(E, \pi, 1) = 1$ (c'est-à-dire, tels que l'application ρ_{π^k} est surjective pour tout entier $k \geqslant 1$). En particulier, si la conjecture de Serre est vraie, ces conditions sont vérifiées pour toute courbe elliptique sans multiplication complexe et tout nombre premier $\pi > 37$. En utilisant le théorème 1.3.14 avec $n = 1$, nous obtenons les formules suivantes :

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \begin{cases} p_1(0,1) + p_1(1,1) & \text{pour } k = 1, \\ \frac{\pi}{\pi^k}\left(p_1(0,1) + (\pi + 1 - \pi^{2-k})p_1(1,1)\right) & \text{si } k \geqslant 2, \end{cases}$$

$$\bar{v}_\pi = \frac{\pi}{\pi - 1}\left(p_1(0,1) + \frac{2\pi + 1}{\pi + 1} p_1(1,1)\right).$$

L'exemple 1.3.7 nous donne les probabilités $p_1(0,1)$ et $p_1(1,1)$ dans ce cas, ce qui nous permet

d'obtenir

$$\mathrm{Prob}(\pi^k \mid \#E(\mathbb{F}_p)) = \begin{cases} \frac{\pi^2-2}{(\pi-1)^2(\pi+1)} & \text{pour } k=1, \\ \frac{\pi^{k+1}-\pi^{k-1}-1}{(\pi+1)(\pi-1)^2\pi^{2(k-1)}} & \text{si } k \geqslant 2, \end{cases}$$
$$\bar{v}_\pi = \frac{(\pi^3+\pi^2-2\pi-1)\pi}{(\pi+1)^2(\pi-1)^3}.$$

Exemple 1.3.16. Dans cet exemple, nous allons comparer les valeurs théoriques et expérimentales de la valuation moyenne pour des nombres premiers $\pi \in \{2,3,5\}$ pour les courbes $E_1/\mathbb{Q}\colon y^2 = x^3+5x+7$ et $E_3/\mathbb{Q}\colon y^2 = x^3-10875x+526250$, qui sont sans multiplication complexe (nous n'avons pas considéré la courbe E_2 de l'exemple 1.3.8 dans cet exemple car elle est avec multiplication complexe). Pour E_1, nous avons utilisé l'exemple 1.3.15 car les groupes de Galois sont isomorphes à $\mathrm{GL}_2(\mathbb{Z}/\pi^k\mathbb{Z})$. Pour E_3, nous avons appliqué le théorème 1.3.14 avec $n=3$ pour $\pi=2$ et $n=1$ pour $\pi=3$ et $\pi=5$, et nous avons calculé les probabilités nécessaires grâce au théorème 1.3.4 (lorsque $n=3$) et au corollaire 1.3.6 (lorsque $n=1$). Les résultats sont donnés dans la table 1.2.

π	$n(E_1,\pi)$	$\bar{v}_{\pi,\mathrm{th}}$ $\bar{v}_{\pi,\mathrm{exp}}$	$n(E_3,\pi)$	$\bar{v}_{\pi,\mathrm{th}}$ $\bar{v}_{\pi,\mathrm{exp}}$
2	1	$\frac{14}{9} \approx 1.556$ 1.555	3	$\frac{895}{576} \approx 1.554$ 1.554
3	1	$\frac{87}{128} \approx 0.680$ 0.679	1	$\frac{39}{32} \approx 1.219$ 1.218
5	1	$\frac{695}{2304} \approx 0.302$ 0.301	1	$\frac{155}{192} \approx 0.807$ 0.807

TABLE 1.2 – Valeurs théoriques et expérimentales de la valuation moyenne en π pour $\#E_1(\mathbb{F}_p)$ et $\#E_3(\mathbb{F}_p)$, pour $\pi \in \{2,3,5\}$. Les valeurs théoriques proviennent du théorème 1.3.14 et les valeurs expérimentales ont été calculées en utilisant tous les nombres premiers inférieurs à 2^{25}. Les valeurs de $n(E_3,\pi)$ et celles de $\bar{v}_\pi$ pour E_3 sont hypothétiques (cf. dernier paragraphe de l'exemple 1.3.16.

Pour pouvoir appliquer le théorème 1.3.14, nous devons d'abord vérifier que les égalités $I(E,\pi,k) = I(E,\pi,n)$ pour tout entier $k \geqslant n$ (ou $n \geqslant n(E,\pi)$ puisque E_1 et E_3 sont sans multiplication complexe). Pour E_1, nous avons pu prouver que $n(E,\pi)=1$ pour $\pi=2$, $\pi=3$ et $\pi=5$ en utilisant les observations faites à la fin de la section 1.3.2. Pour E_3, Andrew Sutherland a calculé les groupes de Galois jusqu'à la 2^5-, 3^3- et 5^2-torsion. Ces calculs nous poussent à croire que $n(E_3,2)=3$, $n(E_3,3)=1$ et $n(E_3,5)=1$ mais nous n'avons pas été capables de prouver que ces valeurs sont correctes. Cela signifie, en particulier, que les probabilités théoriques pour E_3 données dans la table 1.2 sont hypothétiques.

1.4 Applications à certaines familles de courbes elliptiques

Comme nous l'avons montré dans la section précédente, changer les propriétés de torsion d'une courbe elliptique est équivalent à modifier son groupe de Galois. Le fait d'imposer de la torsion rationnelle est une façon de modifier le groupe de Galois. Dans cette section, nous allons modifier le groupe de Galois soit en décomposant les polynômes de division soit en imposant certaines

équations qui vont directement modifier le groupe de Galois. Avec ces idées, nous trouverons de nouvelles familles de courbes adaptées à l'algorithme ECM et nous expliquerons les propriétés de bonnes courbes elliptiques déjà connues.

Dans la suite, quand nous parlerons du *groupe de Galois de la m-torsion pour une famille de courbes*, nous parlerons d'un groupe isomorphe au groupe de Galois de la m-torsion pour toutes les courbes de la famille sauf pour un ensemble creux de courbes (qui peut avoir un groupe de Galois plus petit). Par exemple, considérons le groupe de Galois de la 2-torsion pour la famille suivante de courbes elliptiques : $\left\{ \mathcal{E}_r \colon y^2 = x^3 + rx^2 + x \;\middle|\; r \in \mathbb{Q}\backslash\{\pm 2\} \right\}$. Le groupe de Galois de la 2-torsion de la courbe $\mathcal{E} \colon y^2 = x^3 + Ax^2 + x$ sur $\mathbb{Q}(A)$ est $\mathbb{Z}/2\mathbb{Z}$. Donc, pour presque toutes les valeurs de r, la groupe de Galois sera $\mathbb{Z}/2\mathbb{Z}$ et pour un ensemble creux de valeurs de r le groupe de Galois sera le groupe trivial. Nous dirons donc que le groupe de Galois pour la 2-torsion pour cette famille est $\mathbb{Z}/2\mathbb{Z}$.

À notre connaissance, il n'existe pas d'implémentation d'un algorithme calculant les groupes de Galois de polynômes dont les coefficients sont dans un corps de fonctions. Mais nous pouvons calculer le groupe de Galois pour chaque courbe d'une famille, donc nous pouvons deviner le groupe de Galois pour la famille en regardant un nombre fini de courbes aléatoirement choisies dans la famille. En pratique, nous choisissons une douzaine de courbes aléatoirement dans la famille et si tous les groupes de Galois sont isomorphes, nous devinons que c'est le groupe de Galois pour la famille de courbes.

1.4.1 Meilleures courbes d'Edwards avec torsion $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ en utilisant les polynômes de division

Dans cette section, nous allons chercher des familles de courbes telles que certains facteurs des polynômes de division se décomposent en facteurs de degré plus petit. En faisant cela, nous essayons de changer le groupe de Galois pour améliorer les propriétés de torsion. Nous allons nous intéresser aux sous-familles de la famille des courbes d'Edwards avec $a = -1$ et dont le groupe de torsion sur $\mathbb{Q}$ est $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$. Ce sont exactement les courbes d'Edwards avec $a = -1$ et $d = -e^4$ [18]. Cette méthode peut aussi être utilisée pour trouver d'autres familles.

Recherche de sous-familles.

Pour une valeur de d générique, le polynôme P_8^{new} se décompose en trois facteurs irréductibles : deux de degré 4 et un de degré 16. Si d est de la forme $-e^4$, alors le facteur de degré 16 se décompose à son tour en trois facteurs irréductibles : deux de degré 4, que nous appellerons $P_{8,0}$ et $P_{8,1}$ dans la suite, et un de degré 8, que nous appellerons $P_{8,2}$. En obligeant l'un de ces trois polynômes à se décomposer encore, nous avons trouvé quatre familles, présentées dans la table 1.3.

Dans toutes ces sous-familles, la valeur générique de la valuation moyenne en 2 est augmentée de 1/6, passant de 14/3 à 29/6, sauf pour la famille $e = (g - g^{-1})/2$ pour laquelle elle est augmentée de 2/3. Cette dernière famille a la même valuation moyenne en 2 que les courbes d'Edwards avec $a = 1$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$. Enfin, notons que ces quatre sous-familles couvrent toutes les courbes présentées dans les trois premières colonnes de [18, table 3.1], sauf les deux courbes avec $e = 26/7$ et $e = 19/8$ qui ont un groupe de Galois générique pour la 8-torsion.

La famille $a = -1$ et $e = (g - g^{-1})/2$.

Dans cette section, nous allons donner plus de détails sur la famille de courbes d'Edwards avec $a = -1$ et $e = (g - g^{-1})/2$. En utilisant le théorème 1.3.4, il peut être prouvé que l'ordre

	Degrés des facteurs de			valuation moyenne en 2 pour		
forme spéciale de e	$P_{8,0}$	$P_{8,1}$	$P_{8,2}$	$p \equiv 1 \bmod 4$	$p \equiv 3 \bmod 4$	tous les p
aucune	4	4	8	$\frac{16}{3}$	4	$\frac{14}{3}$
g^2	4	4	4 et 4	$\frac{17}{3}$	4	$\frac{29}{6}$
$\frac{2g^2+2g+1}{2g+1}$	4	4	4 et 4	$\frac{17}{3}$	4	$\frac{29}{6}$
$\frac{g^2}{2}$	2 et 2	4	8	$\frac{17}{3}$	4	$\frac{29}{6}$
$\frac{g-g^{-1}}{2}$	2 et 2	2 et 2	8	$\frac{17}{3}$	5	$\frac{16}{3}$

TABLE 1.3 – Valuation moyenne en 2 pour différents sous-ensembles de nombres premiers de $\#E(\mathbb{F}_p)$, pour E dans l'une des sous-familles de courbes d'Edwards avec $a = -1$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$. Toutes les sous-familles ont $d = -e^4$, où e vérifie à son tour une certaine équation décrite dans la première colonne. Les degrés des facteurs des polynômes $P_{8,i}$ sont donnés de la seconde à la quatrième colonne. Les valuations moyennes en 2 de $\#E(\mathbb{F}_p)$ pour les nombres premiers p congrus à 1 modulo 4, à 3 modulo 4 et tous les nombres premiers sont données dans les cinquième, sixième et septième colonnes, respectivement.

du groupe de la courbe modulo tous les nombres premiers est divisible par 16. Cependant nous donnons une autre preuve qui a aussi son intérêt. Pour cela, nous avons besoin de calculer les points de 8-torsion dont le double est un des points de 4-torsion $(\pm\sqrt[4]{-d^{-1}}, \pm\sqrt[4]{-d^{-1}})$.

Théorème 1.4.1. *Soit $E/\mathbb{Q}$ une courbe d'Edwards telle que $a = -1$, $d = -e^4$, $e = (g - g^{-1})/2$ et $g \in \mathbb{Q}\setminus\{-1, 0, 1\}$. Soit $p > 3$ un nombre premier de bonne réduction. Si $t \in \{1, -1\}$ est tel que $tg(g-1)(g+1)$ est un résidu quadratique modulo p alors les points $(x, y) \in E(\mathbb{F}_p)$ pour lesquels il existe un $w \in \{1, -1\}$ tel que*

$$y = \pm\sqrt{\frac{4tg^{2-w}}{(g - tw)^3(g + tw)}} \quad et \quad x = \pm g^w y \tag{1.7}$$

sont des points d'ordre 8 dont le double est $(\pm e^{-1}, te^{-1})$.

Démonstration. Pour qu'un point (x, y) soit d'ordre 8, ni x ni y ne doivent être égaux à 0 ou à ∞. D'après [19, théorème 2.10], le double d'un point (x, y) est

$$\left((2xy : 1 + dx^2y^2), (x^2 + y^2 : 1 - dx^2y^2)\right) = \left((2xy : -x^2 + y^2), (x^2 + y^2 : 2 - (-x^2 + y^2))\right). \tag{1.8}$$

Soient s et t dans $\{-1, 1\}$ tels que le double de (x, y) soit (se^{-1}, te^{-1}). alors

$$\frac{2xy}{-x^2 + y^2} = \frac{s}{e} \quad \text{et} \quad \frac{x^2 + y^2}{2 - (-x^2 + y^2)} = \frac{t}{e}.$$

De la première égalité nous obtenons $(x/y)^2 + 2exs/y + e^2 = 1 + e^2$. En écrivant $e = (g - g^{-1})/2$ cela devient $(x/y + se)^2 = ((g + g^{-1})/2)^2$. Nous avons donc que $x/y \in \{\pm g, \pm 1/g\}$, suivant le signe de s et le signe une fois la racine carrée prise. Cela donne $x^2 = G^2y^2$ avec $G \in \{g^2, g^{-2}\}$.

De la deuxième équation, nous obtenons $(e-t)x^2+(e+t)y^2 = 2t$ et, en substituant $x^2 = G^2y^2$, $((e - t)G^2 + (e + t))y^2 = 2t$. Cela peut se résoudre en y lorsque $2t((e - t)G^2 + (e + t))$ est un résidu quadratique modulo p. C'est équivalent à vérifier que l'un des nombres suivants est un

résidu quadratique modulo p :

$$2t\left((e-1)g^2+(e+1)\right) = \frac{t(g-1)^3(g+1)}{g}, \tag{1.9}$$

$$2t\left((e-1)+(e+1)g^2\right) = \frac{t(g-1)(g+1)^3}{g}. \tag{1.10}$$

Par hypothèse, $tg(g-1)(g+1)$ est un résidu quadratique modulo p, donc (1.9) et (1.10) sont des résidus quadratiques modulo p. Résoudre en y en gardant trace de tous les signes permet de montrer la formule (1.7). □

Une conséquence directe de ce théorème est donnée dans le corollaire suivant.

Corollaire 1.4.2. *Soient $E/\mathbb{Q}$ une courbe d'Edwards avec $a=-1$, $d=-((g-g^{-1})/2)^4$ pour $g\in\mathbb{Q}\setminus\{-1,0,1\}$ et $p>3$ un nombre premier de bonne réduction. Alors $E(\mathbb{Q})$ a $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/4\mathbb{Z}$ pour groupe de torsion sur $\mathbb{Q}$ et l'ordre du groupe de $E(\mathbb{F}_p)$ est divisible par 16.*

Démonstration. Il faut distinguer deux cas suivant les congruences de p modulo 4.

Si $p\equiv 1 \pmod 4$ alors -1 est un résidu quadratique modulo p. Donc les points de 4-torsion $(\pm i,0)$ existent (cf. figure 1.2) et $16 \mid \#E(\mathbb{F}_p)$.

Si $p\equiv 3 \pmod 4$ alors -1 est un non-résidu quadratique modulo p. Donc exactement l'un des $\{\, g(g-1)(g+1), -g(g-1)(g+1)\,\}$ est un résidu quadratique modulo p et le théorème 1.4.1 permet de montrer que la courbe $E(\mathbb{F}_p)$ a un point de 8-torsion et que donc $16 \mid \#E(\mathbb{F}_p)$. □

Le corollaire 1.4.2 permet d'expliquer le bon comportement de la courbe d'Edwards avec $a=-1$ et $d=-(77/36)^4$ trouvée dans [18]. En effet, nous avons $d=-((g-g^{-1})/2)^4$ pour $g=9/2$ et, donc, l'ordre du groupe de cette courbe est divisible par un facteur 2 supplémentaire.

Corollaire 1.4.3. *Soient $E/\mathbb{Q}$ une courbe d'Edwards avec $a=-1$, $d=-((g-g^{-1})/2)^4$ pour un $g\in\mathbb{Q}\setminus\{-1,0,1\}$ et un nombre premier $p\equiv 1 \pmod 4$ qui soit un premier de bonne réduction. Si $g(g-1)(g+1)$ est un résidu quadratique modulo p alors l'ordre du groupe de $E(\mathbb{F}_p)$ est divisible par 32.*

Démonstration. Les 16 points de 4-torsion sont $\mathbb{F}_p$-rationnels (cf. figure 1.2). Et le théorème 1.4.1 montre qu'il existe au moins un point de 8-torsion. Alors $32 \mid \#E_d(\mathbb{F}_p)$. □

Nous avons généré différentes valeurs de $g\in\mathbb{Q}$ de la forme $g=i/j$ pour $1\leqslant i<j\leqslant 200$ tels que i et j sont premiers entre eux. Nous avons obtenu 12 231 valeurs possibles pour g et Sage [130] a trouvé 614 points d'ordre infini sur $\mathbb{Q}$. Comme attendu, nous avons observé que ces courbes se comportent comme la bonne courbe trouvée dans [18].

Dans [18] une paramétrisation elliptique pour d et les coordonnées d'un point d'ordre infini sur $\mathbb{Q}$ est donnée. Les calculs sur la courbe génératrice peuvent être utilisés pour générer une famille infinie de courbes d'Edwards dont le groupe de torsion sur $\mathbb{Q}$ est $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/4\mathbb{Z}$ et dont les coordonnées d'un point d'ordre infini sur $\mathbb{Q}$ est connu. En utilisant des idées provenant de [30], nous avons trouvé une paramétrisation rationnelle qui permet d'éviter les calculs sur cette courbe.

Théorème 1.4.4. *Soit $t\in\mathbb{Q}\setminus\{0,\pm 1,\pm 3,\pm 1/3\}$. Posons*

$$a=-1,\qquad e=\frac{3(t^2-1)}{8t},\qquad d=-e^4,\qquad x_\infty=\frac{1}{4e^3+3e}\qquad \text{et}\qquad y_\infty=\frac{9t^4-2t^2+9}{9t^4-9}.$$

Alors la courbe d'Edwards $E/\mathbb{Q}$: $ax^2+y^2=1+dx^2y^2$ a $\mathbb{Z}/2\mathbb{Z}\times\mathbb{Z}/4\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$ et (x_∞,y_∞) est un point de $E(\mathbb{Q})$ d'ordre infini sur $\mathbb{Q}$.

Démonstration. Comme $t \neq 0$ et $t \neq \pm 1$, e, d, x_∞ et y_∞ sont des rationnels non nuls. En outre, $e \neq \pm 1$, car $t \neq \pm 3$ et $t \neq \pm 1/3$, donc $d \neq -1$. Donc E est bien une courbe elliptique, son groupe de torsion sur $\mathbb{Q}$ est $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$, car $d = -e^4$. Un rapide calcul montre que le point (x_∞, y_∞) est bien sur la courbe et est d'ordre infini car $x_\infty \notin \left\{ 0, \infty, \pm e^{-1} \right\}$. □

Cette paramétrisation rationnelle nous permet d'imposer une condition supplémentaire sur le paramètre e et d'espérer obtenir une paramétrisation elliptique. Pour les quatre sous-familles présentées précédemment, le paramètre e est donné par une courbe elliptique de rang 0, sauf pour la famille $e = g^2$ dont le cas est traité ci-dessous.

Corollaire 1.4.5. *Soit (x, y) un point d'ordre infini sur la courbe elliptique $E/\mathbb{Q}$: $y^2 = x^3 - 36x$ dont le rang est 1. Posons $t = (x + 6)/(x - 6)$. La courbe d'Edwards définie avec a et d donnés par les formules du théorème 1.4.4 appartient à la famille $e = g^2$ et a un rang non nul.*

1.4.2 Meilleures courbes de Suyama par changement direct du groupe de Galois

Dans cette section, nous allons améliorer les probabilités de divisibilité de certaines courbes elliptiques en changeant leurs groupes de Galois mais sans changer leurs polynômes de division comme dans la section précédente.

Le théorème 1.1.21 suggère qu'en imposant des équations aux paramètres a et d d'une courbe d'Edwards (ou A et B pour une courbe de Montgomery), les propriétés de torsion peuvent être améliorées. Le cas où $\frac{a}{d}$ est un carré a été étudié dans [19] pour la famille de courbes d'Edwards avec $a = 1$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$, et dans [18] pour la famille avec $a = -1$ et $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$. Nous allons nous concentrer sur deux autres équations :

$$\exists c \in \mathbb{Q},\ a = -c^2 \qquad (A + 2 = -Bc^2 \text{ pour les courbes de Montgomery}), \tag{1.11}$$

$$\exists c \in \mathbb{Q},\ a - d = c^2 \qquad (B = c^2 \text{ pour les courbes de Montgomery}). \tag{1.12}$$

Le cardinal du groupe de Galois pour la 4-torsion pour une courbe de Montgomery ou d'Edwards générique est 16 et est réduit à 8 pour la famille de courbes vérifiant l'égalité (1.11). En utilisant le théorème 1.3.4, nous pouvons calculer le changement de probabilités dû à ce nouveau groupe de Galois. Pour la famille de courbes vérifiant l'égalité (1.11), pour tous les nombres premiers p tels que $p \equiv 1 \pmod 4$, la probabilité d'avoir $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$ comme groupe de 2-torsion modulo p passe de 1/4 à 0 et les probabilités d'avoir $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ et $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ comme groupe de 4-torsion passent de 1/8 à 1/4.

Le cardinal du groupe de Galois pour la 8-torsion pour la famille de courbes vérifiant l'égalité (1.12) est 128, au lieu de 256 pour les courbes de Montgomery ou d'Edwards génériques. Le théorème 1.3.4 permet de voir que les probabilités d'avoir un point de 8-torsion sont améliorées.

En utilisant le théorème 1.3.14, nous pouvons montrer que, pour les deux familles de courbes, celles vérifiant l'égalité (1.11) et celles vérifiant l'égalité (1.12), la probabilité que le cardinal soit divisible par 8 passe de 5/8 à 3/4 et la valuation moyenne en 2 passe de 10/3 à 11/3.

Famille Suyama-11

Kruppa a observé dans sa thèse [86] que parmi les courbes de Suyama, celles correspondant à $\sigma = 11$ trouvent exceptionnellement plus de facteurs. Barbulescu [8] a étendu cela en une famille infinie de courbes que nous présentons en détail ici.

Des expériences ont montré que la courbe de Suyama avec $\sigma = 11$ diffère des autres courbes de Suyama seulement par les probabilités d'avoir un groupe de 2^k-torsion donné lorsqu'elle est réduite modulo un nombre premier $p \equiv 1 \pmod 4$. La raison est que la courbe de Suyama avec $\sigma = 11$ vérifie l'équation (1.11). La discussion ci-dessus détaille le changement des probabilités induit par l'équation (1.11) et montre que la valuation moyenne en 2 passe de 10/3 à 11/3.

Appelons Suyama-11 la famille de courbes de Suyama vérifiant l'équation (1.11). En résolvant le système d'équations (1.4) auquel est ajouté l'équation (1.11), nous obtenons une paramétrisation elliptique pour σ. Étant donné un point (u, v) sur la courbe

$$E_{\sigma_{11}} : v^2 = u^3 - u^2 - 120u + 432,$$

la valeur de σ associée est $\sigma = 5 + 120/(u - 24)$. Le groupe $E_{\sigma_{11}}(\mathbb{Q})$ est généré par le point $P_\infty = (-6, 30)$ d'ordre infini et les points $P_2 = (-12, 0)$ et $Q_2 = (4, 0)$ d'ordre 2. Les points $\mathcal{O}$, P_∞, $\boxminus P_\infty$, P_2, Q_2, $P_2 \boxplus Q_2$, $Q_2 \boxplus P_\infty$ et $Q_2 \boxminus P_\infty$ doivent être exclus car ils produisent des valeurs de σ invalides. Si R est un point de la courbe, les points $\boxminus R$, $Q_2 \boxplus R$ et $Q_2 \boxminus R$ produisent des courbes isomorphes. La courbe de Suyama avec $\sigma = 11$ est produite par le point $(44, 280) = P_\infty \boxplus P_2$.

Edwards $\mathbb{Z}/6\mathbb{Z}$ et Suyama-11

Il a été montré dans [18, section 5] que les courbes d'Edwards avec $a = -1$ et $\mathbb{Z}/6\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$ sont exactement les courbes telles que $a = -1$ et

$$d = -\frac{16u^3(u^2 - u + 1)}{(u - 1)^6(u + 1)^2}, \tag{1.13}$$

où u est un paramètre rationnel[2]. En particulier, d'après [18, section 5.3], toute courbe de Suyama peut être traduite en une courbe d'Edwards où la condition que $-a$ soit un carré est imposée pour obtenir une courbe où $a = -1$. En outre, [18, section 5.5] montre que cette famille de courbes a des propriétés de torsion exceptionnelles.

Pour comprendre les propriétés de cette famille, il faut remarquer que la condition que $-a$ soit un carré correspond à l'équation (1.11). Et donc la famille de courbes d'Edwards avec $a = -1$ et $\mathbb{Z}/6\mathbb{Z}$ comme groupe de torsion sur $\mathbb{Q}$ est la traduction des courbes de la famille Suyama-11 dans le langage des courbes d'Edwards, ses bonnes propriétés de torsion ont donc été expliquées précédemment. Ces deux familles ont été découvertes indépendamment dans [8] et [18].

Suyama-$\frac{9}{4}$.

Grâce à des expériences menées par Zimmermann, d'autres courbes de Suyama avec des propriétés de torsion exceptionnelles ont été découvertes, comme la courbe avec $\sigma = 9/4$. Des expériences supplémentaires ont montré que la courbe de Suyama avec $\sigma = 9/4$ diffère des autres courbes de Suyama seulement par les probabilités d'avoir un groupe de 2^k-torsion donné lorsqu'elle est réduite modulo un nombre premier $p \equiv 3 \pmod 4$. La raison est que la courbe de Suyama avec $\sigma = 9/4$ vérifie l'équation (1.12). La discussion ci-dessus détaille le changement des probabilités induit par l'équation (1.12) et montre que la valuation moyenne en 2 passe de 10/3 à 11/3.

2. Dans la preuve de [18, théorème 5.1], il manque un signe moins dans la fraction correspondant à l'équation (1.13).

Appelons Suyama-$\frac{9}{4}$ la famille de courbes de Suyama vérifiant l'équation (1.12). En résolvant le système d'équations (1.4) auquel est ajouté l'équation (1.12), nous obtenons une paramétrisation elliptique pour σ. Étant donné un point (u, v) sur la courbe

$$E_{\sigma_{9/4}} : v^2 = u^3 - 5u,$$

la valeur de σ associée est $\sigma = u$. Le groupe $E_{\sigma_{9/4}}(\mathbb{Q})$ est généré par le point $P_\infty = (-1, 2)$ d'ordre infini et le point $P_2 = (0, 0)$ d'ordre 2. Les points $\mathcal{O}$, P_∞, $\boxminus P_\infty$, P_2, $P_2 \boxplus P_\infty$ et $P_2 \boxminus P_\infty$ doivent être exclus car ils produisent des valeurs de σ invalides. Si la différence entre deux points de $E_{\sigma_{9/4}}(\mathbb{Q})$ est P_2 alors les deux courbes correspondant à ces deux points sont isomorphes. La courbe de Suyama avec $\sigma = 9/4$ est produite par le point $(9/4, -3/8) = 2P_\infty$.

1.4.3 Comparaisons

La table 1.4 donne un résumé et une comparaison de toutes les familles discutées dans ce chapitre. Les valeurs théoriques des valuations moyennes ont été calculées grâce aux théorèmes 1.3.14 et 1.3.4 et au corollaire 1.3.6, en faisant certaines suppositions sur l'exposant de Serre (voir l'exemple 1.3.16 pour plus d'informations).

Famille	Courbe	n_2	$\bar{v}_{2,\text{th}}$ $\bar{v}_{2,\text{exp}}$	n_3	$\bar{v}_{3,\text{th}}$ $\bar{v}_{3,\text{exp}}$
Suyama	$\sigma = 12$	2	$\frac{10}{3} \approx 3.333$ 3.331	1	$\frac{27}{16} \approx 1.688$ 1.689
Suyama-11	$\sigma = 11$	2	$\frac{11}{3} \approx 3.667$ 3.669	1	$\frac{27}{16} \approx 1.688$ 1.687
Suyama-$\frac{9}{4}$	$\sigma = \frac{9}{4}$	3	$\frac{11}{3} \approx 3.667$ 3.664	1	$\frac{27}{16} \approx 1.688$ 1.687
$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ (Edwards $a = -1$, $d = -e^4$)	$e = 11$	3	$\frac{14}{3} \approx 4.667$ 4.666	1*	$\frac{87}{128} \approx 0.680$ 0.679
$e = (g - g^{-1})/2$	$g = \frac{9}{2}$	3	$\frac{16}{3} \approx 5.333$ 5.332	1*	$\frac{87}{128} \approx 0.680$ 0.679
$e = g^2$	$g = 3$	3	$\frac{29}{6} \approx 4.833$ 4.833	1*	$\frac{87}{128} \approx 0.680$ 0.680
$e = g^2/2$	$g = \frac{9}{2}$	3	$\frac{29}{6} \approx 4.833$ 4.831	1*	$\frac{87}{128} \approx 0.680$ 0.679
$e = (2g^2 + 2g + 1)/(2g + 1)$	$g = 1$	3	$\frac{29}{6} \approx 4.833$ 4.833	1*	$\frac{87}{128} \approx 0.680$ 0.679

TABLE 1.4 – Valeurs théoriques et expérimentales de $\bar{v}_2$ et $\bar{v}_3$ pour différentes courbes des familles discutées dans ce chapitre. Les valeurs théoriques ont été calculées grâce au théorème 1.3.14 et les valeurs expérimentales ont été calculées en utilisant tous les nombres premiers inférieurs à 2^{25}. Les colonnes n_2 et n_3 donnent les valeurs de $n(E, 2)$ et $n(E, 3)$. La notation 1* signifie que le groupe de Galois est isomorphe à $\mathrm{GL}_2(\mathbb{Z}/\pi\mathbb{Z})$.

Nous pouvons remarquer que lorsqu'un point de torsion sur $\mathbb{Q}$ est imposé, la valuation moyenne n'augmente pas simplement de 1, comme le montre la valuation en 3 des courbes de Suyama.

1.5 Conclusion et perspectives

Dans ce chapitre, nous avons utilisé la théorie des groupes de Galois pour analyser les propriétés de torsion des courbes elliptiques. En particulier, nous avons exprimé, étant donnée une courbe elliptique $E/\mathbb{Q}$, la probabilité qu'une puissance de nombre premier π^k divise le cardinal du groupe de points de $E(\mathbb{F}_p)$ pour un nombre premier p aléatoire. Nous avons déterminé le comportement des courbes elliptiques génériques et expliqué les propriétés exceptionnelles de certaines courbes connues (courbes d'Edwards dont le groupe de torsion sur $\mathbb{Q}$ est $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}$ et $\mathbb{Z}/6\mathbb{Z}$). Des nouvelles techniques, suggérées par l'étude théorique, nous ont permis de trouver des familles infinies de courbes ayant de bonnes propriétés de torsion.

Les travaux effectués dans ce chapitre nous ont amené à nous poser les questions suivantes :

— Est-il possible d'utiliser les travaux de Serre pour avoir des résultats sur les probabilités pour la m- et m'-torsion, lorsque m et m' sont des entiers premiers entre eux ?

— Existe-t-il un modèle qui permettrait de calculer plus précisément la probabilité de succès de ECM à partir des probabilités données dans le théorème 1.3.14 ?

— Est-il possible d'utiliser de manière effective la méthode des résolvantes [38] dans le but de calculer des équations qui amélioreraient les propriétés de torsion ?

Chapitre 2

L'algorithme NFS pour la factorisation d'entiers

L'algorithme NFS (pour Number Field Sieve*) est l'algorithme le plus rapide pour factoriser de grands entiers (à partir de 90 chiffres décimaux environ) pour lesquels il n'existe pas de petits facteurs pouvant être trouvés par des algorithmes adaptés, comme P-1, P+1 ou ECM. Dans ce chapitre, nous décrirons l'algorithme NFS d'un point de vue global dans la section 2.1 et nous donnerons plus de détails sur les différentes étapes de l'algorithme dans les sections suivantes.*

Dans tout ce chapitre, nous noterons N l'entier à factoriser et $\mathcal{P}$ l'ensemble des nombres premiers.

2.1 Description générale de l'algorithme NFS

L'algorithme NFS se base sur le principe de la congruence de carrés présenté dans l'introduction et cherche donc à construire des paires d'entiers x et y tels que

$$x^2 \equiv y^2 \pmod{N}, \tag{2.1}$$

en espérant que $x \not\equiv \pm y \pmod{N}$, dans le but de trouver un facteur de N en calculant le pgcd de $x-y$ et N et le pgcd de $x+y$ et N. Si N est un nombre entier ayant au moins deux facteurs premiers impairs, alors, en considérant que les entiers x et y sont répartis aléatoirement de façon uniforme, la probabilité que le pgcd de $x-y$ et N ou que le pgcd de $x+y$ et N retourne un facteur de N différent de 1 et N est supérieure à $1/2$.

Pour construire ces congruences de carrés, l'algorithme NFS produit des carrés dans des corps de nombres qui permettent d'obtenir des égalités entre des carrés modulo N. La première version de l'algorithme NFS, ne fonctionnant que pour certains entiers, est décrite par Pollard dans [124]. Ensuite, plusieurs articles ont été écrits pour généraliser la méthode, en particulier [95] et [32]. Un livre [92] contenant une partie des articles à la base de l'algorithme NFS a été publié, il contient aussi une bibliographie annotée [98] donnant plus de détails sur tous les travaux qui ont permis la construction de l'algorithme NFS.

2.1.1 Les différentes étapes de l'algorithme NFS

L'algorithme NFS commence par le choix de deux polynômes f_1 et f_2 à coefficients entiers, irréductibles sur $\mathbb{Q}$, premiers entre eux sur $\mathbb{Q}$ et ayant une racine commune m modulo N. Nous

parlerons de *côté* 1 (resp. *côté* 2) pour parler de ce qui concerne le polynôme f_1 (resp. le polynôme f_2) et nous noterons d_i le degré du polynôme f_i, pour $i \in \{1, 2\}$. Dans la suite de ce chapitre, pour simplifier la présentation, nous considérerons que les polynômes f_1 et f_2 sont unitaires. Les changements induits par l'utilisation de polynômes non unitaires sont décrits dans [32, section 12]. En pratique, dans la majorité des cas, l'algorithme NFS est utilisé avec un côté dont le polynôme est de degré 1. Dans ce cas, tout ce qui se rapporte à ce côté est souvent qualifié de *rationnel* et tout ce qui se rapporte à l'autre côté est souvent qualifié d'*algébrique*. La partie de l'algorithme consistant à trouver les deux polynômes f_1 et f_2 est appelée l'étape de *sélection polynomiale* pour laquelle plus de détails sont donnés dans la section 2.2.

Fixons $i \in \{1, 2\}$ dans ce paragraphe pour définir quelques notations. Tout d'abord, nous noterons F_i le polynôme homogène $F_i(X, Y) = f_i(X/Y)Y^{d_i}$ associé au polynôme f_i. De plus, nous noterons $K_i = \mathbb{Q}(\alpha_i)$, où α_i est une racine de f_i, le corps de nombres défini par le polynôme f_i et $\mathcal{O}_{K_i}$ l'anneau des entiers de K_i. L'anneau $\mathbb{Z}[\alpha_i]$, défini par $\mathbb{Z}[\alpha_i] = \mathbb{Z}[X]/f_i\mathbb{Z}[X]$, est un ordre de $\mathbb{Q}(\alpha_i)$ inclus dans l'ordre maximal $\mathcal{O}_{K_i}$ mais, en général, $\mathbb{Z}[\alpha_i]$ est différent de $\mathcal{O}_{K_i}$. La norme d'un élément γ de $\mathbb{Z}[\alpha_i]$ sera notée $\mathcal{N}_i(\gamma)$. Nous utiliserons dans la suite le fait que, si a et b sont deux nombres entiers, alors $\mathcal{N}_i(a - b\alpha_i) = \mathrm{Res}(a - bX, f_i) = F_i(a, b)$. La norme d'un idéal non nul $\mathfrak{I}$ de $\mathbb{Z}[\alpha_i]$ est égale au cardinal de l'anneau quotient $\mathbb{Z}[\alpha_i]/\mathfrak{I}$ et dans le cas où l'idéal $\mathfrak{I}$ est principal, c'est-à-dire qu'il existe $\gamma \in \mathbb{Z}[\alpha_i]$ tel que $\mathfrak{I} = \gamma\mathbb{Z}[\alpha_i]$, la norme de $\mathfrak{I}$ est égale à la valeur absolue de $\mathcal{N}_i(\gamma)$. Les idéaux premiers non nuls de $\mathbb{Z}[\alpha_i]$ sont les idéaux $\mathfrak{p}$ tels que $\mathbb{Z}[\alpha_i]/\mathfrak{p}$ est un corps, cela correspond aux idéaux dont la norme est une puissance d'un nombre premier. Si $\mathfrak{p}$ est un idéal premier non nul de $\mathbb{Z}[\alpha_i]$ de norme p^f, où $p \in \mathcal{P}$ et f est un entier strictement positif, alors nous appellerons f le degré de l'idéal $\mathfrak{p}$ et nous dirons que l'idéal $\mathfrak{p}$ est au-dessus de p. Nous noterons aussi, pour tout nombre premier $p \in \mathcal{P}$, $R_i(p)$ l'ensemble des racines de f_i modulo p, c'est-à-dire

$$R_i(p) = \{\, r \in \mathbb{Z}/p\mathbb{Z} \mid f_i(r) \equiv 0 \pmod{p} \,\}. \tag{2.2}$$

Il existe une bijection entre l'ensemble des idéaux premiers non nuls de degré 1 de $\mathbb{Z}[\alpha_i]$ et l'ensemble $\{\, (p, r) \mid p \in \mathcal{P}, r \in R_i(p) \,\}$. La paire (p, r), avec p un nombre premier et $r \in R_i(p)$, est associée à l'idéal $\mathfrak{p}$ premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$ engendré par p et $\alpha_i - r$, c'est-à-dire $\mathfrak{p} = p\mathbb{Z}[\alpha_i] + (\alpha_i - r)\mathbb{Z}[\alpha_i]$. Finalement, nous noterons μ_i le morphisme d'anneau de $\mathbb{Z}[X]$ dans $\mathbb{Z}[\alpha_i]$ qui à X associe α_i et ϕ_i le morphisme d'anneau de $\mathbb{Z}[\alpha_i]$ dans $\mathbb{Z}/N\mathbb{Z}$ qui à α_i associe m modulo N.

L'algorithme NFS repose sur le fait que le diagramme de la figure 2.1 est un diagramme commutatif. En effet pour tout élément Π de $\mathbb{Z}[X]$, $\phi_1(\mu_1(\Pi))$ et $\phi_2(\mu_2(\Pi))$ sont égaux dans $\mathbb{Z}/N\mathbb{Z}$ car f_1 et f_2 ont une racine commune modulo N.

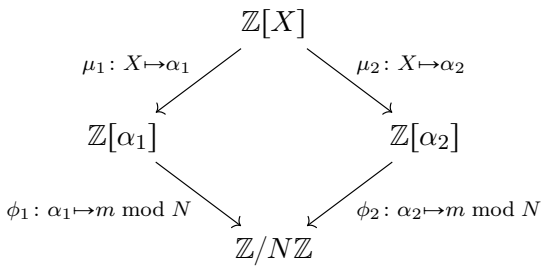

FIGURE 2.1 – Diagramme commutatif à la base de l'algorithme NFS.

Remarque 2.1.1. Dans le cas où le côté i est rationnel, c'est-à-dire $d_i = 1$, α_i est égale à m, la racine commune de f_1 et f_2, le corps de nombres K_i est en fait $\mathbb{Q}$, les anneaux $Z[\alpha_i]$ et $\mathcal{O}_{K_i}$

sont égaux et correspondent à $\mathbb{Z}$ et les idéaux premiers non nuls sont les nombres premiers (et sont donc de degré 1). Si a et b sont deux nombres entiers, alors la norme $\mathcal{N}_i(a - b\alpha_i)$ est égale à $a - bm$.

Fixons encore $i \in \{ 1, 2 \}$ dans ce paragraphe pour donner quelques propriétés sur les idéaux premiers non nuls de degré 1. Une propriété importante [32, corollaire 5.5] est que si a et b sont deux entiers premiers entre eux et que $\mathfrak{p}$ est un idéal premier non nul de $\mathbb{Z}[\alpha_i]$ contenant $a - b\alpha_i$ alors $\mathfrak{p}$ est de degré 1. De plus, un idéal premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$, engendré par l'unique paire (p, r), où $p \in \mathcal{P}$ et $r \in R_i(p)$, contient $a - b\alpha_i$ si et seulement si $a \equiv br \pmod{p}$. Une propriété encore plus forte est que s'il existe un nombre premier p et un entier positif k non nul tels que le facteur p apparaît exactement k fois dans la factorisation de $\mathcal{N}_i(a - b\alpha_i)$, alors il existe un unique $r \in R_i(p)$ tel que $a \equiv br \pmod{p}$ et l'idéal premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$ correspondant à (p, r) divise l'idéal principal $(a - b\alpha_i)\mathbb{Z}[\alpha_i]$ exactement k fois. Ceci permet de définir, pour tout nombre premier p et tout entier $r \in R_i(p)$, la fonction $e_{i,p,r}$ par

$$e_{i,p,r}(a - b\alpha_i) = \begin{cases} v_p(\mathcal{N}_i(a - b\alpha_i)) & \text{si } a \equiv br \pmod{p} \\ 0 & \text{sinon} \end{cases}, \tag{2.3}$$

où a et b sont deux entiers premiers entre eux et v_p est la valuation en p. Nous ferons parfois l'abus de langage suivant : si $\mathfrak{p}$ est l'idéal correspondant à (p, r), alors nous noterons $e_{i,\mathfrak{p}}$ au lieu de $e_{i,p,r}$. Dans le cas où le côté i est rationnel, c'est-à-dire $d_i = 1$, alors $e_{i,p,r}(a - b\alpha_i)$ correspond simplement à la valuation en p de $\mathcal{N}_i(a - b\alpha_i) = a - bm$.

Pour construire une égalité de carrés modulo N, l'algorithme NFS cherche un élément Π de $\mathbb{Z}[X]$ tel que $\mu_1(\Pi)$ et $\mu_2(\Pi)$ sont des carrés dans $\mathbb{Z}[\alpha_1]$ et $\mathbb{Z}[\alpha_2]$. En effet, en notant, pour $i \in \{ 1, 2 \}$, $\mu_i(\Pi) = \gamma_i^2$, où $\gamma_i \in \mathbb{Z}[\alpha_i]$, et en remarquant que comme ϕ_i est un morphisme d'anneau alors $\phi_i(\mu_i(\Pi))$ est le carré de $\phi_i(\gamma_i)$ dans $\mathbb{Z}/N\mathbb{Z}$, nous obtenons, en utilisant le fait que $\phi_1(\mu_1(\Pi))$ et $\phi_2(\mu_2(\Pi))$ sont égaux dans $\mathbb{Z}/N\mathbb{Z}$, que $\phi_1(\gamma_1)^2 \equiv \phi_2(\gamma_2)^2 \pmod{N}$. L'algorithme NFS essaie de construire cet élément $\Pi \in \mathbb{Z}[X]$ comme un produit d'éléments de la forme $a - bX$, où a et b sont deux nombres entiers premiers entre eux. L'image de $a - bX$ par μ_i est $a - b\alpha_i$. Pour construire l'élément Π, il faut donc chercher un ensemble $\mathcal{S}$ de paires (a, b) d'entiers premiers entre eux tel que la propriété suivante est simultanément vérifiée pour $i \in \{ 1, 2 \}$:

$$\prod_{(a,b)\in\mathcal{S}} (a - b\alpha_i) \text{ est un carré dans } \mathbb{Z}[\alpha_i]. \tag{2.4}$$

D'après [32, proposition 5.3], une condition nécessaire pour que cette propriété soit vérifiée est que l'égalité suivante soit vraie pour $i \in \{ 1, 2 \}$ et pour tout idéal $\mathfrak{p}$ premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$:

$$\sum_{(a,b)\in\mathcal{S}} e_{i,\mathfrak{p}}(a - b\alpha_i) \equiv 0 \pmod{2}. \tag{2.5}$$

La condition (2.5) n'est pas suffisante en général mais nous verrons dans la section 2.6.1 comment rendre cette condition suffisante avec une probabilité proche de 1. Nous allons donc considérer, dans la suite, que cette condition est nécessaire et suffisante et nous allons montrer comment construire des ensembles $\mathcal{S}$ qui satisfont cette condition. Pour cela, définissons, pour $i \in \{ 1, 2 \}$, la borne de friabilité B_i comme un entier positif non nul et la *base de facteurs* $\mathcal{B}_i$ comme l'ensemble des idéaux premiers non nuls de degré 1 de $\mathbb{Z}[\alpha_i]$ dont la norme est inférieure ou égale à B_i. Une *relation* est une paire (a, b) d'entiers premiers entre eux telle que, simultanément pour $i \in \{ 1, 2 \}$, $\mathcal{N}_i(a - b\alpha_i)$ est B_i-friable, ou de façon équivalente, pour $i \in \{ 1, 2 \}$, si $\mathfrak{p}$ est un idéal premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$ tel que $e_{i,\mathfrak{p}}(a - b\alpha_i) \neq 0$, alors $\mathfrak{p} \in \mathcal{B}_i$. Le sous-ensemble de $\mathbb{Z}^2$ dans lequel

les relations sont cherchées est appelé la *zone de crible* et est noté Ω. L'ensemble des relations calculées sera noté $\mathcal{R}$. Il n'est pas nécessaire que cet ensemble contienne toutes les relations de la zone de crible Ω mais, comme nous le verrons par la suite, seulement que $\#\mathcal{R} > \#\mathcal{B}_1 + \#\mathcal{B}_2$. Plus de détails sur l'étape de *collecte des relations* sont donnés dans la section 2.3.

Une fois l'ensemble $\mathcal{R}$ des relations calculé, il reste à trouver plusieurs sous-ensembles $\mathcal{S}$ qui satisfont la condition (2.5). Cela se ramène à un problème d'algèbre linéaire qui est traité lors de *l'étape de filtrage*, pour laquelle plus de détails sont donnés dans la section 2.4, et de *l'étape d'algèbre linéaire*, pour laquelle plus de détails sont donnés dans la section 2.5.

La dernière étape de l'algorithme NFS est le *calcul des racines carrées*. Pour chaque ensemble $\mathcal{S}$ et pour $i \in \{1, 2\}$, il faut calculer $\gamma_i \in \mathbb{Z}[\alpha_i]$ tel que $\prod_{(a,b)\in\mathcal{S}} (a - b\alpha_i) = \gamma_i^2$. Plus de détails sur le calcul des γ_i et des explications sur comment gérer les problèmes qui empêchent la condition (2.5) d'être suffisante sont donnés dans la section 2.6. Une fois les racines carrées calculées, il reste à calculer $\text{pgcd}(\phi_1(\gamma_1) \pm \phi_2(\gamma_2), N)$ pour espérer trouver un facteur non trivial de N. Tous les ensembles $\mathcal{S}$ sont considérés tant que tous les facteurs premiers de N ne sont pas trouvés.

2.1.2 Complexité, implémentations et records

La complexité de l'algorithme NFS, pour un choix optimal des différents paramètres, est

$$\mathrm{L}_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right) = \exp\left(\left(\sqrt[3]{\frac{64}{9}} + \mathrm{o}(1)\right)(\log N)^{\frac{1}{3}} (\log\log N)^{\frac{2}{3}}\right).$$

Une analyse détaillée de la complexité peut être trouvée dans [32, section 11]. La complexité dépend en grande partie du choix de la somme $d_1 + d_2$ des degrés des polynômes f_1 et f_2. Pour les plus grands entiers actuellement factorisés à l'aide de l'algorithme NFS, la somme $d_1 + d_2$ vaut entre 5 et 7.

Il existe plusieurs implémentations de NFS. L'une des premières implémentation de NFS, par Bernstein et Lenstra, est décrite dans [21]. Il y a aussi eu une implémentation développée au CWI par Herman te Riele *et al.*, qui n'est plus maintenue mais dont le code source est disponible [47]. Parmi les implémentations actuelles de NFS, il existe `GGNFS` [59], développé par Monico, `Msieve` [111], développé par Papadopoulos, et `cado-nfs` [33], développé principalement à Inria à Nancy. Toutes les nouveautés décrites dans cette thèse ont été implémentées dans `cado-nfs`. Il existe aussi des logiciels implémentant seulement une étape de l'algorithme NFS, comme le crible de Franke et Kleinjung, dont le code source n'est pas publiquement disponible mais dont une ancienne version peut être trouvée dans `GGNFS`.

Au moment où cette thèse est écrite, le record de factorisation avec l'algorithme NFS est la factorisation de RSA-768 en 2010 [83]. L'entier RSA-768 est un entier de 768 bits (232 chiffres décimaux) qui faisait partie des challenges RSA [87]. Cette factorisation a duré environ deux ans sur un millier de machines.

2.2 Sélection polynomiale

L'objectif de la sélection polynomiale est de trouver les deux polynômes qui définissent les corps de nombres qui seront utilisés par NFS. Étant donnés l'entier N et deux entiers positifs d_1 et d_2 tels que $d_1 d_2 > 1$, l'algorithme de sélection polynomiale doit trouver deux polynômes f_1 et f_2 de degrés respectifs d_1 et d_2, à coefficients entiers, irréductibles sur $\mathbb{Q}$, premiers entre eux sur $\mathbb{Q}$ et ayant une racine commune m modulo N. Dans le cas où l'un des côtés est rationnel, c'est-à-dire que le degré de l'un des deux polynômes est 1, nous parlerons de sélection polynomiale

linéaire, en opposition à la sélection polynomiale non linéaire lorsque les deux polynômes sont de degré strictement supérieur à 1.

Dans le cas de la sélection polynomiale linéaire, l'algorithme de Kleinjung [79, 81] est utilisé pour produire des paires de polynômes pour lesquelles $d_1 = 1$, $d_2 > 1$ et la taille des coefficients du polynôme algébrique est en $\mathrm{O}(N^{1/(d_2+1)})$. Dans le cas de la sélection polynomiale non linéaire, un algorithme de génération optimal n'était connu, jusqu'à récemment, que pour $d_1 = d_2 = 2$. Il s'agit d'un algorithme de Montgomery, dont une description peut être trouvée dans [56] ou [113, section 2.3.1]. Le chapitre 3 de cette thèse est consacré à la sélection polynomiale pour l'algorithme NFS.

2.2.1 Variante SNFS

Il existe des entiers N pour lesquels il est plus facile de trouver une bonne paire de polynômes. En particulier, pour les entiers de la forme $uk^n + v$, où u, k et n sont des entiers strictement positifs et où v est un entier non nul, il existe des paires de polynômes très faciles à trouver. Lorsque u et v sont suffisamment petits, ces polynômes ont des coefficients beaucoup plus petits que ceux produits par les méthodes classiques de sélection polynomiale.

Par exemple, pour la factorisation du neuvième nombre de Fermat $F_9 = 2^{2^9} + 1 = 2^{512} + 1$, la paire de polynômes suivante a été utilisée [94] : $f_1 = X - 2^{103}$ et $f_2 = X^5 + 8$, ces polynômes ayant $m = 2^{103}$ comme racine commune modulo N. De façon générale, pour $N = uk^n + v$ comme précédemment, il est facile de trouver deux paires de polynômes de degré 1 et d, pour tout entier $2 \leqslant d \leqslant n$. En effet, si $m = k^{\lfloor n/d \rfloor}$ et $\epsilon = n - d\lfloor n/d \rfloor$, alors les polynômes $f_1 = X - m$ et $f_2 = uk^\epsilon X^d + v$ ont m comme racine commune modulo N. De même, si $m = k^{\lceil n/d \rceil}$ et $\epsilon = d\lceil n/d \rceil - n$, alors les polynômes $f_1 = X - m$ et $f_2 = uX^d + k^\epsilon v$ ont m comme racine commune modulo N.

La variante de l'algorithme NFS où la sélection polynomiale est un problème trivial du fait de la forme spéciale de N est appelée SNFS (pour *Special Number Field Sieve*) et a une complexité en $\mathrm{L}_N\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$.

2.3 Collecte des relations : l'étape de crible

L'objectif de l'étape de collecte des relations, aussi appelée *étape de crible*, est de calculer l'ensemble des relations $\mathcal{R}$, c'est-à-dire un maximum de paires $(a, b) \in \Omega$, avec a et b premiers entre eux, telles que $\mathcal{N}_i(a - b\alpha_i) = F_i(a, b)$ est B_i-friable, pour $i \in \{1, 2\}$. De plus, pour chaque relation (a, b) dans $\mathcal{R}$, il est nécessaire de connaître la factorisation de $\mathcal{N}_1(a - b\alpha_1)$ et $\mathcal{N}_2(a - b\alpha_2)$ pour pouvoir calculer $e_{i,\mathfrak{p}}(a - b\alpha_i)$ pour $i \in \{1, 2\}$ et $\mathfrak{p} \in \mathcal{B}_i$. Remarquons tout d'abord que si la paire (a, b) est une relation alors la paire $(-a, -b)$ l'est aussi mais n'apporte aucune information supplémentaire. Ces deux paires sont donc considérées comme identiques et, en pratique, la zone de crible Ω est souvent un sous-ensemble de $\mathbb{Z} \times \mathbb{N}$.

Des algorithmes de crible sont utilisés pour identifier les paires (a, b) qui sont des relations. Ils ressemblent au crible d'Ératosthène, à la différence que ce ne sont pas des nombres premiers qui sont recherchés mais des nombres friables. L'idée à la base de ces algorithmes de crible est que si $p \mid F_i(a, b)$ alors $p \mid F_i(a + k_a p, b + k_b p)$ pour tous les entiers k_a et k_b. Étant donnés (a, b) une paire d'entiers premiers entre eux et p un nombre premier tels que p divise la norme d'un des deux côtés, ces algorithmes de cribles vont marquer toutes les paires $(a + k_a p, b + k_b p)$, pour des valeurs entières de k_a et k_b qui dépendent de la zone de crible. Une fois que suffisamment de nombres premiers ont été criblés de cette façon, les paires (a, b), avec a et b premiers, qui ont été beaucoup

marquées sont gardées, car ce sont celles qui ont le plus de chances d'être simultanément friables. Ensuite, pour ces valeurs de a et b, les entiers $F_1(a,b)$ et $F_2(a,b)$ sont complètement factorisés à l'aide d'algorithmes comme P-1, P+1, ECM ou MPQS, suivant la taille des facteurs recherchés. Finalement sont gardées seulement les valeurs de a et b pour lesquelles, simultanément, $F_1(a,b)$ est B_1-friable et $F_2(a,b)$ est B_2-friable.

Le crible le plus simple est le crible par lignes. Dans le cas du crible par lignes, la région de crible est très souvent de la forme $\Omega = [-A, A] \times]0, B]$, où A et B sont deux entiers strictement positifs. Tout d'abord un entier $b_0 \in]0, B]$ est fixé et le crible s'effectue sur la ligne correspondant à $b = b_0$. Pour cette ligne, pour chaque $i \in \{ 1, 2 \}$ et pour chaque idéal $\mathfrak{p} \in \mathcal{B}_i$, représenté par un nombre premier p et un entier $r \in R_i(p)$, l'ensemble des valeurs de a qui doivent être marquées, c'est-à-dire pour lesquelles $p \mid F_i(a, b_0)$, est l'ensemble $\{ a \in [-A, A] \mid a \equiv rb_0 \pmod{p} \}$. Il suffit de calculer $a_0 \in [0, p[$ tel que $a_0 \equiv b_0 r \pmod{p}$ et de considérer tous les $a_0 + kp$, pour tous les entiers k dans l'intervalle $[-(A + a_0)/p, (A - a_0)/p[$. Le crible par lignes devient peu efficace lorsque p et A ont une taille similaire car il y a, pour chaque idéal $\mathfrak{p}$, très peu de valeurs de a à considérer, voire parfois aucune lorsque p est plus grand que A.

Une autre méthode de crible est le crible par réseaux, décrit par Pollard dans [125]. Si $\mathfrak{p} \in \mathcal{B}_i$ est un idéal, correspondant à la paire (p, r) avec p un nombre premier et $r \in R_i(p)$, qui doit être criblé, l'idée du crible par réseaux est de trouver un maximum de points du réseaux engendré par $(p, 0)$ et $(r, 1)$ qui appartiennent aussi à Ω. Pour ce faire, Pollard propose de calculer une base réduite $\{ (a_0, b_0), (a_1, b_1) \}$ de ce réseau, de fixer deux bornes entières K et J et de marquer toutes les positions $(ka_0 + ja_1, kb_0 + jb_1)$, pour $k \in [-K, K]$ et $j \in [-J, J]$ tels que $kb_0 + jb_1$ est positif. Certains points appartenant au réseau et à Ω peuvent être manqués en utilisant cette méthode. Une meilleure méthode pour trouver des points du réseau qui appartiennent aussi à Ω a été proposée dans [58]. Pour les grands idéaux, le crible par réseaux est souvent moins coûteux que le crible par lignes.

La technique dite du *spécial-*$\mathfrak{q}$ peut être utilisée pour améliorer les cribles par lignes et par réseaux. L'idée est que, pour $i \in \{ 1, 2 \}$ fixé, si $\mathfrak{q}$ est un idéal premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$ correspondant à la paire (q, ρ), avec q un nombre premier et $\rho \in R_i(q)$, alors les paires (a, b) pour lesquelles $a - b\alpha_i \in \mathfrak{q}$ (ce qui implique $q \mid F_i(a, b)$) sont exactement les points d'un réseau, noté $L(\mathfrak{q})$, engendré par $(q, 0)$ et $(\rho, 1)$. Lorsqu'un spécial-$\mathfrak{q}$ est utilisé, le crible, par lignes ou par réseaux, n'est plus fait dans Ω mais dans $L(\mathfrak{q}) \cap \Omega$. Cela a l'avantage de forcer un facteur q dans la norme d'un des deux côtés et donc d'augmenter la probabilité de friabilité. Par contre l'utilisation de spécial-$\mathfrak{q}$ entraîne que certaines relations seront trouvées plusieurs fois (pour des spécial-$\mathfrak{q}$ différents). Ces doublons devront être repérés et éliminés mais cela peut être fait une fois l'étape de crible terminée. Il existe aussi des variantes des cribles par lignes et par réseaux qui autorisent des *grands premiers*, c'est-à-dire qui autorisent les entiers $F_i(a, b)$ à avoir un ou plusieurs facteurs premiers supérieurs à la borne de friabilité B_i.

Il existe des relations, dites gratuites, qui peuvent être trouvées avec très peu de calcul. Elles correspondent aux paires (a, b) avec $a = p$, où p est un nombre premier inférieur au minimum de B_1 et B_2, et $b = 0$. Une paire $(p, 0)$ correspond à une relation si f_1 et f_2 se factorisent complètement en facteurs de degré 1 lorsqu'ils sont considérés comme des polynômes sur $\mathbb{Z}/p\mathbb{Z}$, ce qui arrive avec probabilité $1/(\#\operatorname{Gal}(K_1/\mathbb{Q})\#\operatorname{Gal}(K_2/\mathbb{Q}))$, d'après le théorème de Chebotarev. À part pour l'algorithme SNFS, les polynômes utilisés n'ont pas de propriétés particulières du point de vue du groupe de Galois, donc, $\#\operatorname{Gal}(K_i) = d_i!$ en général, ce qui signifie que à $d_1 + d_2$ constant, plus d_1 et d_2 sont proches, plus la probabilité d'avoir une relation gratuite est grande.

2.4 Filtrage

Le but de l'étape de filtrage est de construire une matrice dont le noyau, qui sera calculé lors de l'étape d'algèbre linéaire, contient l'information sur des ensembles $\mathcal{S}$ inclus dans l'ensemble des relations $\mathcal{R}$ qui satisfont l'équation (2.5) pour $i \in \{1, 2\}$. La matrice est définie sur $\mathbb{Z}/2\mathbb{Z}$ et est construite de la façon suivante : une ligne de la matrice correspond à une relation et la ligne correspondant à la relation $(a, b) \in \mathcal{R}$ est le vecteur $(e_{i,\mathfrak{p}}(a - b\alpha_i))_{(i,\mathfrak{p})\in\{1\}\times\mathcal{B}_1\cup\{2\}\times\mathcal{B}_2}$. En remarquant que multiplier deux relations revient à additionner les vecteurs correspondants de la matrice, le problème de trouver des ensembles $\mathcal{S}$ se transforme en un problème de calcul du noyau à gauche de la matrice. Tout vecteur non nul du noyau à gauche de la matrice correspond à un ensemble $\mathcal{S}$ différent pour lequel l'équation (2.5) est vérifiée pour $i \in \{1, 2\}$.

Pour s'assurer que le noyau à gauche de la matrice ne soit pas réduit au vecteur nul, il suffit d'avoir plus de lignes que de colonnes, c'est-à-dire plus de relations que d'éléments dans les deux bases de facteurs réunies. Ceci justifie la condition $\#\mathcal{R} > \#\mathcal{B}_1 + \#\mathcal{B}_2$ énoncée dans la section 2.1. Lorsque le crible par spécial-$\mathfrak{q}$ est utilisé, des relations peuvent apparaître plus d'une fois. Une table de hachage peut être utilisée pour repérer et supprimer ces doublons au début de l'étape de filtrage. L'étape de filtrage consiste ensuite à réduire la taille de la matrice construite à partir des relations, sans perdre trop d'information sur le noyau à gauche, afin d'accélérer le calcul de l'étape d'algèbre linéaire. Le chapitre 5 de cette thèse est consacré à l'étude de l'étape de filtrage pour les algorithmes de crible, comme NFS pour la factorisation et NFS-DL ou FFS pour le calcul de logarithme discret.

2.5 Algèbre linéaire

Le but de l'étape d'algèbre linéaire est le calcul du noyau à gauche de la matrice produite par l'étape de filtrage. Les matrices apparaissant dans le contexte de l'algorithme NFS sont des matrices très creuses. En effet, pour ces matrices de plusieurs dizaines de milliers à plusieurs centaines de millions de lignes et de colonnes, suivant la taille de l'entier N, le nombre moyen de coefficients non nuls par ligne est autour de 25 au début de l'étape de filtrage. Une fois passée par l'étape de filtrage, le nombre moyen de coefficients non nuls par ligne des matrices dont le noyau est calculé lors de l'étape d'algèbre linéaire est compris entre 100 et 200 environ. Deux algorithmes sont majoritairement utilisés pour calculer le noyau de matrices très creuses sur $\mathbb{Z}/2\mathbb{Z}$: *Wiedemann par bloc* et *Lanczos par bloc*. L'algorithme de Wiedemann par bloc est une amélioration de l'algorithme de Wiedemann [145] décrite par Coppersmith dans [41]. L'algorithme de Lanczos par bloc, décrit dans [40] et [107], est une adaptation de l'algorithme de Lanczos [89], qui sert initialement à calculer des valeurs propres de matrices. L'opération principale de ces deux algorithmes est le calcul de produits matrice–vecteur. La complexité de ces deux algorithmes est similaire et dépend, en première approximation, du produit de la taille de la matrice par le poids de la matrice, où le poids est le nombre de coefficients non nuls. Les avantages de l'algorithme de Lanczos par bloc sont que la mémoire nécessaire est moins importante (l'algorithme de Wiedemann par bloc à une étape centrale très gourmande en mémoire) et que le nombre d'opérations requises est moins grand. Cependant l'algorithme de Lanczos par bloc utilise à la fois la matrice dont le noyau est calculé et sa transposée dans les calculs de produits matrice–vecteur, ce qui n'est pas le cas de l'algorithme de Wiedemann par bloc où la transposée n'apparaît pas. De plus, il est plus facile de distribuer les calculs de l'algorithme de Wiedemann par bloc sur plusieurs machines, ce qui permet de traiter des matrices plus grandes. En fonction de la taille de l'entier N à factoriser, donc de la taille de la matrice lors de l'étape d'algèbre linéaire, les deux algorithmes

peuvent être utilisés.

2.6 Calcul des racines carrées

L'étape d'algèbre linéaire produit plusieurs ensembles $\mathcal{S}$ tels que l'équation (2.5) est vérifiée pour $i \in \{1, 2\}$. Comme nous l'avions remarqué dans la section 2.1, et ignoré jusqu'à présent, cela n'implique pas nécessairement que l'équation (2.4) soit vérifiée pour $i \in \{1, 2\}$. Nous allons maintenant montrer comment obtenir, avec une probabilité proche de 1, des ensembles $\mathcal{S}$ pour lesquels l'équation (2.4) est vérifiée pour $i \in \{1, 2\}$, et nous donnerons quelques détails sur le calcul de la racine carrée des produits $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)$.

2.6.1 Obstructions et caractères

Les deux côtés se traitant de la même façon, nous allons fixer pour cette discussion $i \in \{1, 2\}$. Il existe 4 obstructions qui peuvent empêcher un ensemble $\mathcal{S}$ qui vérifie l'équation (2.5) de vérifier l'équation (2.4) :

1. l'idéal $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)\mathcal{O}_{K_i}$ n'est pas nécessairement le carré d'un idéal dans $\mathcal{O}_{K_i}$, car nous avons considéré les idéaux premiers de $\mathbb{Z}[\alpha_i]$ et non ceux de $\mathcal{O}_{K_i}$;
2. même si l'idéal $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)\mathcal{O}_{K_i}$ est le carré d'un idéal $\mathfrak{I}$, l'idéal $\mathfrak{I}$ n'est pas nécessairement principal ;
3. même si l'idéal $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)\mathcal{O}_{K_i}$ est égal à l'idéal $\gamma_i^2\mathcal{O}_{K_i}$ pour un certain $\gamma_i \in \mathcal{O}_{K_i}$, alors le produit $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)$ n'est pas nécessairement égal à γ_i^2 ;
4. finalement, même si $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i) = \gamma_i^2$ pour un certain $\gamma_i \in \mathcal{O}_{K_i}$, alors γ_i n'appartient pas nécessairement à $\mathbb{Z}[\alpha_i]$.

Ces 4 obstructions sont dues au fait qu'en général $\mathbb{Z}[\alpha_i]$ n'est pas égal à $\mathcal{O}_{K_i}$, que l'anneau $\mathcal{O}_{K_i}$ n'est pas principal et que le groupe des unités de $\mathcal{O}_{K_i}$ n'est pas trivial.

La dernière obstruction peut facilement être levée en utilisant le fait que si γ_i appartient à $\mathcal{O}_{K_i}$, alors $f_i'(\alpha_i)\gamma_i$ appartient à $\mathbb{Z}[\alpha_i]$. Il suffit donc de multiplier le produit $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)$ par $f_i'(\alpha_i)^2$. Pour gérer les 3 autres obstructions, Adleman [1] a proposé d'utiliser des *caractères*. Un caractère est un morphisme de groupe d'un sous-groupe multiplicatif de $\mathbb{Z}[\alpha_i]$ dans $\{\pm 1\}$. Dans le cas de l'algorithme NFS, pour tout idéal $\mathfrak{q}$ premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$, un caractère $\chi_\mathfrak{q}$ de $\mathbb{Z}[\alpha_i]\backslash\mathfrak{q}$ dans $\{\pm 1\}$ peut être défini, pour $\gamma \in \mathbb{Z}[\alpha_i]\backslash\mathfrak{q}$, par

$$\chi_\mathfrak{q}(\gamma) = \left(\frac{\gamma \bmod \mathfrak{q}}{q}\right),$$

où q est le nombre premier au-dessous de $\mathfrak{q}$ et $\left(\frac{\cdot}{\cdot}\right)$ représente le symbole de Legendre. Si γ appartient à $\mathbb{Z}[\alpha_i]\backslash\mathfrak{q}$, alors $\gamma \bmod \mathfrak{q}$ est un élément non nul de $\mathbb{Z}[\alpha_i]/\mathfrak{q} \simeq \mathbb{F}_q$, et donc son symbole de Legendre avec q est non nul. Le résultat suivant donne une indication sur l'utilité des caractères pour trouver un carré.

Proposition 2.6.1 ([32, proposition 8.3]). *Si $\mathcal{S}$ est un ensemble tel que $\prod_{(a,b)\in\mathcal{S}}(a - b\alpha_i)$ est le carré d'un élément de K_i et si $\mathfrak{q}$ est un idéal premier non nul de degré 1 de $\mathbb{Z}[\alpha_i]$ tel que $f_i'(\alpha_i) \notin \mathfrak{q}$ et, pour toute paire $(a, b) \in \mathcal{S}$, $a - b\alpha_i \notin \mathfrak{q}$, alors*

$$\prod_{(a,b)\in\mathcal{S}} \chi_\mathfrak{q}(a - b\alpha_i) = 1. \tag{2.6}$$

La réciproque de la proposition ci-dessus n'est, en général, pas vraie mais il est possible de montrer que si l'équation (2.6) est vérifiée pour suffisamment de caractères choisis aléatoirement alors la probabilité que le produit $\prod_{(a,b)\in\mathcal{S}}(a-b\alpha_i)$ soit un carré dans $\mathbb{Z}[\alpha_i]$ est élevée. Il faut choisir des idéaux $\mathfrak{q}$ premiers non nuls de degré 1 de $\mathbb{Z}[\alpha_i]$ ne contenant pas $f_i'(\alpha_i)$ et n'appartenant pas à la base de facteurs $\mathcal{B}_i$, pour assurer que $a-b\alpha_i \notin \mathfrak{q}$ pour toute relation $(a,b)\in\mathcal{R}$. Une justification plus rigoureuse de l'utilisation des caractères ainsi qu'une estimation du nombre de caractères à utiliser pour l'algorithme NFS sont présentées dans [32, section 8].

Remarque 2.6.2. Dans le cas où le côté i est rationnel, c'est-à-dire $d_i = 1$, alors les caractères ne sont pas nécessaires. En effet, pour tout ensemble $\mathcal{S}$ produit par l'algèbre linéaire, la valeur absolue du produit $\prod_{(a,b)\in\mathcal{S}}(a-bm)$ est un carré dans $\mathbb{Z}$ par construction. Il est cependant nécessaire d'ajouter un élément correspondant à -1 dans la base de facteurs pour gérer le signe de ce produit et le forcer à être positif.

En résumé, les obstructions présentées en début de section sont levées en choisissant un certain nombre de caractères à la fin de l'étape d'algèbre linéaire et en ne gardant, parmi tous les ensembles $\mathcal{S}$ produits par l'étape d'algèbre linéaire, que ceux pour lesquels l'équation (2.6) est vérifiée pour tous les caractères des deux côtés.

2.6.2 Calcul de racine carrée

Il reste à calculer les racines carrées des produits $\prod_{(a,b)\in\mathcal{S}}(a-b\alpha_i)$, pour $i\in\{1,2\}$, pour tous les ensembles $\mathcal{S}$ restants après la vérification des caractères. Dans le cas rationnel, c'est-à-dire lorsque $d_i = 1$, le calcul de la racine carrée est facile. En effet, soit une racine carrée du produit $\prod_{(a,b)\in\mathcal{S}}(a-bm)$ sur $\mathbb{Z}$ est calculée avant de la réduire modulo N, soit la racine carrée est calculée directement modulo N en remarquant que la factorisation en facteurs premiers de chaque relation, et donc du produit et de sa racine carrée, est connue sur $\mathbb{Z}$. Dans le cas algébrique, c'est-à-dire lorsque $d_i > 1$, il existe plusieurs algorithmes pour calculer la racine carrée. Le produit $\prod_{(a,b)\in\mathcal{S}}(a-b\alpha_i)$ est représenté par le polynôme P_i à coefficients entiers défini par $(\prod_{(a,b)\in\mathcal{S}}(a-bX)) \bmod f_i$. Le calcul de la racine carrée du produit devient le calcul d'un polynôme C_i à coefficients entiers tel que $C_i^2 \equiv P_i \pmod{f_i}$. La première méthode, présentée dans [32, section 9], consiste à calculer d'abord une solution en considérant tous les polynômes modulo un nombre premier p, puis de relever la solution modulo $p^2, p^4, \ldots$ Cette méthode ne fonctionne que si le nombre premier p est inerte, c'est-à-dire si $f_i \bmod p$ est irréductible. Or pour certains polynômes, en particulier pour ceux provenant de SNFS, il se peut que de tel nombre premier n'existe pas. Une méthode différente, utilisant le théorème des restes chinois, est décrite par Couveignes dans [42]. Elle ne fonctionne que pour les polynômes de degré impair. Dans [142], Thomé décrit une autre méthode basée sur le théorème des restes chinois qui fonctionne pour tous les polynômes. Une dernière méthode qui peut être utilisée pour calculer les racines carrées est décrite par Montgomery dans [106, 108] et par Nguyen dans [118].

Finalement, une fois les deux racines carrées calculées, il reste à calculer le pgcd de leur somme avec N, ainsi que le pgcd de leur différence avec N. Tous les ensembles $\mathcal{S}$ sont considérés jusqu'à ce que tous les facteurs de N aient été trouvés.

Chapitre 3

La sélection polynomiale pour NFS

Dans ce chapitre, nous étudierons le problème de la sélection polynomiale pour l'algorithme NFS, dont une description a été donnée dans le chapitre précédent. Dans la section 3.1, nous présenterons les différents aspects de la sélection polynomiale. Ensuite, dans la section 3.2, nous étudierons les algorithmes de Kleinjung pour la génération de paires de polynômes dans le cas de la sélection polynomiale linéaire. Dans la section 3.3, nous présenterons une amélioration de l'algorithme d'optimisation de la taille des polynômes dans le cas de la sélection polynomiale linéaire. Enfin, dans la section 3.4, nous présenterons la sélection polynomiale non linéaire et nous montrerons comment générer des paires de polynômes de degré 3. Les résultats de la section 3.3 sont tirés de l'article [5], écrit en collaboration avec S. Bai, A. Kruppa et P. Zimmermann.

Dans tout ce chapitre, nous noterons N l'entier positif pour lequel le problème de la sélection polynomiale est considéré.

3.1 Généralités sur la sélection polynomiale

L'étape de sélection polynomiale est la première étape de l'algorithme NFS, dont le but est de trouver les deux polynômes qui serviront à définir les corps de nombres dans lesquels le reste des calculs seront effectués. Dans cette section, nous présenterons les différents aspects de la sélection polynomiale.

3.1.1 Description du problème de sélection polynomiale

Comme nous l'avons vu dans le chapitre 2, le but de la sélection polynomiale pour NFS est, étant donnés l'entier N et deux nombres entiers strictement positifs d_1 et d_2 tels que $d_1 d_2 > 1$, de trouver deux polynômes f_1 et f_2 de degrés respectifs d_1 et d_2, à coefficients entiers, irréductibles sur $\mathbb{Q}$, premiers entre eux sur $\mathbb{Q}$ et ayant une racine commune modulo N. Une paire de polynômes qui satisfait tous ces critères sera dite adaptée à N. Si (f_1, f_2) est une paire de polynômes adaptée à N, alors, comme f_1 et f_2 sont premiers entre eux, le résultant $\mathrm{Res}(f_1, f_2)$ est non nul et, comme f_1 et f_2 ont une racine commune modulo N, le résultant $\mathrm{Res}(f_1, f_2)$ est nul modulo N. Donc, si (f_1, f_2) est une paire de polynômes adaptée à N, le résultant $\mathrm{Res}(f_1, f_2)$ est un multiple non nul de N.

Étant donnée une paire de polynômes adaptée à N, il est facile de générer d'autres paires adaptées à N à l'aide de deux transformations appelées *translations* et *rotations*. La translation

d'une paire de polynômes (f_1, f_2) est définie, pour tout entier k, comme la paire $(\tilde{f}_{1,k}, \tilde{f}_{2,k})$, où $\tilde{f}_{i,k}(X) = f_i(X + k)$ pour $i \in \{1, 2\}$. Il est facile de vérifier que si la paire de polynômes (f_1, f_2) est adaptée à N, alors, pour tout entier k, la paire $(\tilde{f}_{1,k}, \tilde{f}_{2,k})$ l'est aussi et en particulier $\mathrm{Res}(\tilde{f}_{1,k}, \tilde{f}_{2,k}) = \mathrm{Res}(f_1, f_2)$. La rotation d'une paire de polynômes (f_1, f_2) pour laquelle $\deg(f_1) \leqslant \deg(f_2)$ est définie, pour tout polynôme r à coefficients entiers de degré inférieur ou égal à $\deg(f_2) - \deg(f_1)$, comme la paire $(f_1, f_2 + rf_1)$. Si (f_1, f_2) est une paire de polynômes adaptée à N telle que $\deg(f_1) \leqslant \deg(f_2)$ et si r est un polynôme à coefficients entiers de degré inférieur ou égal à $\deg(f_2) - \deg(f_1)$ tel que $f_2 + rf_1$ est un polynôme irréductible de degré $\deg(f_2)$ alors la paire $(f_1, f_2 + rf_1)$ est adaptée à N et en particulier $\mathrm{Res}(f_1, f_2 + rf_1) = \mathrm{Res}(f_1, f_2)$.

L'efficacité de l'étape de collecte des relations dépend de la paire de polynômes sélectionnée lors de l'étape de sélection polynomiale, il est donc nécessaire d'avoir des grandeurs qui nous permettent d'estimer la qualité des différentes paires de polynômes produites lors de l'étape de sélection polynomiale afin de sélectionner la paire de polynômes qui produira le plus de relations lors de l'étape de crible.

3.1.2 Évaluer la qualité des polynômes

Rappelons quelques notations définies dans le chapitre 2 qui nous seront utiles dans cette section. La zone de crible, c'est-à-dire l'ensemble des paires d'entiers (a, b) parmi lesquelles les relations sont recherchées, est notée Ω. De plus, pour $i \in \{1, 2\}$, la borne supérieure sur les nombres premiers appartenant à la base de facteurs est noté B_i et le polynôme homogène associé à f_i est noté F_i et est défini par $F_i(X, Y) = f_i(X/Y)Y^{d_i}$.

Une paire de polynômes est de bonne qualité si elle produit un grand nombre de relations lors de l'étape de crible, c'est-à-dire si, pour un grand nombre de paires $(a, b) \in \Omega$ avec a et b premiers entre eux, $F_1(a, b)$ est B_1-friable et $F_2(a, b)$ est B_2-friable. Nous voulons donc estimer la probabilité de friabilité de $F_1(a, b)$ et $F_2(a, b)$ lorsque (a, b) parcourt la zone de crible.

Notons Ψ l'application qui compte le nombre d'entiers friables, c'est-à-dire Ψ est défini, pour x et y deux entiers strictement positifs, par

$$\Psi(x, y) = \#\{\, 1 \leqslant n \leqslant x \mid \text{n est } y\text{-friable} \,\}. \tag{3.1}$$

La fonction ρ de Dickman–de Bruijn [67], définie comme l'unique fonction vérifiant

$$u\rho'(u) + \rho(u - 1) = 0 \text{ pour } u > 1 \text{ et } \rho(u) = 1 \text{ pour } 0 \leqslant u \leqslant 1,$$

peut servir à estimer la densité de nombres friables $\Psi(x, x^{1/u})/x$, pour tout réel $u > 0$. En effet, il a été démontré [37, 50] que pour tout réel $u > 0$,

$$\lim_{x \to \infty} \frac{\Psi(x, x^{1/u})}{x} = \rho(u).$$

Une première grandeur permettant de juger la qualité d'une paire de polynômes est donnée par la formule ci-dessous. En supposant que, pour $i \in \{1, 2\}$ et pour $(a, b) \in \Omega$, $F_i(a, b)$ se comporte comme un entier aléatoire de taille $|F_i(a, b)|$, alors une approximation du nombre de relations qui sont trouvées par le crible est donnée par

$$\frac{6}{\pi^2} \iint_{\Omega} \rho\left(\frac{\log|F_1(a,b)|}{\log B_1}\right) \rho\left(\frac{\log|F_2(a,b)|}{\log B_2}\right) \mathrm{d}a\, \mathrm{d}b. \tag{3.2}$$

Le facteur constant $6/\pi^2$ représente la probabilité pour a et b d'être premiers entre eux et peut ne pas être pris en compte lorsqu'il s'agit de comparer des paires de polynômes entre elles. Dans

le cas de la sélection polynomiale linéaire, il est souvent considéré que la probabilité de friabilité du côté rationnel est constante sur toute la zone de crible et qu'il est donc suffisant de considérer, si le côté algébrique est le côté 2,

$$\iint_{\Omega} \rho\left(\frac{\log|F_2(a,b)|}{B_2}\right) \mathrm{d}a\,\mathrm{d}b. \tag{3.3}$$

Il n'existe pas de forme close pour ρ et le calcul d'une bonne approximation est coûteux, donc en pratique, comme ρ est une fonction décroissante, il est courant de faire l'approximation suivante : la probabilité que $F_i(a,b)$ soit B_i-friable dépend tout d'abord de la taille de $F_i(a,b)$. Une paire de polynômes sera donc considérée comme bonne si en moyenne sur la zone de crible les valeurs de $F_1(a,b)$ et $F_2(a,b)$ sont petites. Il existe différentes normes servant à mesurer la taille moyenne d'un polynôme, celle que nous utiliserons dans ce chapitre, notée $\|.\|_2$, est définie par

$$\|f_i\|_2 = \min_{s>0} \|f_i\|_{2,s} \quad \text{où} \quad \|f_i\|_{2,s} = \sqrt{\frac{1}{\pi s^{d_i}} \int_0^{2\pi} \int_0^1 F_i^2(s\cos\theta, \sin\theta) r^{2d_i+1}\,\mathrm{d}r\,\mathrm{d}\theta}. \tag{3.4}$$

Le paramètre s qui minimise $\|f_i\|_{2,s}$ dans la définition est appelé la *skewness*[3] du polynôme. Elle sert à prendre en compte le fait que la zone de crible n'est pas exactement circulaire mais plutôt elliptique. La *skewness* peut être vue comme la racine carrée du rapport entre le grand axe et le petit axe de l'ellipse par laquelle la zone de crible est la mieux approchée. Si f est un polynôme de degré d, alors l'expression de $\|f\|_{2,s}^2$ est une fraction rationnelle en s de dénominateur $c(d)s^d$, où $c(d)$ est une constante entière ne dépendant que de d, et de numérateur un polynôme pair de degré $2d$ en s dont les coefficients sont des polynômes à coefficients entiers en les coefficients de f. Par exemple, pour $f = a_3X^3 + a_2X^2 + a_1X + a_0$,

$$\|f\|_{2,s}^2 = \frac{5a_3^2s^6 + (a_2^2 + 2a_1a_3)s^4 + (a_1^2 + 2a_0a_2)s^2 + 5a_0^2}{64s^3}.$$

Le produit des normes de f_1 et f_2 est utilisé comme première approximation pour déterminer la qualité d'une paire de polynômes mais ce n'est pas suffisant. Pour être plus précis, il faut revenir sur l'hypothèse que $F_i(a,b)$ se comporte comme un entier aléatoire. Si c'était le cas alors la valuation moyenne de $F_i(a,b)$ pour toutes les paires $(a,b) \in \Omega$ en un nombre premier p serait $1/(p-1)$. Boender, Brent, Montgomery et Murphy [24, 112–114] ont proposé plusieurs grandeurs pour mesurer l'écart entre la valuation moyenne si $F_i(a,b)$ se comportait comme un entier aléatoire et la valuation moyenne attendue du fait que les entiers $F_i(a,b)$ sont obtenus comme évaluation d'un polynôme. Ceci a abouti à la définition de la grandeur α qui mesure la contribution logarithmique pour chaque nombre premier de l'écart entre la valuation moyenne d'un entier aléatoire et la valuation moyenne pour un entier provenant de l'évaluation d'un polynôme. Si f est un polynôme à coefficients entiers, la grandeur α est définie par

$$\alpha(f) = \sum_{p\in\mathcal{P}} \left(\frac{1}{p-1} - \frac{1}{p+1} \sum_{k=1}^{\infty} \frac{n_{p^k}(f)}{p^{k-1}}\right) \log p \tag{3.5}$$

où $n_{p^k}(f)$ est le nombre de racines (affines et projectives) de f dans $\mathbb{Z}/p^k\mathbb{Z}$, comptées avec multiplicités, pour un nombre premier p et un entier strictement positif k. Pour la sélection

3. À défaut de traduction satisfaisante en français, le mot anglais est utilisé dans ce chapitre.

polynomiale, nous voulons des polynômes avec des valeurs de α négatives et grandes en valeurs absolues. Comme les grands nombres premiers et les grandes puissances contribuent peu à la somme totale, en pratique, une approximation de α est calculée où ils ne sont pas pris en compte.

La norme sert à mesurer la taille des polynômes et la grandeur α sert à mesurer *les propriétés des racines* des polynômes. Une grandeur, appelée *score combiné*, réunit ces deux mesures en une seule et est définie, pour un polynôme f à coefficients entiers, par :

$$\log\|f\|_2 + \alpha(f). \tag{3.6}$$

Pour la sélection polynomiale, nous voulons des polynômes dont le score combiné est le plus petit possible. Cette grandeur motive la définition de la *log-norme* d'un polynôme f par $\log\|f\|_2$.

Enfin, la valeur E de Murphy est une approximation du calcul de la grandeur définie par l'équation (3.2) dans laquelle la grandeur α est utilisée pour prendre en compte les propriétés des racines des polynômes. La zone de crible Ω est approchée par une ellipse de centre $(0,0)$, de grand rayon rs et de petit rayon r/s, où s est la *skewness* de la paire de polynômes et r est tel que l'aire de Ω vaut πr^2. Ensuite, K points $(x_k, y_k) = (sr\cos\theta_k, r/s\sin\theta_k)$ régulièrement espacés sur cette ellipse sont utilisés. La valeur E de Murphy pour la paire de polynômes (f_1, f_2) et les bornes de friabilité B_1 et B_2 est définie par

$$E = \frac{1}{K}\sum_{k=1}^{K} \rho\left(\frac{\log|F_1(x_k,y_k)| + \alpha(f_1)}{\log B_1}\right)\rho\left(\frac{\log|F_2(x_k,y_k)| + \alpha(f_2)}{\log B_2}\right). \tag{3.7}$$

La valeur E de Murphy étant une approximation de la probabilité pour une paire (a,b) d'être une relation, nous voulons, pour la sélection polynomiale, des paires de polynômes pour lesquelles la valeur E de Murphy est la plus grande possible.

La norme d'un polynôme est peu coûteuse à calculer et sert comme première approximation pour repérer les bonnes paires de polynômes. Ensuite, la grandeur α, un peu plus coûteuse à calculer, est utilisée pour faire un tri supplémentaire. Et finalement, le valeur E de Murphy, plus coûteuse à calculer mais plus précise, permet de choisir la meilleure paire de polynômes parmi celles gardées par les tris précédents.

3.1.3 La sélection polynomiale linéaire

Dans cette section nous allons décrire la sélection polynomiale dans le cas linéaire, c'est-à-dire dans le cas où $d_1 = 1$ et $d_2 > 1$. Pour la sélection polynomiale linéaire, seulement le polynôme algébrique, c'est-à-dire f_2, est utilisé pour comparer les paires de polynômes entre elles, que ce soit du point de vue de la norme ou du point de vue de la grandeur α. En effet, dans le cas de la sélection polynomiale linéaire, la norme du polynôme linéaire, c'est-à-dire f_1, est plus petite que celle du polynôme algébrique et la grandeur α d'un polynôme linéaire dont les coefficients sont premiers entre eux est constante, car pour tout nombre premier p et pour tout entier positif non nul k, $n_{p^k}(f_1) = 1$.

L'étape de sélection polynomiale dans le cas linéaire peut se décomposer en trois sous-étapes : la génération de paires de polynômes, l'optimisation de la taille et l'optimisation des propriétés des racines. La génération de paires de polynômes peut être effectuée à l'aide des algorithmes de Kleinjung [79, 81] et est décrite dans la section 3.2.

Le problème de l'optimisation de la taille est le suivant : étant donnée une paire de polynômes (f_1, f_2) adaptée à N, il faut trouver parmi toutes les paires de polynômes qui peuvent être obtenues par translations et rotations celle dont la norme du polynôme algébrique est la plus

petite. Cela revient à résoudre le problème d'optimisation suivant :

$$\underset{k\in\mathbb{Z},r\in\mathbb{Z}_\delta[X]}{\operatorname{argmin}} \left\| \tilde{f}_{2,k} + r\tilde{f}_{1,k} \right\|_2 = \underset{k\in\mathbb{Z},r\in\mathbb{Z}_\delta[X],s>0}{\operatorname{argmin}} \left\| \tilde{f}_{2,k} + r\tilde{f}_{1,k} \right\|_{2,s}, \tag{3.8}$$

où $\tilde{f}_{i,k}(X) = f_i(X+k)$ pour $i \in \{1, 2\}$, $\delta < d_2$ est une borne sur le degré des rotations et $\mathbb{Z}_\delta[X]$ est l'ensemble des polynômes à coefficients entiers de degré inférieur ou égal à δ. Dans la section 3.3, l'algorithme utilisé pour résoudre ce problème est présenté et des améliorations sont proposées.

La dernière sous-étape consiste à trouver, parmi toutes les paires de polynômes qui peuvent être obtenues par translations et rotations, celle dont la valeur E de Murphy est la plus grande. Pour cela, il faut commencer par chercher parmi toutes les paires de polynômes qui peuvent être obtenues par translations et rotations celle dont le score combiné du polynôme algébrique est le plus petit possible, c'est-à-dire les paires pour lesquelles la grandeur α peut être diminuée sans trop augmenter la norme du polynôme algébrique. Ensuite parmi les paires avec le meilleur score combiné, celle avec la valeur E de Murphy la plus grande est gardée. Une description détaillée du problème de l'optimisation des propriétés des racines peut être trouvée dans [6].

3.2 Génération de polynômes dans le cas linéaire

Dans cette section, nous nous intéressons aux algorithmes de Kleinjung pour la génération de polynômes dans le cas linéaire. Dans toute cette section, nous aurons donc $d_1 = 1$, $d_2 > 1$ et nous noterons g le polynôme rationnel, f le polynôme algébrique et d son degré. Si f est un polynôme de degré d à coefficients entiers, irréductible sur $\mathbb{Q}$ et admettant une racine $m = m_1/m_2$ modulo N, alors, en posant $g = m_2X - m_1$, la paire de polynômes (f, g) est adaptée à N. Le problème de la sélection polynomiale linéaire est donc de trouver un polynôme f de degré d à coefficients entiers, irréductible sur $\mathbb{Q}$ et admettant une racine modulo N, c'est-à-dire tel qu'il existe deux entiers m_1 et m_2 premiers entre eux tels que $f(m_1/m_2)m_2^d$ est égal à un multiple non nul de N.

Les deux polynômes f et $-f$ sont équivalents en terme de taille, du corps de nombres qu'ils définissent pour l'algorithme NFS et en terme de relations produites lors de l'étape de crible de l'algorithme NFS, alors dans la suite nous choisirons un seul de ces deux polynômes en supposant que le coefficient dominant du polynôme algébrique est positif.

Deux algorithmes de Kleinjung sont utilisés pour générer des paires de polynômes pour la sélection polynomiale linéaire. Le premier algorithme, décrit dans la section 3.2.1, montre comment, à partir de m_1, m_2 et des coefficients de plus haut degré du polynôme f, il est possible de reconstruire complètement le polynôme f. Le second algorithme de Kleinjung, décrit dans la section 3.2.2, montre comment trouver des entiers m_1 et m_2 pour lesquels l'algorithme précédent peut être utilisé.

Le fait que le résultant des polynômes f et g est un multiple non nul de N impose que le produit $\|f\|_2\|g\|_2^d$ soit en $\Omega(N)$. Dans la suite nous chercherons donc à construire des paires de polynômes telles que les normes $\|f\|_2$ et $\|g\|_2$ sont en $\mathrm{O}(N^{1/(d+1)})$.

3.2.1 Le premier algorithme de Kleinjung

L'algorithme 3.1 est une version plus générale du premier algorithme de Kleinjung pour la sélection polynomiale linéaire. Il montre comment, étant donnés les coefficients de plus haut degré d'un polynôme et deux entiers m_1 et m_2, reconstruire un polynôme ayant m_1/m_2 comme racine modulo N. Le théorème 3.2.1 donne les conditions nécessaires pour que l'algorithme 3.1

renvoie un polynôme à coefficients entiers admettant une racine modulo N, et donne une borne sur la taille de certains coefficients du polynôme. L'algorithme 3.1 et le théorème 3.2.1 décrits dans cette section sont basés sur [79, lemme 2.1] et [43, algorithme 4.3].

Algorithme 3.1 Premier algorithme de Kleinjung

1: **procédure** KleinjungReconstruct(N, d, m_1, m_2, j, $[a_j, \ldots, a_d]$)
2: Calculer
$$r_j \leftarrow \frac{N - \sum_{i=j+1}^{d} a_i m_1^i m_2^{d-i}}{m_2^{d-j}}$$
3: **pour** $i = j-1, j-2, \ldots, 0$ **faire**
4: $r_i \leftarrow \dfrac{r_{i+1} - a_{i+1} m_1^{i+1}}{m_2}$
5: $a_i \leftarrow \dfrac{r_i + t_i m_2}{m_1^i}$, où t_i est un entier tel que $-\dfrac{m_1^i}{2} \leqslant t_i < \dfrac{m_1^i}{2}$ et $t_i \equiv -\dfrac{r_i}{m_2} \bmod m_1^i$
6: **renvoyer** le polynôme $a_d X^d + a_{d-1} X^{d-1} + \cdots + a_1 X + a_0$

Théorème 3.2.1. *Soient N, m_1, m_2, d et j des entiers positifs non nuls et $a_j, \ldots, a_d$ des entiers tels que $j \leqslant d$, $a_d > 0$, $\sum_{i=j}^{d} a_i m_1^i m_2^{d-i} \equiv N \pmod{m_2^{d-j+1}}$, $\gcd(m_1, m_2) = 1$ et $\gcd(a_d, N) = 1$. Alors le polynôme $f = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_1 X + a_0$ renvoyé par l'algorithme* KleinjungReconstruct *est un polynôme à coefficients entiers, de degré d tel que :*

1. *$f(m_1/m_2) m_2^d = N$,*
2. *$|a_i| \leqslant (m_1 + m_2)/2$ pour $0 \leqslant i \leqslant j-2$.*

Démonstration. Remarquons tout d'abord que les hypothèses assurent que r_j est un entier. Ensuite, une récurrence sur $i = j-1, j-2, \ldots, 0$ permet de montrer simultanément les trois propositions suivantes, pour $0 \leqslant i \leqslant j-1$:

— r_i est un entier,
— a_i est un entier,
— $N = a_d m_1^d + a_{d-1} m_1^{d-1} m_2 + \cdots + a_{i+1} m_1^{i+1} m_2^{d-i-1} + r_i m_2^{d-i}$.

Ceci montre que le polynôme f est un polynôme à coefficients entiers. De plus, en utilisant la dernière proposition avec $i = 0$ et en remarquant que $a_0 = r_0$, nous obtenons que $f(m_1/m_2) m_2^d = N$.

Pour démontrer la borne sur la taille des coefficients, il faut remarquer que pour $0 \leqslant i \leqslant j-2$,

$$|r_i| = \frac{\left|r_{i+1} - a_{i+1} m_1^{i+1}\right|}{m_2} = \frac{|t_{i+1} m_2|}{m_2} \leqslant \frac{m_1^{i+1}}{2}.$$

Nous avons donc, pour $0 \leqslant i \leqslant j-2$,

$$|a_i| = \frac{|r_i + t_i m_2|}{m_1^i} \leqslant \frac{|r_i|}{m_1^i} + \frac{|t_i| m_2}{m_1^i} \leqslant \frac{m_1}{2} + \frac{m_2}{2}.$$

□

Pour un polynôme produit par l'algorithme 3.1, la taille de tous ses coefficients, sauf celle du coefficient de degré $j-1$, est connue. La taille des coefficients $a_j, \ldots, a_d$ est connue car ce sont des entrées de l'algorithme et une borne sur la taille des coefficients $a_0, \ldots, a_{j-2}$ est donnée par le théorème précédent. Il reste uniquement à connaître la taille de a_{j-1} pour pouvoir estimer la norme du polynôme.

Exemple 3.2.2. L'algorithme de sélection polynomiale le plus simple utilisant l'algorithme 3.1 est le cas où $j = d$. Il suffit de choisir un entier positif non nul a_d pour le coefficient dominant du polynôme et un entier positif non nul m_2 tel qu'il est possible de calculer des racines de polynômes modulo m_2 (par exemple, en choisissant m_2 comme produit de petits nombres premiers). Ensuite, en définissant $\bar{m} = \sqrt[d]{N/a_d}$, il reste à calculer un entier m_1 supérieur ou égale à $\bar{m}$ tel que $a_d m_1^d \equiv N \pmod{m_2}$. Alors, les conditions du théorème sont réunies (si m_1 et m_2 ne sont pas premiers entre eux, alors nous avons trouvé un facteur de N) et nous pouvons donc construire une paire de polynômes adaptée à N. Le seul coefficient pour lequel nous n'avons aucune information sur la taille est le coefficient a_{d-1}. Pour obtenir une borne sur la taille de a_{d-1}, il faut d'abord borner l'entier r_{d-1} défini par l'algorithme 3.1 :

$$|r_{d-1}| = \frac{|N - a_d m_1^d|}{m_2} = \frac{|a_d \bar{m}^d - a_d m_1^d|}{m_2} = \frac{a_d |\bar{m}^d - m_1^d|}{m_2} \leqslant \frac{d a_d |\bar{m} - m_1| m_1^{d-1}}{m_2},$$

où la dernière inégalité est due au fait que $\bar{m} \leqslant m_1$. Grâce à cela, nous obtenons la borne suivante pour a_{d-1} :

$$|a_{d-1}| \leqslant \frac{m_2}{2} + d a_d \frac{|\bar{m} - m_1|}{m_2}. \tag{3.9}$$

En choisissant a_d et m_2 de taille $\mathrm{O}(N^{1/(d+1)})$ et m_1 tel que $0 \leqslant m_1 - \bar{m} \leqslant m_2$, ce qui est toujours possible si N admet une racine $d^{\text{ième}}$ modulo m_2, alors l'algorithme 3.1 avec $j = d$ produit un polynôme dont la norme est en $\mathrm{O}(N^{1/(d+1)})$ pour une *skewness* égale à 1.

3.2.2 Le second algorithme de Kleinjung

Le second algorithme de Kleinjung pour la sélection polynomiale linéaire permet de trouver des entiers m_1, m_2, a_d et a_{d-1} et utilise l'algorithme 3.1 avec $j = d - 1$. Le but est de choisir les coefficients de degré d et $d - 1$ et de contrôler la taille du coefficient a_{d-2} grâce à une équation similaire à l'équation (3.9) de l'exemple 3.2.2. Le second algorithme de Kleinjung a été décrit lors d'une présentation au workshop CADO d'octobre 2008 [81].

Étant donnés N, d et a_d tels que $\gcd(a_d, N) = 1$, nous voulons trouver des entiers m_1, m_2 et a_{d-1} tels que a_{d-1} est «petit», $\gcd(m_1, m_2) = 1$ et

$$a_d m_1^d + a_{d-1} m_1^{d-1} m_2 \equiv N \pmod{m_2^2}. \tag{3.10}$$

Pour cela, nous pouvons nous ramener au problème de trouver deux entiers $\tilde{m}_1$ et m_2 tels que $\gcd(m_2, \tilde{m}_1 d a_d) = 1$ et $\tilde{m}_1^d \equiv \tilde{N} \pmod{m_2^2}$, pour un certain $\tilde{N}$ dépendant de N. En effet, en multipliant l'équation (3.10) par $d^d a_d^{d-1}$ et en posant $\tilde{N} = d^d a_d^{d-1} N$, nous obtenons l'égalité

$$(d a_d m_1)^d + d a_{d-1} (d a_d m_1)^{d-1} m_2 \equiv \tilde{N} \pmod{m_2^2}, \tag{3.11}$$

dans laquelle apparaît le début du développement de $(d a_d m_1 + a_{d-1} m_2)^d$. Donc en posant $\tilde{m}_1 = d a_d m_1 + a_{d-1} m_2$, nous obtenons

$$\tilde{m}_1^d \equiv \tilde{N} \pmod{m_2^2}. \tag{3.12}$$

Si $\tilde{m}_1$ et m_2 sont deux entiers satisfaisant l'équation (3.12) tels que $\gcd(m_2, \tilde{m}_1 d a_d) = 1$, alors l'entier a_{d-1} peut être retrouvé comme l'unique entier vérifiant $-(d a_d)/2 \leqslant a_{d-1} < (d a_d)/2$ et $a_{d-1} \equiv \tilde{m}_1 / m_2 \pmod{d a_d}$ et l'entier m_1 vaut alors $m_1 = (\tilde{m}_1 - a_{d-1} m_2)/(d a_d)$. Les conditions du théorème 3.2.1 sont satisfaites et donc l'algorithme 3.1 peut être utilisé pour construire un polynôme de degré d ayant m_1/m_2 comme racine modulo N.

En utilisant le théorème 3.2.1 avec $j = d-1$ nous obtenons une borne sur la taille des coefficients a_i, pour $0 \leqslant i \leqslant d-3$ mais nous n'avons aucune information sur la taille du coefficient a_{d-2}. Avant d'estimer la taille du coefficient a_{d-2}, nous allons faire les suppositions suivantes : $(m_1+m_2)^{d-1} \leqslant dm_1^{d-1}$ et $(m_1+m_2)^{d-2} \leqslant (d-1)m_1^{d-2}$. Ces suppositions se basent sur le fait que, en pratique, le deuxième algorithme de Kleinjung est utilisé pour générer des polynômes avec une grande *skewness*, ce qui impose que m_2 soit très petit par rapport à m_1. Pour estimer la taille du coefficient a_{d-2}, nous allons d'abord estimer la taille de R défini comme la différence entre $\tilde{m}_1^d$ et $(da_dm_1)^d + da_{d-1}(da_dm_1)^{d-1}m_2$:

$$\begin{aligned}
|R| &= \left|\tilde{m}_1^d - ((da_dm_1)^d + da_{d-1}(da_dm_1)^{d-1}m_2)\right| = \left|\sum_{i=0}^{d-2}\binom{d}{i}(da_dm_1)^i(a_{d-1}m_2)^{d-i}\right| \\
&\leqslant \sum_{i=0}^{d-2}\binom{d}{i}(da_dm_1)^i(|a_{d-1}|m_2)^{d-i} \leqslant (da_d)^d\sum_{i=0}^{d-2}\binom{d}{i}m_1^im_2^{d-i} \qquad (\text{car } |a_{d-1}| \leqslant da_d) \\
&\leqslant (da_d)^d d(d-1)m_2^2\sum_{i=0}^{d-2}\binom{d-2}{i}m_1^im_2^{d-2-i} \leqslant d(d-1)^2(da_d)^dm_1^{d-2}m_2^2.
\end{aligned}$$

Il est aussi nécessaire d'estimer $\tilde{m}_1^{d-1}$:

$$\tilde{m}_1^{d-1} = \sum_{i=0}^{d-1}\binom{d-1}{i}(da_dm_1)^i(a_{d-1}m_2)^{d-1-i} \leqslant (da_d)^{d-1}\sum_{i=0}^{d-1}\binom{d-1}{i}m_1^im_2^{d-1-i} \leqslant d^da_d^{d-1}m_1^{d-1}.$$

En supposant que $\tilde{m}_1 \geqslant \bar{m}$, où $\bar{m} = \sqrt[d]{\tilde{N}}$, nous pouvons maintenant estimer r_{d-2} en fonction de la différence entre $\tilde{m}_1$ et $\bar{m}$:

$$\begin{aligned}
|r_{d-2}| &= \frac{\left|N - (a_dm_1^d + a_{d-1}m_1^{d-1}m_2)\right|}{m_2^2} \\
&= \frac{\left|d^da_d^{d-1}N - ((da_dm_1)^d + da_{d-1}(da_dm_1)^{d-1}m_2)\right|}{d^da_d^{d-1}m_2^2} = \frac{\left|\tilde{N} - (\tilde{m}_1^d - R)\right|}{d^da_d^{d-1}m_2^2} \\
&\leqslant \frac{\left|\tilde{N} - \tilde{m}_1^d\right|}{d^da_d^{d-1}m_2^2} + \frac{|R|}{d^da_d^{d-1}m_2^2} \leqslant \frac{d|\bar{m} - \tilde{m}_1|\tilde{m}_1^{d-1}}{d^da_d^{d-1}m_2^2} + \frac{|R|}{d^da_d^{d-1}m_2^2} \\
&\leqslant \frac{d|\bar{m} - \tilde{m}_1|m_1^{d-1}}{m_2^2} + d(d-1)^2a_dm_1^{d-2},
\end{aligned}$$

ce qui nous permet d'obtenir une borne sur la taille de a_{d-2} :

$$|a_{d-2}| = \frac{|r_{d-2} + t_{d-2}m_2|}{m_1^{d-2}} \leqslant \frac{|r_{d-2}|}{m_1^{d-2}} + \frac{|t_{d-2}|m_2}{m_1^{d-2}} \leqslant \frac{d|\bar{m} - \tilde{m}_1|m_1}{m_2^2} + \frac{m_2}{2} + d(d-1)^2a_d. \tag{3.13}$$

Afin que la taille du coefficient a_{d-2} soit petite, nous allons chercher des solutions entières de taille similaire à celle de m_2 à l'équation $(\tilde{m}_0 + x)^d \equiv \tilde{N} \pmod{m_2^2}$, où $\tilde{m}_0 = \lfloor\bar{m}\rfloor = \left\lfloor\sqrt[d]{\tilde{N}}\right\rfloor$. Le second algorithme de Kleinjung, dont une description est donnée dans l'algorithme 3.2, propose de faire cela en cherchant deux nombres entiers p_1 et p_2 tels qu'il existe un entier r satisfaisant $(\tilde{m}_0 + r)^d \equiv \tilde{N} \pmod{p_i^2}$, pour $i \in \{1, 2\}$.

Algorithme 3.2 Second algorithme de Kleinjung

Entrée : N, d, a_d et P quatre entiers non nuls tels que $\text{pgcd}(a_d, N) = 1$
Sortie : Un ensemble de paires de polynômes ayant une racine commune modulo N et dont le polynôme algébrique est de degré d avec a_d comme coefficient dominant

1: $\tilde{N} \leftarrow d^d a_d^{d-1} N$ et $\tilde{m}_0 \leftarrow \left\lfloor \sqrt[d]{\tilde{N}} \right\rceil$
2: $\mathcal{C} \leftarrow \varnothing$
3: **pour** p nombre premier dans $[P, 2P]$ **faire**
4: **pour** r dans $[0, p^2 - 1]$ vérifiant $(\tilde{m}_0 + r)^d \equiv \tilde{N} \pmod{p^2}$ **faire**
5: Ajouter (p, r) à $\mathcal{C}$
6: $\mathcal{F} \leftarrow \varnothing$
7: **pour** chaque collision (p_1, r), (p_2, r) dans $\mathcal{C}$ **faire**
8: $\tilde{m}_1 \leftarrow \tilde{m}_0 + r$ et $m_2 \leftarrow p_1 p_2$
9: $a_{d-1} \leftarrow \tilde{m}_1 / m_2 \bmod d a_d$
10: $m_1 \leftarrow (\tilde{m}_1 - a_{d-1} m_2)/(d a_d)$
11: $g \leftarrow m_2 X - m_1$
12: $f \leftarrow \text{KleinjungReconstruct}(N, d, m_1, m_2, d-1, [a_{d-1}, a_d])$
13: Ajouter (f, g) à $\mathcal{F}$
14: **renvoyer** $\mathcal{F}$

En utilisant l'algorithme 3.2, nous obtenons un entier m_2 compris entre P^2 et $4P^2$ et nous pouvons borner la différence $|\bar{m} - \tilde{m}_1|$ par $4P^2$, ce qui permet d'obtenir, en utilisant l'équation (3.13), la borne suivante pour la taille du coefficient a_{d-2} :

$$|a_{d-2}| \leqslant \frac{4dm_1}{P^2} + 2P^2 + d(d-1)^2 a_d. \tag{3.14}$$

Nous allons maintenant estimer le nombre moyen de collisions dans l'algorithme 3.2. Le nombre de nombres premiers dans l'intervalle $[P, 2P]$ est en $\Theta(P/\log(P))$. Une fois toutes les racines calculées, la taille de l'ensemble $\mathcal{C}$ est encore en $\Theta(P/\log(P))$. Si (p, r) est un élément de l'ensemble $\mathcal{C}$ alors l'entier r, pour lequel nous recherchons des collisions, appartient à l'intervalle $[0, 4P^2]$. Le nombre moyen de collisions est donc en $\Theta((P/\log(P))^2/(4P^2))$, c'est-à-dire en $\Theta(1/\log^2(P))$. Il faut donc utiliser environ $O(\log^2(P))$ fois l'algorithme 3.2, avec des valeurs de a_d différentes, pour espérer obtenir une collision.

3.2.3 Utilisation de spécial-q

Dans cette section, nous allons voir comment augmenter le nombre moyen de collisions de l'algorithme 3.2. Pour ce faire nous allons utiliser des *spécial-q* pour forcer un entier q à diviser m_2. Ceci entraîne une augmentation de m_2, qui est compris entre qP^2 et $4qP^2$, et la différence $|\bar{m} - \tilde{m}_1|$ est bornée par $4q^2P^2$, ce qui permet d'obtenir la borne suivante pour le coefficient a_{d-2} :

$$|a_{d-2}| \leqslant \frac{4dm_1}{P^2} + 2qP^2 + d(d-1)^2 a_d. \tag{3.15}$$

Supposons connu un entier q qui n'a aucun facteur premier dans $[P, 2P]$ et un entier r_q qui vérifie $(\tilde{m}_0 + r_q)^d \equiv \tilde{N} \pmod{q^2}$. Étant donné un nombre premier $p \in [P, 2P]$ et un entier r_p tel que $(\tilde{m}_0 + r_p)^d \equiv \tilde{N} \pmod{p^2}$, alors $(\tilde{m}_0 + r_q + i_p q^2)^d \equiv \tilde{N} \pmod{q^2 p^2}$ où $i_p = (r_p - r_q)/q^2 \bmod p^2$. Le but de l'algorithme utilisant les spécial-q, décrit dans l'algorithme 3.3, est de trouver deux nombres premiers p_1 et p_2 tels que $i_{p_1} = i_{p_2}$.

Algorithme 3.3 Second algorithme de Kleinjung : variante avec spécial-q

Entrée : N, d, a_d et P quatre entiers non nuls tels que $\text{pgcd}(a_d, N) = 1$ et $\mathcal{Q}$ un ensemble de nombres premiers dont l'intersection avec $[P, 2P]$ est vide

Sortie : Un ensemble de paires de polynômes ayant une racine commune modulo N et dont le polynôme algébrique est de degré d avec a_d comme coefficient dominant

$\tilde{N} \leftarrow d^d a_d^{d-1} N$ et $\tilde{m}_0 \leftarrow \left\lfloor \sqrt[d]{\tilde{N}} \right\rfloor$
$\mathcal{F} \leftarrow \varnothing$, $\mathcal{R}_p \leftarrow \varnothing$
pour p nombre premier dans $[P, 2P]$ **faire**
    pour r_p dans $[0, p^2 - 1]$ vérifiant $(\tilde{m}_0 + r_p)^d \equiv \tilde{N} \pmod{p^2}$ **faire**
        Ajouter (p, r_p) à $\mathcal{R}_p$
pour chaque sous-ensemble $\{q_1, \ldots, q_n\}$ de $\mathcal{Q}$ **faire**
    $q \leftarrow q_1 \times \cdots \times q_n$
    pour r_q dans $[0, q^2 - 1]$ vérifiant $(\tilde{m}_0 + r_q)^d \equiv \tilde{N} \pmod{q^2}$ **faire**
        $\mathcal{C} \leftarrow \varnothing$
        pour (p, r_p) dans $\mathcal{R}_p$ **faire**
            $i \leftarrow (r_p - r_q)/q^2 \bmod p^2$
            Ajouter (p, i) à $\mathcal{C}$
        pour chaque collision (p_1, i), (p_2, i) dans $\mathcal{C}$ **faire**
            $\tilde{m}_1 \leftarrow \tilde{m}_0 + r_q + iq^2$ et $m_2 \leftarrow qp_1p_2$
            $a_{d-1} \leftarrow \tilde{m}_1/m_2 \bmod da_d$
            $m_1 \leftarrow (\tilde{m}_1 - a_{d-1}m_2)/(da_d)$
            $g \leftarrow m_2X - m_1$
            $f \leftarrow \textsc{KleinjungReconstruct}(N, d, m_1, m_2, d-1, [a_{d-1}, a_d])$
            Ajouter (f, g) à $\mathcal{F}$
renvoyer $\mathcal{F}$

Pour une paire (q, r_q) donnée, le nombre moyen de collisions reste en $\Theta(1/\log^2(P))$. Mais par contre, pour un entier a_d donné, le nombre de paires (q, r_q) pouvant être utilisées peut être très grand. En effet, si $\mathcal{Q}$ est choisi pour ne contenir que des nombres premiers q_i tels que le polynôme $X^d - \tilde{N}$ admet d racines modulo q_i, alors, pour un entier q fixé, le nombre de valeurs de r_q possible est d^n, où n est le nombre de facteurs q_i utilisés.

La présentation de l'algorithme 3.3 n'est pas optimale et plusieurs améliorations peuvent être utilisées. Par exemple, les racines de $X^d - \tilde{N}$ modulo tous les q_i dans $\mathcal{Q}$ peuvent être précalculées et les entiers r_q seront ensuite calculés à l'aide du théorème des restes chinois. De plus, l'inverse de q^2 modulo tous les nombres premiers p dans $[P, 2P]$ peut être calculé une seule fois et réutilisé pour toutes les valeurs de r_q. Pour un entier q fixé, le calcul de tous ces inverses peut être fait efficacement en utilisant une variante due à Kruppa de l'algorithme d'inversion par lots de Montgomery [103]. En notant $\{ p_1, \ldots, p_k \}$ les nombres premiers de l'intervalle $[P, 2P]$, ce que nous voulons calculer est l'ensemble $\{ u_1, \ldots, u_k \}$ où $u_i = 1/q^2 \bmod p_i^2$. Il faut tout d'abord calculer les entiers $v_i = 1/p_i^2 \bmod q^2$, pour $1 \leqslant i \leqslant k$, en utilisant l'algorithme classique d'inversion par lots qui permet de faire cela en calculant une seule inversion modulaire et $3(k-1)$ multiplications. Finalement, il suffit de remarquer que les entiers u_i que nous recherchons peuvent être obtenus simplement en calculant $u_i = (1 - p_i^2 v_i)/q^2$, où la division est exacte.

3.3 Amélioration de l'algorithme d'optimisation de la taille

Dans cette section nous allons nous intéresser au problème de l'optimisation de la taille pour la sélection polynomiale linéaire, c'est-à-dire à la résolution du problème posé par l'équation (3.8). Pour chaque paire de polynômes générée, l'algorithme d'optimisation de la taille est utilisé pour diminuer la norme du polynôme algébrique en utilisant des translations et des rotations. Dans la suite de cette section, nous ne considérerons que les rotations de degré au plus δ avec $\delta = d-3$, car nous supposerons que les polynômes sont produits par les algorithmes de Kleinjung décrits dans la section précédente qui nous assure que a_d et a_{d-1} sont petits, car a_d est choisi et $|a_{d-1}| \leqslant d a_d$. De plus, pour garder les mêmes notations que précédemment, le polynôme rationnel d'une paire de polynômes produit par les algorithmes de Kleinjung sera noté $g = m_2 X - m_1$.

Les résultats de cette section sont tirés de l'article [5], écrit en collaboration avec S. Bai, A. Kruppa et P. Zimmermann.

3.3.1 Descente locale

L'algorithme 3.4 est un algorithme d'optimisation locale utilisé pour résoudre le problème de l'optimisation de la taille.

L'algorithme 3.4 s'arrête lorsqu'il a atteint un minimum *local*. Pour les petits degrés, c'est-à-dire pour $d \in \{ 3, 4, 5 \}$, le minimum local est souvent suffisamment petit pour qu'il ne soit pas nécessaire de se préoccuper de savoir s'il est éloigné du minimum global. Pour contre, pour les degrés plus grands, c'est-à-dire $d \in \{ 6, 7 \}$, l'algorithme de descente locale est souvent bloqué dans des minima locaux éloignés du minimum global (voir, par exemple, la comparaison de la figure 3.1).

Nous allons donc proposer dans les sections suivantes des améliorations de cet algorithme, en particulier pour les degrés $d = 6$ et $d = 7$. Toutes ces améliorations se basent sur l'idée d'utiliser l'algorithme de descente locale avec plusieurs points de départs assez éloignés pour espérer se rapprocher du minimum global.

Algorithme 3.4 Algorithme de descente locale pour l'optimisation de la taille

Entrée : Deux polynômes f et g tels que $\deg g = 1$ et $\deg f > 1$.
Sortie : Deux polynômes f_{opt} et g_{opt} tels que $\deg g_{\text{opt}} = 1$, $\deg f_{\text{opt}} = \deg f$, $\|f_{\text{opt}}\|_2 \leqslant \|f\|_2$ et $\text{Res}(f_{\text{opt}}, g_{\text{opt}}) = \text{Res}(f, g)$

1: $\delta \leftarrow d - 3$, $k \leftarrow 1$, $r_i \leftarrow 1$ pour $0 \leqslant i \leqslant \delta$ et $c \leftarrow$ VRAI
2: **tant que** $c =$ VRAI **faire**
3: $c_k \leftarrow$ FAUX et $c_{r_i} \leftarrow$ FAUX pour $0 \leqslant i \leqslant \delta$
4: **si** $\|f(X+k)\|_2 < \|f\|_2$ **alors**
5: $(f, g) \leftarrow (f(X+k), g(X+k))$
6: $c_k \leftarrow$ VRAI
7: **sinon si** $\|f(X-k)\|_2 < \|f\|_2$ **alors**
8: $(f, g) \leftarrow (f(X-k), g(X-k))$
9: $c_k \leftarrow$ VRAI
10: **pour** $i \in [0, \delta]$ **faire**
11: **si** $\left\|f + r_i X^i g\right\|_2 < \|f\|_2$ **alors**
12: $(f, g) \leftarrow (f + r_i X^i g, g)$
13: $c_{r_i} \leftarrow$ VRAI
14: **sinon si** $\left\|f - r_i X^i g\right\|_2 < \|f\|_2$ **alors**
15: $(f, g) \leftarrow (f - r_i X^i g, g)$
16: $c_{r_i} \leftarrow$ VRAI
17: **si** $c_k =$ FAUX et $k = 1$ et $c_{r_0} =$ FAUX et $r_0 = 1$ et … et $c_{r_\delta} =$ FAUX et $r_\delta = 1$ **alors**
18: $c \leftarrow$ FAUX
19: **sinon**
20: $c \leftarrow$ VRAI
21: **si** $c_k =$ VRAI **alors**
22: $k \leftarrow 2k$
23: **sinon si** $k > 1$ **alors**
24: $k \leftarrow k/2$
25: **pour** $i \in [0, \delta]$ **faire**
26: **si** $c_{r_i} =$ VRAI **alors**
27: $r_i \leftarrow 2r_i$
28: **sinon si** $r_i > 1$ **alors**
29: $r_i \leftarrow r_i/2$
30: **renvoyer** (f, g)

3.3.2 Utilisation de translations avant la descente locale

Une première façon de générer plusieurs points de départs pour l'algorithme de descente locale est de l'appeler pour plusieurs paires de polynômes de la forme $(f(X+k_j), g(X+k_j))$, pour plusieurs entiers k_j. Pour choisir ces entiers, nous allons considérer les polynômes $\tilde{a}_i(k)$, pour $0 \leqslant i \leqslant d$, définis comme les coefficients du polynôme $f(X+k)$, c'est-à-dire :

$$f(X+k) = \sum_{i=0}^{d} \tilde{a}_i(k) X^i. \tag{3.16}$$

Ces polynômes sont des polynômes à coefficients entiers en la variable k et $\deg(\tilde{a}_i(k)) = d-i$. En particulier, $\tilde{a}_{d-3}(k) = (d(d-1)(d-2)/6)a_d k^3 + ((d-1)(d-2)/2)a_{d-1}k^2 + (d-2)a_{d-2}k + a_{d-3}$ est un polynôme de degré 3 en k et admet donc toujours au moins une racine réelle. Une approximation entière des racines de $\tilde{a}_{d-3}$ peut être utilisée pour trouver des translations qui minimisent le coefficient de degré $d-3$ de f, qui est le premier coefficient dont la taille n'est pas contrôlée par les algorithmes de Kleinjung. Cette idée est résumée dans l'algorithme 3.5.

Algorithme 3.5 Algorithme de descente locale avec translations

Entrée : Deux polynômes f et g tels que $\deg g = 1$ et $\deg f > 1$.
Sortie : Deux polynômes f_{opt} et g_{opt} tels que $\deg g_{\text{opt}} = 1$, $\deg f_{\text{opt}} = \deg f$, $\|f_{\text{opt}}\|_2 \leqslant \|f\|_2$ et $\text{Res}(f_{\text{opt}}, g_{\text{opt}}) = \text{Res}(f, g)$

1: $\mathcal{L} \leftarrow \varnothing$
2: Ajouter à $\mathcal{L}$ la paire de polynômes renvoyée par l'algorithme 3.4 appliqué à (f, g)
3: **pour** r racine réelle de $\tilde{a}_{d-3}(k)$ **faire**
4: Calculer $k_0 \in \{ \lceil r \rceil, \lfloor r \rfloor \}$ qui minimise $\tilde{a}_{d-3}(k)$
5: $(f_{k_0}, g_{k_0}) \leftarrow (f(X+k_0), g(X+k_0))$
6: Ajouter à $\mathcal{L}$ la paire de polynômes renvoyée par l'algorithme 3.4 appliqué à (f_{k_0}, g_{k_0})
7: **renvoyer** la paire $(f_{opt}, g_{opt}) \in \mathcal{L}$ dont la norme du polynôme algébrique est minimale

3.3.3 Utilisation de l'algorithme LLL avant la descente locale

Les algorithmes d'optimisation de la taille décrits précédemment renvoient uniquement des paires de polynômes tels que le résultant des deux polynômes est égal au résultant des deux polynômes de la paire donnée en entrée. Or, il est possible que des paires de polynômes dont le résultant est un multiple du résultant de la paire d'entrée aient un polynôme algébrique avec un norme plus petite. Le problème de l'optimisation de la taille donnée par l'équation (3.8) se réécrit donc de la façon suivante :

$$\underset{c \in \mathbb{Z}\setminus\{0\}, k \in \mathbb{Z}, r \in \mathbb{Z}_\delta[X], s>0}{\text{argmin}} \left\| c\tilde{f}_{2,k} + r\tilde{f}_{1,k} \right\|_{2,s}. \tag{3.17}$$

Pour trouver de telles paires de polynômes l'algorithme LLL [93, 117] peut être utilisé. Après avoir fixé une valeur pour la skewness s, nous pouvons appliquer l'algorithme LLL au vecteur $\left(a_0, a_1 s, \ldots, a_d s^d\right)$, correspondant au polynôme f, et aux vecteurs $\left(\ldots, 0, -m_1 s^i, m_2 s^{i+1}, 0, \ldots\right)$, correspondant aux translations $X^i g$, pour $0 \leqslant i \leqslant \delta$. Les vecteurs renvoyés par l'algorithme LLL seront de la forme $\left(v_0, v_1 s, \ldots, v_d s^d\right)$ et le polynôme $f_{LLL} = \sum_{i=0}^{d} v_i X^i$ est un polynôme de la forme $cf + (r_\delta X^\delta + \cdots + r_0)g$. Pour augmenter nos chances de trouver un polynôme avec une norme petite, l'algorithme LLL peut être appelé avec différentes valeurs pour la skewness s.

L'utilisation de l'algorithme LLL nous permet donc de construire des polynômes de la forme $cf + (r_\delta X^\delta + \cdots + r_0)g$ dont le résultant avec g vaut $c\,\mathrm{Res}(f,g)$. Les algorithmes de Kleinjung génèrent des paires de polynômes (f,g) tels que $\mathrm{Res}(f,g) = N$ et donc, après utilisation de l'algorithme LLL, nous obtiendrons des paires de polynômes dont le résultant est cN. L'entier c est appelé le *multiplicateur*, pour garder la terminologie de la sélection polynomiale de l'algorithme MPQS [139]. Les travaux précédents n'autorisaient qu'un multiplicateur égal à 1. En utilisant l'algorithme LLL nous autorisons tout multiplicateur non nul, ce qui permet d'obtenir de meilleures paires de polynômes comme le montre la comparaison de la figure 3.1.

3.3.4 Minimiser simultanément les coefficients de degré $d-2$ et $d-3$

Dans cette section, nous allons décrire une nouvelle méthode pour trouver des translations qui fourniront des polynômes qui pourront être utilisés comme entrées de l'algorithme décrit dans la section 3.3.3. Cette méthode est une amélioration de la méthode décrite dans la section 3.3.2 où nous voulons trouver des translations k qui minimisent simultanément les coefficients de degré $d-2$ et $d-3$ de f.

Si (f,g) est une paire de polynômes dont la taille doit être optimisée, après avoir appliqué une translation k et l'algorithme LLL comme décrit dans la section 3.3.3, et si seulement la rotation de plus haut degré est considérée, le polynôme obtenu sera de la forme $cf(X+k)+r_{d-3}X^{d-3}g(X+k)$, où c et r_{d-3} sont des entiers. En notant $q = r_{d-3}/c$, le problème revient à trouver des rationnels q pour lesquels il existe des entiers k qui rendent $\hat{a}_{d-2}(k)$ et $\hat{a}_{d-3}(k)$ petits, où $\hat{a}_i(k)$ est le coefficient de degré i de $\hat{f}(X) = f(X+k) + qX^{d-3}g(X+k)$. Nous allons maintenant essayer de minimiser simultanément $\hat{a}_{d-2}(k)$ et $\hat{a}_{d-3}(k)$ en cherchant des rationnels q pour lesquels $\mathrm{Res}_k(\hat{a}_{d-3}(k), \hat{a}_{d-2}(k))$ a une racine ou est petit en valeur absolue.

Pour les polynômes de degré $d = 6$, $\hat{a}_{d-2}(k)$ et $\hat{a}_{d-3}(k)$ ont la forme suivante :

$$\hat{a}_4(k) = 15a_6k^2 + 5a_5k + a_4 + qm_2;$$
$$\hat{a}_3(k) = 20a_6k^3 + 10a_5k^2 + 4a_4k + qm_2k + a_3 - qm_1.$$

Le résultant $\mathrm{Res}(\hat{a}_3(k), \hat{a}_4(k))$ est un polynôme de degré 3 en q et à coefficients entiers. Il admet donc une ou trois racines réelles. Dans le cas où le résultant a trois racines réelles, chaque racine peut être approchée par un nombre rationnel en utilisant les fractions continues, les approximations de Farey ou en essayant toutes les fractions rationnelles avec un dénominateur borné. En remplaçant q par les approximations rationnelles obtenues dans $\hat{a}_4(k)$ et $\hat{a}_3(k)$, nous obtenons des polynômes en k dont nous pouvons calculer les racines (elles sont proches par choix de q) qui, une fois arrondies, permettent d'obtenir des valeurs possibles pour la translation k. Dans le cas où le résultant n'a qu'une seule racine réelle, les expériences montrent qu'il est nécessaire de considérer aussi les extrema du résultant et pas seulement la racine. Alors, comme dans le cas précédent, des approximations rationnelles de la racine et des extrema peuvent être utilisées pour trouver des valeurs pour la translation k.

Pour d'autres valeurs du degré d, une méthode similaire peut être utilisée. En effet, pour tout degré d autre que 5, le résultant $\mathrm{Res}(\hat{a}_{d-2}(k), \hat{a}_{d-3}(k))$ est un polynôme à coefficients entiers de degré 3 en q et la méthode décrite ci-dessus peut s'appliquer. Dans le cas où $d = 5$, le résultant $\mathrm{Res}(\hat{a}_{d-2}(k), \hat{a}_{d-3}(k))$ est un polynôme à coefficients entiers de degré 2 en q et il est nécessaire de considérer les racines et les extrema de ce polynôme. Dans tous les cas, les valeurs de k trouvées par cette méthode fournissent des paires de polynômes $(f(X+k), g(X+k))$ qui peuvent être utilisées comme entrées de l'algorithme LLL, qui devrait renvoyer une paire de polynômes telle que le multiplicateur c et la rotation r_{d-3} sont tels que r_{d-3}/c devrait être proche de la valeur de q qui a permis de trouver la translation k.

L'utilisation de cette nouvelle méthode et l'utilisation de l'algorithme LLL comme décrit dans la section 3.3.3 sont détaillées dans l'algorithme 3.6.

Algorithme 3.6 Algorithme d'optimisation utilisant LLL avec translations

Entrée : Deux polynômes f et g tels que $\deg g = 1$ et $\deg f > 1$ et $\mathcal{S}$ un ensemble de *skewness* pour la norme de l'algorithme LLL.

Sortie : Deux polynômes f_{opt} et g_{opt} tels que $\deg g_{\text{opt}} = 1$, $\deg f_{\text{opt}} = \deg f$, $\|f_{\text{opt}}\|_2 \leqslant \|f\|_2$ et $\text{Res}(f_{\text{opt}}, g_{\text{opt}}) = c\,\text{Res}(f, g)$ pour un entier c non nul

$\mathcal{K} \leftarrow \varnothing$, $\mathcal{L} \leftarrow \varnothing$
pour $q_{\mathbb{R}}$ racines et extrema de $\text{Res}(\hat{a}_{d-2}(k), \hat{a}_{d-3}(k))$ **faire**
  pour $q_{\mathbb{Q}}$ approximations rationnelles de $q_{\mathbb{R}}$ **faire**
    pour $k_{\mathbb{R}}$ racines de $\hat{a}_{d-2}(k)$ ou de $\hat{a}_{d-3}(k)$ où q est remplacé par $q_{\mathbb{Q}}$ **faire**
      Ajouter $\lfloor k_{\mathbb{R}} \rceil$ à $\mathcal{K}$
pour k dans $\mathcal{K}$ **faire**
  $(f_k, g_k) \leftarrow (f(X+k), g(X+k))$
  pour s dans $\mathcal{S}$ **faire**
    $\mathbf{v_f} \leftarrow \left(a_0, a_1 s, \ldots, a_d s^d\right)$ où $f_k = \sum_{i=0}^{d} a_i X^i$
    $\mathbf{v_i} \leftarrow \left(\ldots, 0, -m_1 s^i, m_2 s^{i+1}, 0, \ldots\right)$, pour $0 \leqslant i \leqslant \delta$, où $g_k = m_2 X - m_1$
    $\mathcal{B} \leftarrow$ les vecteurs renvoyés par l'algorithme LLL appliqué aux vecteurs $\mathbf{v_f}, \mathbf{v_0}, \ldots, \mathbf{v}_\delta$
    pour $\mathbf{v} \in \mathcal{B}$ tel que le dernier coefficient est non nul **faire**
      $f_{LLL} \leftarrow \sum_{i=0}^{d} v_i X^i$, où $\mathbf{v} = \left(v_0, v_1 s, \ldots, v_d s^d\right)$
      Ajouter à $\mathcal{L}$ la paire renvoyée par l'algorithme 3.4 appliqué à (f_{LLL}, g_k)
renvoyer la paire $(f_{opt}, g_{opt}) \in \mathcal{L}$ dont la norme du polynôme algébrique est minimale

3.3.5 Résultats expérimentaux

Dans cette section, nous donnons des résultats expérimentaux des nouvelles méthodes décrites dans les sections précédentes sur différents nombres du challenge RSA [87] : RSA-155, RSA-768, RSA-896 et RSA-1024. À part pour RSA-155, le degré du polynôme algébrique est 6. Pour RSA-155, le degré du polynôme algébrique est 5.

Comparaisons

La première comparaison a été faite pour RSA-768 sur un ensemble de 10^5 paires de polynômes non optimisés. Ces polynômes ont été générés par `cado-nfs` [33] et `Msieve` [111] qui implémentent les algorithmes de Kleinjung décrits dans la section 3.2. La table 3.1 compare la valeur moyenne et l'écart type de la log-norme pour les différents algorithmes décrits précédemment. La figure 3.1 montre la distribution de la log-norme des paires de polynômes non optimisés et optimisés.

RSA-768	Polynômes non optimisés	Algorithme 3.5	Algorithme 3.6
Log-norme moyenne	80,75	69,84	68,42
Écart type de la log-norme	1,00	0,56	0,72

TABLE 3.1 – Comparaison des méthodes d'optimisation de la taille pour 10^5 paires de polynômes pour RSA-768.

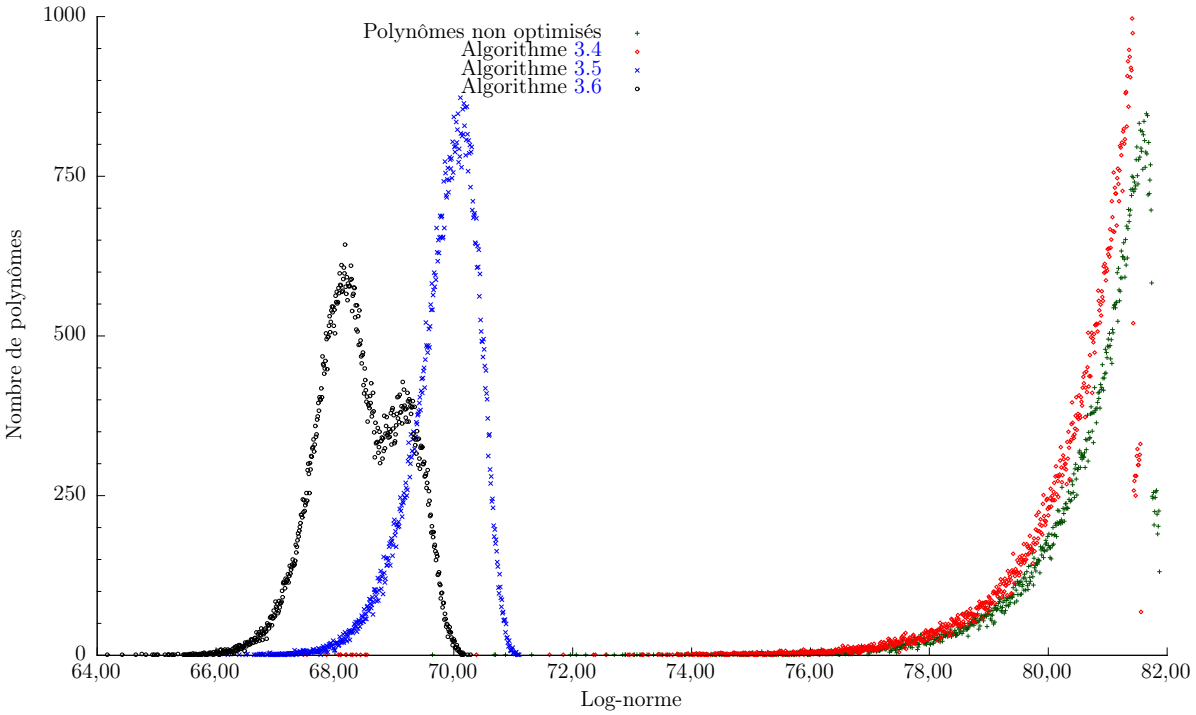

FIGURE 3.1 – Répartition des polynômes avant et après l'optimisation de la taille pour 10^5 paires de polynômes pour RSA-768.

Une comparaison similaire a été faite pour RSA-155 sur un ensemble de 5795 paires de polynômes non optimisés produit par `cado-nfs`. Bien que le gain sur la valeur moyenne de la log-norme soit plus faible, cela montre bien que les nouveaux algorithmes sont aussi plus performants pour le degré 5.

RSA-155	Polynômes non optimisés	Algorithme 3.5	Algorithme 3.6
Log-norme moyenne	53,54	50,43	49,36
Écart type de la log-norme	1,96	1,49	0,62

TABLE 3.2 – Comparaison des méthodes d'optimisation de la taille pour 5795 paires de polynômes pour RSA-155.

Pour RSA-768, nous avons effectué une comparaison supplémentaire sur les paires de polynômes obtenues après l'optimisation de la taille et l'optimisation des propriétés des racines. La table 3.3 contient la moyenne de la log-norme, de la grandeur α et de la valeur E de Murphy pour les 10^5 paires de polynômes après l'optimisation de la taille et l'optimisation des propriétés des racines. En supposant que la valeur E de Murphy est une estimation précise du rendement de la paire de polynômes lors de l'étape de crible, alors nous pouvons voir que les nouvelles méthodes d'optimisation de la taille produisent des paires de polynômes avec un meilleur rendement en moyenne.

RSA-768	Algorithme 3.5	Algorithme 3.6
Log-norme moyenne	71,86	69,90
Grandeur α moyenne	−7,019	−6,812
Valeur E de Murphy moyenne	$8{,}60 \times 10^{-14}$	$1{,}10 \times 10^{-13}$
Valeur E de Murphy moyenne (pour les 100 meilleures)	$2{,}14 \times 10^{-13}$	$2{,}53 \times 10^{-13}$

TABLE 3.3 – Comparaison de deux méthodes d'optimisation de la taille sur 10^5 paires de polynômes non optimisés pour RSA-768 après optimisation de la taille et optimisation des propriétés des racines. Pour le calcul de la valeur E de Murphy nous avons utilisé $B_1 = 1{,}1 \times 10^9$, $B_2 = 2{,}0 \times 10^8$ et une aire de $2{,}362 \times 10^{18}$ pour la zone de crible Ω.

Meilleurs polynômes pour RSA-768

Nous avons utilisé l'algorithme 3.6 sur un ensemble de paires de polynômes pour RSA-768 fourni par Jason Papadopoulos et nous avons trouvé, après optimisation de la taille et optimisation des propriétés des racines, plusieurs paires de polynômes meilleures que celle utilisée pour le calcul de la factorisation de RSA-768, dont deux sont données dans la table 3.7 dans l'annexe à la fin du chapitre.

Ces deux paires de polynômes, appelées A_{768} et B_{768} correspondent aux multiplicateurs $c = 36$ et $c = 15$ respectivement. Elles n'auraient donc pas pu être trouvées avec les anciennes méthodes d'optimisation de la taille. La paire de polynôme utilisée pour la factorisation de RSA-768 a un polynôme algébrique dont la log-norme est 64,08 et sa valeur E de Murphy est $4{,}28 \times 10^{-13}$. Les deux paires de polynômes A_{768} et B_{768} ont de meilleures valeurs E de Murphy, $4{,}42 \times 10^{-13}$ et $4{,}52 \times 10^{-13}$ respectivement. Pour le calcul de la valeur E de Murphy, nous avons utilisé $B_1 = 1{,}1 \times 10^9$, $B_2 = 2{,}0 \times 10^8$ et une aire de $2{,}362 \times 10^{18}$ pour la zone de crible.

Une comparaison sur le rendement de l'étape de crible a été faite entre la paire de polynômes utilisée pour la factorisation de RSA-768 et les deux paires de polynômes A_{768} et B_{768} en utilisant le même binaire que celui utilisé pour la factorisation de RSA-768 avec les mêmes paramètres et pour toutes les valeurs possible de spécial-$\mathfrak{q}$ dans l'intervalle $[3\,400\,000\,000, 3\,400\,100\,000]$. La paire de polynômes A_{768} produit environ 7 % de relations supplémentaires et 5 % de relations supplémentaires par seconde par rapport à la paire de polynômes utilisée pour la factorisation de RSA-768. La paire de polynômes B_{768} produit environ 5 % de relations supplémentaires et 3 % de relations supplémentaires par seconde par rapport à la paire de polynômes utilisée pour la factorisation de RSA-768.

Meilleurs polynômes pour RSA-896

Pour RSA-896, nous avons considéré l'ensemble des 10 paires de polynômes non optimisés construites en utilisant l'algorithme 3.1 avec les valeurs de a_6, m_1 et m_2 données dans la table 3.5 dans l'annexe à la fin du chapitre. La table 3.4 contient, pour ces 10 paires de polynômes, la log-norme du polynôme algébrique avant optimisation et après optimisation en utilisant les algorithmes 3.5 et 3.6.

L'algorithme 3.6 produit des polynômes dont la log-norme est plus petite de 2,40 en moyenne (79,94 au lieu de 82,34) et, sauf pour le polynôme #8, toujours plus petite que celle des polynômes produits par l'algorithme 3.5.

#	1	2	3	4	5	6	7	8	9	10
Polynômes non optimisés	98,28	98,11	96,89	98,00	97,84	98,53	97,18	98,37	96,97	96,63
Algorithme 3.5	82,88	82,74	82,30	82,03	82,37	83,33	82,12	79,36	83,79	82,45
Algorithme 3.6	80,53	80,16	79,33	79,75	79,78	79,83	80,04	80,72	79,92	79,38

TABLE 3.4 – Log-norme du polynôme algébrique pour 10 paires de polynômes pour RSA-896.

Meilleurs polynômes pour RSA-1024

Pour RSA-1024, nous avons tout d'abord considéré la paire de polynômes avec le polynôme algébrique de degré 6 donnée dans [96, appendice A]. Le polynôme algébrique de cette paire a une log-norme de 100,02. Nous avons utilisé l'algorithme 3.6 pour optimiser de nouveau cette paire de polynômes et nous avons obtenu la paire de polynômes A_{1024}, donnée dans la table 3.6 dans l'annexe à la fin du chapitre, dont la log-norme du polynôme algébrique est 94,91. Ceci représente un gain d'un facteur environ $\exp(100{,}02 - 94{,}91) \approx 166$ sur la norme du côté algébrique.

De plus, nous avons aussi utilisé notre implémentation de l'algorithme 3.6 dans `cado-nfs` pour trouver une meilleure paire de polynômes, appelée B_{1024} et donnée dans la table 3.6 dans l'annexe à la fin du chapitre. Cette paire correspond à un multiplicateur $c = 5$. En se basant sur sa valeur E de Murphy de $8{,}86 \times 10^{-12}$ contre $9{,}75 \times 10^{-13}$ pour la paire de polynômes de [96], nous pouvons estimer que le rendement lors de l'étape de crible serait meilleur d'un facteur 9,1. En utilisant les mêmes paramètres ($B_1 = B_2 = 10^{11}$ et une aire de 10^{18}), nous obtenons une valeur E de Murphy de $3{,}56 \times 10^{-9}$ pour la paire de polynômes utilisée pour la factorisation de RSA-768, ce qui nous permet d'estimer le temps de crible pour RSA-1024 comme 402 fois le temps de crible pour la factorisation de RSA-768 (au lieu d'un facteur 1000 comme annoncé dans [83]). La paire de polynômes B_{1024} est aussi meilleure que la paire donnée dans [78], dont la valeur E de Murphy est $6{,}79 \times 10^{-12}$. Enfin, remarquons que cette paire de polynômes a été trouvée après l'équivalent d'environ 1000 heures de calcul sur un processeur, et que nous nous attendons donc à pouvoir trouver une bien meilleure paire si un vrai effort de calcul de l'équivalent de quelques milliers d'années sur un processeur était entrepris.

3.4 La sélection polynomiale non linéaire

Dans cette section, nous allons étudier le cas particulier de la sélection polynomiale non linéaire où les deux polynômes ont le même degré. Ce degré sera noté $d = d_1 = d_2$ dans la suite. Dans cette section, nous utiliserons une norme différente de celle définie précédemment, car tous les résultats que nous utiliserons sont donnés pour cette nouvelle norme. Cependant, par équivalence des normes en dimension finie, les résultats devraient être identiques, à une constante dépendant du degré près, pour la norme définie par l'équation (3.4). Pour un polynôme $f = \sum_{i=0}^{d} a_i X^i$ de degré d, la norme avec *skewness* est définie par

$$\|f\|_{2,s} = \sqrt{\sum_{i=0}^{d} \left(a_i s^{i-\frac{d}{2}}\right)^2} \tag{3.18}$$

et, pour un vecteur $\mathbf{c} = (c_0, \ldots, c_d)$ de longueur $d+1$, la norme avec *skewness* est définie par

$$\|\mathbf{c}\|_{2,s} = \sqrt{\sum_{i=0}^{d} \left(c_i s^{\frac{d}{2}-i}\right)^2}. \tag{3.19}$$

La borne sur la norme des polynômes d'une paire adaptée à N donnée dans [43, corollaire 2.3] montre qu'une paire de deux polynômes de degré d de taille optimale est telle que le produit des normes $\|f_1\|_2\|f_2\|_2$ est en $\mathrm{O}(N^{1/d})$. L'outil principal utilisé pour la sélection polynomiale non linéaire est la progression géométrique, qui sera étudiée dans la section 3.4.1. Ensuite, dans la section suivante, nous montrerons comment construire des progressions géométriques pour générer une paire de polynômes de degré $d = 3$.

3.4.1 Progressions géométriques

Définition 3.4.1 (Progression géométrique)**.** Soient N et k des entiers strictement positifs et m un entier. Une progression géométrique de longueur k et de raison m modulo N est un vecteur $\mathbf{c} = (c_0, \ldots, c_{k-1}) \in \mathbb{Z}^k$ tel que, pour $0 \leqslant i \leqslant k-1$, $c_i \equiv c_0 m^i \pmod{N}$.

Dans la suite, lorsque nous parlerons de deux vecteurs orthogonaux, ce sera toujours par rapport au produit scalaire canonique, c'est-à-dire que si $\mathbf{a}$ et $\mathbf{b}$ sont deux vecteurs de $\mathbb{Z}^k$, pour un entier k strictement positif, $\mathbf{a}$ et $\mathbf{b}$ seront dit orthogonaux si et seulement si $\sum_{i=0}^{k-1} a_i b_i$ est nul dans $\mathbb{Z}$.

Si $\mathbf{c} = (c_0, \ldots, c_d)$ est une progression géométrique de longueur $d+1$ et de raison m et $f = \sum_{i=0}^{d} a_i X^i$ est un polynôme de degré d à coefficients entiers dont le vecteur des coefficients est orthogonal à $\mathbf{c}$, c'est-à-dire tel que $\sum_{i=0}^{d} a_i c_i = 0$, alors le polynôme f admet m comme racine modulo N. Si $\mathbf{c}$ est aussi une progression géométrique sur $\mathbb{Q}$, alors le polynôme f admet une racine rationnelle et n'est donc pas irréductible. Dans la suite, nous construirons des progressions géométriques qui ne seront pas des progressions géométriques sur $\mathbb{Q}$. À partir d'une progression géométrique de raison m modulo N et de longueur $d+k$, où k est un entier strictement positif, il est possible de construire k progressions géométriques de longueur $d+1$ et de raison m modulo N en considérant les k façons possibles d'extraire $d+1$ coefficients consécutifs d'un vecteur de longueur $d+k$.

En généralisant une méthode de Montgomery, Coxon a montré dans [44, théorème 5.1] que le problème de trouver deux polynômes de degré d premiers entre eux sur $\mathbb{Q}$, ayant une racine commune modulo N et dont le produit des normes est en $\mathrm{O}(N^{1/d})$ peut se ramener à trouver une progression géométrique de longueur $2d-1$ dont la norme est en $\mathrm{O}(N^{1-1/d})$ et dont les $d-1$ sous-progressions géométriques de longueur $d+1$ ne sont pas des progressions géométriques sur $\mathbb{Q}$.

Exemple 3.4.2 (Construction de Montgomery pour $d = 2$)**.** D'après le paragraphe précédent, pour construire deux polynômes de degré 2 ayant une racine commune modulo N et de norme optimale, il suffit de construire une progression géométrique de longueur 3 de norme en $\mathrm{O}(N^{1/2})$. Montgomery [105] a proposé de considérer la progression géométrique suivante :

$$\mathbf{c} = (c_0, c_1, c_2) = \left(a_2 m_2, a_2 m_1, \frac{a_2 m_1^2 - kN}{m_2}\right),$$

où k, a_2, m_1 et m_2 sont des entiers non nuls tels que a_2, m_1 et m_2 sont positifs, m_1 et m_2 sont premiers entre eux, $\mathrm{pgcd}(a_2 m_2, N) = 1$ et $a_2 m_1^2 \equiv kN \pmod{m_2}$. Alors, $\mathbf{c}$ est une progression géométrique de raison m_1/m_2 modulo N et, si $a_2 m_2$ est en $\mathrm{O}(N^{1/2})$ et $0 \leqslant m_1 - \sqrt{(kN)/a_2} \leqslant m_2$, sa norme est en $\mathrm{O}(N^{1/2})$. L'ensemble des vecteurs à coefficients entiers orthogonaux à $\mathbf{c}$ est engendré par $(m_1, -m_2, 0)$ et $((m_1 t - c_2)/m_2, -t, a_2)$, où $t \in [-m_2/2, m_2/2[$ vérifie $t \equiv c_2/m_1 \pmod{m_2}$. La paire de polynômes de degré 2 est obtenue en appliquant l'algorithme LLL à ces deux vecteurs et en considérant les deux vecteurs les plus courts produits par LLL dont la dernière coordonnée est non nulle.

3.4.2 Progressions géométriques de longueur 5 et polynômes de degré 3

Plusieurs constructions ont été proposées pour générer une paire de deux polynômes de degré 3. Williams [147] puis Prest et Zimmermann [128] ont proposé des méthodes pour construire des progressions géométriques de longueur 4, ce qui produit des polynômes dont la norme n'est pas optimale. Koo, Jo et Kwon [84] puis Coxon [43] ont proposé des méthodes pour construire des progressions géométriques de longueur 5. La forme la plus générale de progression géométrique, décrite par Coxon, est

$$\mathbf{c} = (c_0, c_1, c_2, c_3, c_4) = \left(a_3 m_2^2, a_3 m_2 m_1, a_3 m_1^2, \frac{a_3 m_1^3 - kN}{m_2}, \frac{m_1(a_3 m_1^3 - kN)}{m_2^2}\right), \tag{3.20}$$

où k, a_3, m_1 et m_2 sont des entiers non nuls tels que a_3, m_1 et m_2 sont positifs, $\gcd(m_1, m_2) = 1$, $\gcd(a_3 m_2, N) = 1$ et $a_3 m_1^3 \equiv kN \pmod{m_2^2}$. Coxon a montré que si les conditions suivantes sont satisfaites :

$$0 \leqslant m_1 - \sqrt[3]{\frac{kN}{a_3}} \leqslant \frac{m_2 s}{3}, \quad m_2 s \leqslant m_1, \quad m_2 = \Theta(\frac{m_1}{s}), \quad k = \mathrm{O}(1), \quad a_3 = \mathrm{O}(1) \text{ et } s = \frac{1}{\sqrt[4]{6}}\sqrt{\frac{m_2}{\bar{a}_3}} \tag{3.21}$$

où s est la *skewness* pour laquelle la norme sera minimum et $\bar{a}_3 = a_3/\gcd(a_3, c_3/m_2)$, alors la norme d'une progression géométrique de la forme (3.20) est en $\mathrm{O}(N^{2/3})$ et donc, d'après le lien entre progressions géométriques et polynômes, cela montre que nous pouvons construire une paire de polynômes dont le produit des normes est en $\mathrm{O}(N^{1/3})$. La façon de calculer les deux polynômes à partir de la progression géométrique est détaillée dans [43, algorithme 5.2].

Cependant, aucune méthode efficace pour générer de telles progressions géométriques n'est proposée ni dans [84] ni dans [43]. Les seules méthodes proposées ne fonctionnent que pour les petits entiers. Nous allons montrer comment la variante du second algorithme de Kleinjung avec spécial-q, c'est-à-dire l'algorithme 3.3, peut être utilisée pour générer de telles progressions géométriques. En effet, en le modifiant, cet algorithme peut être utilisé pour trouver deux entiers m_1 et m_2 premiers entre eux tels que $a_3 m_1^3 \equiv kN \pmod{m_2^2}$. Il suffit pour cela de remplacer la définition de $\tilde{N}$ par $\tilde{N} \leftarrow N/a_3$ et le corps de la dernière boucle par $m_1 \leftarrow m_0 + r_q + iq^2$ et $m_2 \leftarrow qp_1p_2$. L'entier m_2 produit par cet algorithme sera de la forme qp_1p_2, avec q un entier positif non nul et p_1 et p_2 deux nombres entiers de l'intervalle $[P, 2P]$, où P est un paramètre de l'algorithme. L'algorithme peut aussi être modifié pour ne considérer que les spécial-q d'une certaine taille. Calculons les tailles de P et q pour que l'algorithme produise des entiers m_1 et m_2 qui satisfassent les conditions (3.21). L'entier m_2 sera compris dans l'intervalle $[qP^2, 4qP^2]$, ce qui permet d'obtenir l'encadrement suivante pour $m_2 s$:

$$\frac{P^3 q^{\frac{3}{2}}}{\sqrt[4]{6}\sqrt{\bar{a}_3}} \leqslant m_2 s \leqslant \frac{8P^3 q^{\frac{3}{2}}}{\sqrt[4]{6}\sqrt{\bar{a}_3}}. \tag{3.22}$$

Comme l'entier m_1 satisfait $0 \leqslant m_1 - \sqrt[3]{\frac{kN}{a_3}} \leqslant 4q^2P^2$ (voir section 3.2.3), en imposant que le terme de gauche soit supérieur à $12q^2P^2$, la première inégalité des conditions (3.21) est satisfaite et nous obtenons :

$$q \leqslant \frac{P^2}{144\sqrt{6}\bar{a}_3} \quad \text{et} \quad m_2 s \leqslant \frac{8P^3 q^{\frac{3}{2}}}{\sqrt[4]{6}\sqrt{\bar{a}_3}} \leqslant \frac{P^6}{1296\bar{a}_3^2}. \tag{3.23}$$

De la même façon, en imposant que le terme de droite soit inférieur à m_1, la deuxième inégalité des conditions (3.21) est satisfaite et nous obtenons, en notant $\bar{m} = \sqrt[3]{kN/a_3}$:

$$P^6 \leqslant 1296\bar{a}_3^2 m_1 \leqslant 1944\bar{a}_3^2\bar{m}. \tag{3.24}$$

Pour produire des progressions géométriques dont la taille convient, il faut donc choisir une valeur de P qui satisfait l'inégalité (3.24) ci-dessus et ne considérer que les spécial-q qui satisfont l'inégalité (3.23).

Exemple 3.4.3. Pour cet exemple, nous avons appliqué les idées décrites ci-dessus à l'entier de 91 chiffres suivant :

$$N = 456717603989410870435875216065562819203492730696982839773907434662898832715547522284379339 3,$$

avec $a_3 = 1$ et $k = 1$. Nous avons considéré le plus grand entier P qui satisfait l'inégalité (3.24), $P = 384383$, et des spécial-q de 30 bits environ. Nous avons obtenu les deux polynômes suivants :

$$f_1 = 5X^3 + 73057935204528725963X - 183372459241923986859589974901,$$
$$f_2 = 9X^3 - 5513732206198062614X + 71897700272244162598363685 65,$$

qui sont irréductibles, premiers entre eux sur $\mathbb{Q}$ et ont m_1/m_2 comme racine commune modulo N où

$$m_1 = 168630098331343796303549161 6934,$$
$$m_2 = qp_1p_2 = (11 \times 61 \times 149 \times 157 \times 181) \times 468913 \times 514243 = 685090077871748846737.$$

Le résultant de f_1 et f_2 est N et le produit des normes $\|f_1\|_{2,s}\|f_2\|_{2,s}$ vaut environ $N^{0,334}$ pour $s = 1\,499\,816\,976$.

Les polynômes obtenus en utilisant des progressions géométriques de la forme de l'équation (3.20) ont nécessairement le coefficient de degré 2 nul. Pour générer des polynômes dont le coefficient de degré 2 est non nul, nous proposons de construire deux progressions géométriques de longueur 4 avec la même raison, qui ne sont pas issues d'une même progression géométrique de longueur 5. Un exemple de telles progressions géométriques est donné par la construction suivante :

$$\mathbf{c_0} = \left(a_3m_2^2, a_3m_2m_1, a_3m_1^2, \frac{a_3m_1^3 - kN}{m_2}\right),$$
$$\mathbf{c_1} = \left(a_3m_1m_2 + a_2m_2^2, a_3m_1^2 + a_2m_1m_2, \frac{a_3m_1^3 + a_2m_1^2m_2 - kN}{m_2}, m_1\frac{a_3m_1^3 + a_2m_1^2m_2 - kN}{m_2^2}\right),$$

où k, a_3, a_2, m_1 et m_2 sont des entiers non nuls tels que a_3, m_1 et m_2 sont positifs, m_1 et m_2 sont premiers entre eux et $a_3m_1^3 + a_2m_1^2m_2 \equiv kN \pmod{m_2^2}$. Cette nouvelle construction permet d'obtenir une paire de polynômes de degré 3 de la même façon qu'avec deux progressions géométriques issues d'une même progression géométrique de longueur 5. De plus de telles progressions géométriques peuvent être générées en utilisant l'algorithme 3.3 sans modifications.

Exemple 3.4.4. En reprenant le même entier N que celui de l'exemple 3.4.3, ainsi que les mêmes valeurs pour a_3, k et P, nous avons obtenu les deux polynômes suivants :

$$f_1 = 4X^3 + 4X^2 + 5995146827842608807 5X - 270727067116717048548109923948,$$
$$f_2 = 5X^3 + 5X^2 - 2088383365843202374 9X + 80123467711030200647676178655,$$

qui sont irréductibles, premiers entre eux sur $\mathbb{Q}$ et ont m_1/m_2 comme racine commune modulo N où

$$m_1 = 167412920642770604533125433436 0,$$
$$m_2 = qp_1p_2 = (19 \times 23 \times 109 \times 127 \times 157) \times 528053 \times 764261 = 383292676025858535371.$$

Le résultant de f_1 et f_2 est N et le produit des normes $\|f_1\|_{2,s}\|f_2\|_{2,s}$ vaut environ $N^{0,334}$ pour $s = 3\,234\,658\,763$.

Exemple 3.4.5. Pour cet exemple, nous avons considéré pour l'entier N le nombre RSA-155, avec $a_3 = 1$ et $k = 1$. Nous avons utilisé $P = 2^{30}$ et des spécial-q de 53 bits environ. Nous avons obtenu les deux polynômes suivants :

$$\begin{aligned} f_1 &= X^3 - 557195513354199790388947345372209 2X \\ &\quad + 155761015273249115785758263395415201268628767065420 2, \\ f_2 &= X^3 + 175916122069825586267710241522957 9X \\ &\quad - 665959042154314795478775389336924962644038269678973, \end{aligned}$$

qui sont irréductibles, premiers entre eux sur $\mathbb{Q}$ et ont m_1/m_2 comme racine commune modulo N où

$$\begin{aligned} m_1 &= 22235691948868059533363580232910769753303259403331 75, \\ m_2 &= qp_1p_2 = 73311163542402537665665758689516 71 \\ &= (37 \times 157 \times 191 \times 199 \times 223 \times 227 \times 241) \times 1369599503 \times 1987197077. \end{aligned}$$

Le résultant de f_1 et f_2 est $-N$ et le produit des normes $\|f_1\|_{2,s}\|f_2\|_{2,s}$ vaut environ $N^{0,333}$ pour $s = 100\,813\,074\,954\,426\,576$.

3.5 Conclusion et perspectives

Dans ce chapitre, nous avons présenté différents aspects de la sélection polynomiale pour l'algorithme NFS. Dans le cas de la sélection polynomiale linéaire, nous avons montré comment générer des paires de polynômes adaptées à un entier positif N et nous avons proposé des améliorations à l'algorithme d'optimisation de la taille. Dans le cas de la sélection polynomiale non linéaire, nous avons décrit une méthode pour produire des progressions géométriques de longueur 5 permettant de générer des paires de polynômes de degré 3. Cela permet de montrer que les algorithmes décrits jusqu'à présent pour la sélection polynomiale non linéaire peuvent être utilisés en pratique pour des grands entiers.

Les améliorations proposées pour le problème de l'optimisation de la taille ne permettent pas encore d'être sûr de trouver le minimum global. Pour cela, il faudrait regarder du côté des algorithmes d'optimisation globale. Par exemple, en utilisant la méthode des moments de Lasserre [90], le problème de l'optimisation de la taille peut se ramener à la résolution d'un problème d'optimisation SDP (*semidefinite programming*) avec des variables entières. Cependant, à notre connaissance il n'existe pas de solveur multiprécision pour résoudre des problèmes d'optimisation SDP avec des variables entières. Il existe des solveurs multiprécision pour des variables réelles, comme SDPA-GMP [134] par exemple, mais la solution d'un problème d'optimisation globale sur les réels peut être très éloignée de la solution pour le même problème sur les entiers (voir, par exemple, la discussion de la section 9.4 dans [28]).

Pour la sélection polynomiale non linéaire, il reste encore des cas à étudier. Concernant les paires de polynômes de même degré, la prochaine étape naturelle est la génération de paires de polynômes de degré 4, qui sont plus adaptées pour factoriser de très grands entiers, comme RSA-1024. Pour y parvenir, il faudrait construire des progressions géométriques de longueur 7 dont la norme est en $\mathrm{O}(N^{3/4})$. Enfin, il y aussi le cas des paires de polynômes de degrés différents. Pour cela, un travail est en cours avec N. Coxon sur les progressions géométriques permettant de générer des paires de polynômes (f_1, f_2) tels que $\deg f_1 = \deg f_2 + 1$.

Annexe

#	a_6	m_2	m_1
1	120	30598679948073727114694567	388409740367611516819003632552473419494959325
2	120	39584252255977153653238969	388409740360914711183938301673437684112903927
3	180	24122614381393211892378929	36302920888345037533594763993085265139635480 0
4	240	12720295719832774984 3027	34603373980760708268993210773289788832928667 2
5	300	205701125136203480631 4333	33340090751470079438193623545241468481416391 5
6	480	5403413584512371865850751	30828101522793436115065585124124690098215856 9
7	540	2017884993246970143225589	30228831523556724404048084094917851630760879 5
8	600	55808326686191457067	29702644113166639147887049434519093950215270 3
9	600	52318705091858802318954527	29702644113374832823658042126568041701934542 7
10	600	10352691606130810454808797 3	29702644113522571092179926003616226729306870 5

TABLE 3.5 – Données pour les expériences pour RSA-896.

	Paire de polynômes A_{1024} : log-norme = 94,91
m_2	1
m_1	62904286063558990272557233200273917221632886994 13
a_6	11735979892429214822 40
a_5	-43608157020293570037272873757855616
a_4	69195814034117398762503510474365754553 7
a_3	45051120216120873437095774813233011859735 19
a_2	-17304452519439643403755585110507512764935257500
a_1	-28313100773851304238101962712925551719741165867633
a_0	24996329564944807789602917136794373782308959799485325
	Paire de polynômes B_{1024} : log-norme = 91,90 et $E = 8{,}86 \times 10^{-12}$
m_2	23877076888820427604098421
m_1	33325633007552533075965065591785662545082049497 38
a_6	492999999999872400
a_5	19986130996295579328005858 00
a_4	14776348389733418096949161617663667
a_3	-17369563296702789247942467572798015432351 6
a_2	-58245139481832624147323198441400656783348781 8962
a_1	29609635772301623248273428019688928620985521680508271 56
a_0	-20364558899868538420816205898474403074641452593893682451540 65

TABLE 3.6 – Deux paires de polynômes pour RSA-1024.

	Paire de polynômes A_{768} : log-norme = 65,40 et $E = 4{,}42 \times 10^{-13}$
m_2	3653258925429788683931
m_1	154477669109765136712756724037850686 26
a_6	3258961776
a_5	288664131841057800
a_4	11030506237737074466307
a_3	-893188977600857037294644587163
a_2	943910582396304678841343366483141 51
a_1	377093715995883343269663077625960978403307
a_0	-904516168995007172662900583473883296530585 4530
	Paire de polynômes non optimisée pour A_{768} : log-norme = 81,82
m_2	3653258925429788683931
m_1	154477669649080444719058876631998545 37
a_6	90526716
a_5	241012074
a_4	-297351230165464732635680
a_3	729479045157502847705046405886586876 4
a_2	283452995840471562081987321376267536 5
a_1	288524965019008859802888800564521145 3
a_0	651895590880755556906420587188754888 3
	Paire de polynômes B_{768} : log-norme = 64,39 et $E = 4{,}52 \times 10^{-13}$
m_2	5924452599136152496277
m_1	3057113257792712360104874467239834068 6
a_6	22604400
a_5	33946122310442580
a_4	74850428174211171973801
a_3	-25318719430823718613440606474 2
a_2	-81310927091457333751052543066797504
a_1	28772541579434794385398996592471406483945 8
a_0	-81187709244683044191049418357655772035310114 65
	Paire de polynômes non optimisée pour B_{768} : log-norme = 82,44
m_2	5924452599136152496277
m_1	3057113406076669441919073839597843614 8
a_6	1506960
a_5	-2418388
a_4	-1408315672283641690514244
a_3	-1426558909376529949901656712434177270 5
a_2	-97814534121904016489335854969382238 91
a_1	68698141669167832949898124129634274 66
a_0	11656834361594150492420981192388245365

TABLE 3.7 – Deux meilleures paires de polynômes pour RSA-768. La paire de polynôme utilisée pour la factorisation de RSA-768 a une valeur E de Murphy de $4{,}28 \times 10^{-13}$.

Chapitre 4

Les algorithmes NFS-DL et FFS pour le calcul de logarithme discret dans les corps finis

L'algorithme NFS-DL est une variante de l'algorithme NFS pour le calcul de logarithme discret dans les corps finis de grande caractéristique. Dans le cas des corps finis de petite caractéristique, le calcul de logarithme discret peut se faire avec l'algorithme FFS, qui possède beaucoup de similarités avec l'algorithme NFS-DL. Après avoir présenté le problème du calcul de logarithme discret dans les corps finis dans la section 4.1, nous présenterons les algorithmes NFS-DL et FFS dans les sections 4.2 et 4.4 respectivement, en particulier les similarités et les différences avec l'algorithme NFS pour la factorisation. Dans la section 4.3, nous détaillerons le calcul de logarithme discret dans un corps premier de 180 chiffres effectué avec l'algorithme NFS-DL et qui est, au moment où cette thèse est écrite, le record de calcul de logarithme discret dans un corps premier. Dans la section 4.5, nous détaillerons le calcul de logarithme discret dans $\mathbb{F}_{2^{809}}$ effectué avec l'algorithme FFS.

Dans tout ce chapitre, nous noterons p un nombre premier, n un entier positif non nul et nous nous intéresserons au calcul de logarithme discret dans le corps fini $\mathbb{F}_{p^n}$.

4.1 Le logarithme discret dans les corps finis

Comme nous l'avons vu dans l'introduction, la difficulté du calcul de logarithme discret dans les corps finis est à la base de la sécurité de beaucoup de systèmes cryptographiques. Dans la suite, ℓ désignera le cardinal du sous-groupe de $\mathbb{F}_{p^n}^*$ dans lequel les logarithmes sont calculés. D'après le théorème de Pohlig–Hellman [121], il suffit de calculer les logarithmes modulo tous les facteurs premiers de $p^n - 1$. Cependant, pour les algorithmes que nous allons décrire dans ce chapitre, il n'est pas nécessaire que ℓ soit un nombre premier, la seule contrainte est, en général, que ℓ divise $p^n - 1$.

Tous les algorithmes utilisés pour le calcul de logarithme discret dans les corps finis se basent sur le même principe, appelé *calcul d'indices*, qui peut se décomposer en trois étapes. La première étape est le choix d'un ensemble $\mathcal{B}$ de «petits» éléments de $\mathbb{F}_{p^n}$ et la recherche de combinaisons linéaires entre les logarithmes des éléments de $\mathcal{B}$. La deuxième étape est le calcul des logarithmes des éléments de $\mathcal{B}$ *via* la résolution d'un système linéaire construit à partir des relations de

l'étape précédente. La dernière étape est la recherche d'une relation entre l'élément cible, dont le logarithme doit être calculé, et les éléments de $\mathcal{B}$ dont les logarithmes sont connus.

Il existe, pour tout corps fini, au moins un algorithme permettant le calcul de logarithme discret dont la complexité est, au plus, en $L_{p^n}(1/3, c)$, où c est une constante qui dépend du corps fini. Le choix de l'algorithme à utiliser dépend du rapport entre n et la taille de p. Dans le cas où $p > L_{p^n}(2/3)$, appelé *grande caractéristique*, le meilleur algorithme est l'algorithme NFS-DL, dont la complexité est en $L_{p^n}\left(1/3, \sqrt[3]{64/9}\right)$, et qui sera présenté dans la section 4.2. Dans le cas où $L_{p^n}(1/3) < p < L_{p^n}(2/3)$, appelé *moyenne caractéristique*, le meilleur algorithme est l'algorithme NFS-HD [76, section 3.1], dont la complexité est en $L_{p^n}\left(1/3, \sqrt[3]{128/9}\right)$. Et dans le cas où $p < L_{p^n}(1/3)$, appelé *petite caractéristique*, l'algorithme FFS, dont la complexité est en $L_{p^n}\left(1/3, \sqrt[3]{32/9}\right)$ et qui sera présenté dans la section 4.4, était, jusqu'en 2013, le meilleur algorithme. Il existe aussi des variantes des algorithmes précédents pour les corps finis se trouvant proches de la limite entre deux algorithmes.

Durant l'année 2013, plusieurs avancées majeures ont été faites pour le calcul du logarithme discret dans les corps finis, en particulier pour les corps finis de petite caractéristique. Cela a commencé par des nouveaux records pour la moyenne caractéristique par Joux en décembre 2012 et janvier 2013 utilisant une nouvelle méthode décrite dans [74]. Puis, pour la petite caractéristique, un nouvel algorithme a été proposé par Göloğlu *et al.* [62] en février 2013, dont la complexité est en $L_{p^n}\left(1/3, \sqrt[3]{4/9}\right)$. Ensuite, toujours pour la petite caractéristique, Joux a décrit fin février 2013 dans [72] un algorithme dont la complexité est en $L_{p^n}(1/4 + o(1), c)$, pour une constante c non précisée. En avril 2013, Göloğlu *et al.* ont montré dans [63] qu'en mettant en commun ces deux nouvelles méthodes, il était possible de construire un algorithme dont la complexité est en $L_{p^n}(1/4, c)$, où c est une constante connue et inférieure à 1. Finalement, en juin 2013, un algorithme de complexité quasi-polynomiale pour le calcul de logarithme discret dans les corps finis de petite caractéristique a été publié par Barbulescu *et al.* [15]. Une variante de cet algorithme comportant moins d'heuristiques a ensuite été proposée en 2014 par Granger *et al.* [66].

Il existe plusieurs records de calcul de logarithme discret en fonction du type de corps fini considéré et donc de l'algorithme utilisé. Le record pour les corps du type $\mathbb{F}_{2^n}$, où n n'est pas un nombre premier, est le calcul de logarithme discret dans le corps $\mathbb{F}_{2^{9234}} = \mathbb{F}_{(2^{18})^{513}}$ [65]. Dans le cas des corps de caractéristique 3, le record est le calcul de logarithme discret dans le corps $\mathbb{F}_{3^{2395}} = \mathbb{F}_{(3^{479})^5}$ [77]. Ces calculs ont été effectués avec les nouveaux algorithmes découverts durant l'année 2013. Ces nouveaux algorithmes fonctionnent pour des corps dont l'exposant n a une forme particulière. Pour pouvoir les utiliser, il faut donc parfois plonger le corps dans lequel le logarithme discret doit être calculé dans un corps plus grand, le pire cas étant le cas des extensions premières pour lesquelles la limite pratique entre ces nouveaux algorithmes, asymptotiquement plus rapides, et FFS n'était pas évidente. En octobre 2014, Kleinjung a calculé un logarithme discret dans $\mathbb{F}_{2^{1279}}$ [82] en utilisant ces nouveaux algorithmes. Le record précédent, effectué avec l'algorithme FFS, était le calcul de logarithme discret dans le corps $\mathbb{F}_{2^{809}}$ [12], qui est détaillé dans la section 4.5. Le calcul dans $\mathbb{F}_{2^{1279}}$ a nécessité environ moitié moins de temps que le calcul dans $\mathbb{F}_{2^{809}}$, ce qui montre donc que les nouveaux algorithmes sont compétitifs en pratique pour les extensions premières pour des corps de cette taille. Le record dans le cas de la caractéristique moyenne est le calcul de logarithme discret dans $\mathbb{F}_{p^{57}}$, avec $p = 33\,341\,353$, par Joux [73] en janvier 2013. Le record dans le cas des corps premiers est le calcul de logarithme discret dans $\mathbb{F}_p$ avec un nombre premier p de 180 chiffres [26]. Des détails sur ce calcul sont donnés dans la section 4.3. Dans le cas de la grande caractéristique avec $n = 2$, le record est le calcul de logarithme discret dans $\mathbb{F}_{p^2}$ avec un nombre premier p de 90 chiffres [14].

4.2 L'algorithme NFS-DL pour le calcul de logarithme discret dans les corps premiers

L'algorithme NFS-DL est une adaptation de l'algorithme NFS pour le calcul de logarithme discret dans les corps finis de grande caractéristique. La première description de l'algorithme NFS-DL pour le calcul de logarithme discret dans les corps premiers, c'est-à-dire dans le cas où $n = 1$, a été donnée par Gordon dans [64]. L'algorithme NFS-DL pour $n > 1$ a été décrit pour la première fois par Schirokauer dans [132]. Dans cette section, nous présenterons la version de l'algorithme NFS-DL pour le calcul de logarithme discret dans les corps premiers.

4.2.1 Description générale de l'algorithme NFS-DL

Le contexte de l'algorithme NFS-DL est similaire à celui de l'algorithme NFS, décrit dans le chapitre 2. Nous utiliserons certaines notations définies dans le chapitre 2 sans les définir de nouveau dans ce chapitre. L'algorithme NFS-DL commence de la même façon que l'algorithme NFS, où l'entier N est remplacé par le nombre premier p, en sélectionnant deux polynômes f_1 et f_2 à coefficients entiers, irréductibles sur $\mathbb{Q}$, premiers entre eux sur $\mathbb{Q}$ et ayant une racine commune modulo p. Comme lors de la présentation de l'algorithme NFS, nous supposerons dans la suite que les polynômes f_1 et f_2 sont unitaires. La première différence entre NFS-DL et NFS est le fait que l'algorithme NFS-DL ne cherche plus à factoriser des idéaux dans les ordres $\mathbb{Z}[\alpha_1]$ et $\mathbb{Z}[\alpha_2]$ des corps de nombres K_1 et K_2 mais dans les anneaux des entiers $\mathcal{O}_{K_1}$ et $\mathcal{O}_{K_2}$. La figure 4.1 est l'adaptation du diagramme commutatif de la figure 2.1 pour l'algorithme NFS-DL.

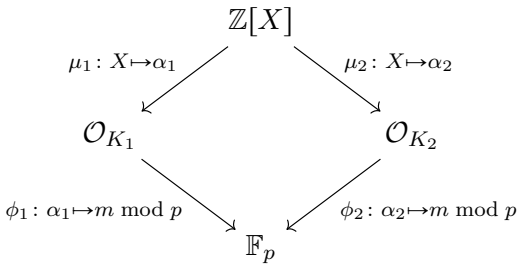

FIGURE 4.1 – Diagramme commutatif à la base de l'algorithme NFS-DL.

Une relation est toujours définie comme deux entiers a et b premiers entre eux tels que, simultanément pour $i \in \{1, 2\}$, $\mathcal{N}_i(a - b\alpha_i)$ est B_i-friable. Sauf que, dans le cas de NFS-DL, elle est interprétée comme la factorisation de l'idéal principal $(a - b\alpha_i)\mathcal{O}_{K_i}$ en idéaux premiers de $\mathcal{O}_{K_i}$, pour $i \in \{1, 2\}$. Pour passer de la factorisation de la norme en nombres premiers à la factorisation en idéaux premiers de $\mathcal{O}_{K_i}$, il faut distinguer deux cas. Soient $i \in \{1, 2\}$ et π un nombre premier tels que $\pi \mid \mathcal{N}_i(a - b\alpha_i)$, alors, comme pour NFS, il existe un unique $\rho \in R_i(\pi)$ tel que $a \equiv b\rho \pmod{\pi}$ et l'idéal $\mathfrak{p} = \pi\mathcal{O}_{K_i} + (\alpha_i - \rho)\mathcal{O}_{K_i}$ contient $a - b\alpha_i$. Les deux cas à distinguer sont les suivants :

— soit π ne divise pas $[\mathcal{O}_{K_i} : \mathbb{Z}[\alpha_i]]$, l'indice de $\mathbb{Z}[\alpha_i]$ dans $\mathcal{O}_{K_i}$, alors l'idéal $\mathfrak{p}$ est un idéal premier non nul de $\mathcal{O}_{K_i}$ et la valuation de $(a - b\alpha_i)\mathcal{O}_{K_i}$ en cet idéal est égale à la valuation de la norme $\mathcal{N}_i(a - b\alpha_i)$ en π ;

— soit π divise $[\mathcal{O}_{K_i} : \mathbb{Z}[\alpha_i]]$, alors l'idéal $\mathfrak{p}$ peut ne pas être un idéal premier de $\mathcal{O}_{K_i}$, et il faut, dans ce cas, le factoriser et calculer les valuations de ses facteurs, en utilisant par exemple l'algorithme 4.8.17 de [38].

Les idéaux $\mathfrak{p}$ tels que π divise $[\mathcal{O}_{K_i} : \mathbb{Z}[\alpha_i]]$ et qui ne sont pas des idéaux premiers de $\mathcal{O}_{K_i}$ seront appelés dans la suite des *mauvais idéaux* et tous les idéaux premiers apparaissant dans leurs factorisations doivent être ajoutés à la base de facteurs. En pratique, comme le calcul de $[\mathcal{O}_{K_i} : \mathbb{Z}[\alpha_i]]$ peut être plus coûteux que le calcul de logarithme discret dans $\mathbb{F}_p$, le discriminant du polynôme f_i, qui est divisible par le carré de $[\mathcal{O}_{K_i} : \mathbb{Z}[\alpha_i]]$, peut être utilisé pour distinguer les deux cas précédents.

Pour faire le lien entre les relations trouvées par l'algorithme NFS-DL et le calcul de logarithme dans $\mathbb{F}_p$, il est nécessaire d'étudier les unités de l'anneau $\mathcal{O}_{K_i}$. Le groupe des unités de $\mathcal{O}_{K_i}$, noté $\mathcal{O}_{K_i}^*$, est le groupe des éléments non nuls et inversibles de $\mathcal{O}_{K_i}$. D'après le théorème des unités de Dirichlet,

$$\mathcal{O}_{K_i}^* \simeq \mathcal{O}_{K_i,\text{tors}}^* \times \mathbb{Z}^{r_{i,\mathbb{R}}+r_{i,\mathbb{C}}-1},$$

où $\mathcal{O}_{K_i,\text{tors}}^*$ est le groupe cyclique fini des racines de l'unité de K_i, $r_{i,\mathbb{R}}$ est le nombre de racines réelles de f_i et $2r_{i,\mathbb{C}}$ est le nombre de racines complexes de f_i. Dans la suite, nous supposerons que ℓ est premier avec l'ordre de tous les éléments de $\mathcal{O}_{K_i,\text{tors}}^*$. En pratique, ce n'est pas contraignant car s'il existe un élément de $\mathcal{O}_{K_i,\text{tors}}^*$ d'ordre π, où π est un nombre premier, alors $\pi - 1$ doit diviser d_i, donc π est petit, le calcul de logarithme discret dans le sous-groupe d'ordre $\pi^{v_\pi(\ell)}$ est alors facile et ℓ peut être remplacé par $\ell/\pi^{v_\pi(\ell)}$. Dans la suite, nous noterons ε_i un générateur de $\mathcal{O}_{K_i,\text{tors}}^*$, r_i le rang du groupe des unités, c'est-à-dire $r_i = r_{i,\mathbb{R}} + r_{i,\mathbb{C}} - 1$ et $\{ u_{i,j} \}_{0 \leqslant j < r_i}$ un ensemble d'*unités fondamentales*, c'est-à-dire un ensemble de générateurs de la partie de non torsion de $\mathcal{O}_{K_i}^*$.

Pour l'algorithme NFS-DL, une relation est vue comme une égalité modulo ℓ entre des valeurs associées aux idéaux, appelées *logarithmes virtuels* et notées vlog, qui s'expriment en fonction de logarithmes d'éléments de $\mathbb{F}_p$ [133]. Nous allons expliquer le lien entre les logarithmes virtuels des idéaux de $\mathcal{O}_{K_i}$, pour $i \in \{ 1, 2 \}$ fixé, et les logarithmes dans $\mathbb{F}_p$. Pour cela, nous noterons h_i le nombre de classes de $\mathcal{O}_{K_i}$ et nous supposerons que ℓ et h_i sont premiers entre eux. Par définition de h_i, pour tout idéal $\mathfrak{I}$ de $\mathcal{O}_{K_i}$, $\mathfrak{I}^{h_i}$ est un idéal principal. Deux générateurs d'un idéal principal de $\mathcal{O}_{K_i}$ diffèrent d'une unité de $\mathcal{O}_{K_i}$. Supposons que pour tout idéal $\mathfrak{p} \in \mathcal{B}_i$, un générateur de l'idéal principal $\mathfrak{p}^{h_i}$, noté $\gamma_{i,\mathfrak{p}}$, soit choisi, alors, si la paire (a, b) est une relation, l'égalité suivante entre les idéaux :

$$(a - b\alpha_i)\mathcal{O}_{K_i} = \prod_{\mathfrak{p} \in \mathcal{B}_i} \mathfrak{p}^{e_{i,\mathfrak{p}}}$$

peut se réécrire comme l'égalité suivante entre éléments de $\mathcal{O}_{K_i}$:

$$(a - b\alpha_i)^{h_i} = \varepsilon_i^{e_{\varepsilon_i}} \prod_{j=0}^{r_i-1} u_{i,j}^{e_{u_{i,j}}} \prod_{\mathfrak{p} \in \mathcal{B}_i} \gamma_{i,\mathfrak{p}}^{e_{i,\mathfrak{p}}},$$

où les $e_{i,\mathfrak{p}}$ correspondent aux valuations en $\mathfrak{p}$ et où les e_{ε_i} et $e_{u_{i,j}}$ sont des entiers dépendant du choix des générateurs $\gamma_{i,\mathfrak{p}}$. L'image de ce produit par le morphisme ϕ_i donne une combinaison linéaire entre les logarithmes d'éléments de $\mathbb{F}_p$:

$$h_i \log \phi_i(a - b\alpha_i) \equiv e_{\varepsilon_i} \log \phi_i(\varepsilon_i) + \sum_{j=0}^{r_i-1} e_{u_{i,j}} \log \phi_i(u_{i,j}) + \sum_{\mathfrak{p} \in \mathcal{B}_i} e_{i,\mathfrak{p}} \log \phi_i(\gamma_{i,\mathfrak{p}}) \pmod{\ell}.$$

En définissant le logarithme virtuel des idéaux de la base de facteurs et des unités par

$$\begin{aligned} \text{vlog}\, \mathfrak{p} &\equiv h_i^{-1} \log \phi_i(\gamma_{i,\mathfrak{p}}) \pmod{\ell} && \text{pour } \mathfrak{p} \in \mathcal{B}_i, \\ \text{vlog}\, u_{i,j} &\equiv h_i^{-1} \log \phi_i(u_{i,j}) \pmod{\ell} && \text{pour } 0 \leqslant j \leqslant r_i - 1, \\ \text{vlog}\, \varepsilon_i &\equiv h_i^{-1} \log \phi_i(\varepsilon_i) \pmod{\ell}, \end{aligned} \tag{4.1}$$

la dernière égalité peut s'écrire :

$$\log \phi_i(a - b\alpha_i) \equiv e_{\varepsilon_i} \operatorname{vlog} \varepsilon_i + \sum_{j=0}^{r_i-1} e_{u_{i,j}} \operatorname{vlog} u_{i,j} + \sum_{\mathfrak{p}\in\mathcal{B}_i} e_{i,\mathfrak{p}} \operatorname{vlog} \mathfrak{p} \pmod{\ell}.$$

Finalement, en se rappelant que $\phi_1(a-b\alpha_1) = \phi_2(a-b\alpha_2)$ et en remarquant que, comme l'ordre de ε_i est premier avec ℓ, $\operatorname{vlog} \varepsilon_i \equiv 0 \pmod{\ell}$, nous obtenons l'égalité suivante entre les logarithmes virtuels des deux côtés :

$$\sum_{j=0}^{r_1-1} e_{u_{1,j}} \operatorname{vlog} u_{1,j} + \sum_{\mathfrak{p}\in\mathcal{B}_1} e_{1,\mathfrak{p}} \operatorname{vlog} \mathfrak{p} \equiv \sum_{j=0}^{r_2-1} e_{u_{2,j}} \operatorname{vlog} u_{2,j} + \sum_{\mathfrak{p}\in\mathcal{B}_2} e_{2,\mathfrak{p}} \operatorname{vlog} \mathfrak{p} \pmod{\ell}. \tag{4.2}$$

Cette égalité repose sur la capacité à calculer un générateur de chaque idéal principal $\mathfrak{p}^{h_i}$, pour $\mathfrak{p} \in \mathcal{B}_i$, et à calculer un ensemble d'unités fondamentales. À part dans des cas particuliers, comme dans [13], ce sont des problèmes très difficiles pour des polynômes génériques. Cependant il est possible de remplacer la résolution de ces deux problèmes par le calcul de *fonctions de Schirokauer*. Une fonction de Schirokauer est une fonction Λ de $K_{i,\ell}/(K_{i,\ell})^\ell$ dans $(\mathbb{Z}/\ell\mathbb{Z})^{r_i}$, où $K_{i,\ell}$ est le sous-groupe multiplicatif de K_i^* défini par $K_{i,\ell} = \{\, \gamma \in K_i^* \mid \operatorname{pgcd}(\mathcal{N}_i(\gamma), \ell) = 1 \,\}$, qui vérifie $\Lambda(\beta\gamma) = \Lambda(\beta) + \Lambda(\gamma)$, pour tout β et γ dans $K_{i,\ell}$ et tel que Λ restreint à $\mathcal{O}_{K_i}^*$ est surjectif. Une fois qu'une fonction de Schirokauer est fixée, le choix de l'ensemble des unités fondamentales et des générateurs de $\mathfrak{p}^{h_i}$ se fait de façon implicite. De plus, il n'est pas nécessaire de les calculer. Si $\mathfrak{p}$ est un idéal de $\mathcal{B}_i$, $\gamma_{i,\mathfrak{p}}$ est défini comme l'unique générateur de l'idéal principal $\mathfrak{p}^{h_i}$ qui vérifie $\Lambda(\gamma_\mathfrak{p}) = (0, \ldots, 0)$ et, pour $0 \leqslant j < r_i$, l'unité fondamentale $u_{i,j}$ est définie comme l'unique unité telle que $\Lambda(u_{i,j}) = (0, \ldots, h_i, \ldots, 0)$, où h_i se trouve en $j^{\text{ème}}$ position. De plus, pour une relation (a, b) et pour $0 \leqslant j < r_i$, le coefficient $e_{u_{i,j}}$ de l'unité $u_{i,j}$ correspond à la $j^{\text{ème}}$ coordonnée de $\Lambda(a - b\alpha_i)$. Schirokauer a montré dans [133] qu'en définissant les logarithmes virtuels des idéaux de la base de facteurs et des unités comme dans (4.1), l'égalité (4.2) est toujours vraie. Dans [131], Schirokauer propose d'utiliser la fonction de Schirokauer suivante et montre qu'elle est bien définie :

$$\begin{array}{ccccc} K_{i,\ell}/(K_{i,\ell})^\ell & \to & \mathbb{Z}/\ell\mathbb{Z}[X] & \to & (\mathbb{Z}/\ell\mathbb{Z})^{r_i} \\ \gamma & \mapsto & G = \frac{\Gamma^{\Delta_i} - 1 \bmod (f_i, \ell^2)}{\ell} & \mapsto & \{\, G_0, G_1, \ldots, G_{r_i-1} \,\} \end{array}, \tag{4.3}$$

où $\Delta_i = \operatorname{ppcm} \{\, \ell^\delta - 1 \mid f_i \bmod \ell$ a un facteur irréductible de degré $\delta \,\}$, Γ est un polynôme de $\mathbb{Z}[X]$ tel que $\mu_i(\Gamma) = \gamma$ et G_j est le coefficient de degré j du polynôme G. Dans [133], des arguments heuristiques sont donnés pour justifier que cette fonction est surjective avec une grande probabilité lorsqu'elle est restreinte à $\mathcal{O}_{K_i}^*$. Pour résumer, une relation (a, b) sera interprétée comme l'égalité (4.2), où les $e_{u_{i,j}}$ sont soit les valuations des unités soit calculés à partir de la fonction de Schirokauer définie par (4.3).

Remarque 4.2.1. Dans le cas où le côté i est rationnel, c'est-à-dire dans le cas où $d_i = 1$, comme $\mathcal{O}_{K_i} = \mathbb{Z}$, le nombre de classes h_i est égal à 1 et le rang des unités r_i est égal à 1. Donc les logarithmes virtuels sont égaux aux logarithmes dans $\mathbb{F}_p$ sans qu'il ne soit nécessaire de calculer la contribution des unités ou d'utiliser des fonctions de Schirokauer.

Une fois que le nombre de relations trouvées est supérieur à la somme de la taille des deux bases de facteurs, il reste à résoudre un système linéaire pour trouver le logarithme virtuel de chaque idéal des bases de facteurs apparaissant dans au moins une relation. Finalement, si w est l'élément cible de $\mathbb{F}_p$ dont le logarithme doit être calculé, il reste à exprimer $\log(w)$ en fonction de logarithmes virtuels connus.

4.2.2 Les différentes étapes de l'algorithme NFS-DL

Les objectifs de la sélection polynomiale sont identiques dans le cas de NFS et NFS-DL. De plus, les différents algorithmes utilisés pour la sélection polynomiale linéaire et non linéaire pour NFS peuvent être utilisés sans changement pour NFS-DL. Il existe aussi des algorithmes qui ne fonctionnent que pour NFS-DL, car ils tirent parti du fait que p est premier et que donc des racines de polynômes modulo p peuvent être calculées, ce qui n'est pas le cas modulo un entier N dont la factorisation n'est pas connue. Un exemple d'un tel algorithme de sélection polynomiale est l'algorithme de Joux–Lercier décrit dans [75, section 2.1].

L'étape de crible est strictement identique pour les algorithmes NFS et NFS-DL. Cependant, il est nécessaire, avant l'étape de filtrage, de traiter toutes les relations trouvées pour gérer le cas des mauvais idéaux, c'est-à-dire ceux au-dessus de nombres premiers divisant le discriminant des polynômes f_1 ou f_2. L'ensemble des mauvais idéaux ainsi que la façon de les gérer peut être précalculé dès que les polynômes f_1 et f_2 sont connus, c'est-à-dire dès la fin de l'étape de sélection polynomiale.

Pour l'étape de filtrage, il existe deux différences majeures entre les algorithmes NFS et NFS-DL. La matrice est construite comme pour l'algorithme NFS — une ligne de la matrice correspond au vecteur des valuations d'une relation — mais n'est plus définie sur $\mathbb{Z}/2\mathbb{Z}$ mais sur $\mathbb{Z}/\ell\mathbb{Z}$. La deuxième différence est que ce n'est plus le noyau à gauche mais le noyau à droite de la matrice qui doit être calculé. Le chapitre 5 de cette thèse est consacré à l'étude de l'étape de filtrage pour NFS, NFS-DL et FFS. Une fois l'étape de filtrage terminée, il faut calculer et ajouter, pour chaque ligne de la matrice et pour $i \in \{1, 2\}$, les r_i colonnes qui correspondent à la contribution, du côté i, des unités, lorsque cela est possible, ou des fonctions de Schirokauer, dans le cas général.

L'étape d'algèbre linéaire est différente entre NFS et NFS-DL. Tout d'abord l'algèbre linéaire ne s'effectue plus modulo 2 mais modulo ℓ. Cela ne change pas la complexité des algorithmes en nombre d'opérations, mais augmente le coût de chaque opération et la mémoire nécessaire. De plus, les colonnes ajoutées après l'étape de filtrage alourdissent la matrice. En effet, la matrice à la fin de l'étape de filtrage est creuse et ne contient que des petits coefficients (d'après les données expérimentales provenant des exemples de la section 5.6.1, environ 99 % des coefficients non nuls valent entre -10 et 10). Or les coefficients des colonnes ajoutées entre l'étape de filtrage et l'étape d'algèbre linéaire sont uniformément distribués dans $[0, \ell - 1]$. Une description de l'étape d'algèbre linéaire dans le cas de NFS-DL peut être trouvée dans [71].

Les étapes de calcul de caractères et de racines carrées n'existent pas pour l'algorithme NFS-DL. Cependant, une fois l'étape d'algèbre linéaire terminée, et donc les logarithmes virtuels des idéaux des bases de facteurs connus, il reste à calculer $\log(w)$, le logarithme de l'élément cible. Cette étape est appelée *l'étape de descente* ou *l'étape de logarithme individuel*. Tout d'abord, deux bornes $B_{1,\text{desc}}$ et $B_{2,\text{desc}}$, supérieures respectivement à B_1 et B_2 sont choisies et le logarithme de w est exprimé en fonction de logarithmes virtuels d'idéaux premiers non nuls de degré 1 de normes inférieures à $B_{1,\text{desc}}$ du côté 1 et $B_{2,\text{desc}}$ du côté 2. Ensuite, il faut, pour $i \in \{1, 2\}$ et pour chaque idéal $\mathfrak{p}$ dont la norme est comprise entre $B_{i,\text{desc}}$ et B_i, trouver une relation faisant intervenir $\mathfrak{p}$ et des idéaux de normes strictement plus petites. Pour faire cela, la technique de crible par spécial-$\mathfrak{q}$ de l'étape de crible est utilisée, ce qui permet, en pratique, de réutiliser presque sans changement le code de l'étape de crible. Il faut continuer ainsi, jusqu'à ce que tous les logarithmes virtuels puissent être exprimés en fonction de logarithmes virtuels d'idéaux des bases de facteurs. Ceci permet d'obtenir le logarithme de w comme une combinaison linéaire des logarithmes virtuels connus, et donc de calculer $\log(w)$.

La complexité théorique de NFS-DL est la même que celle de NFS, mais, en pratique, pour

un entier N de la même taille que p, l'algorithme NFS est plus rapide que l'algorithme NFS-DL. Cela est principalement dû au fait que l'étape d'algèbre linéaire est beaucoup plus coûteuse dans le cas de NFS-DL que dans le cas de NFS.

4.3 Calcul de logarithme discret dans un corps premier de 180 chiffres

Nous allons maintenant donner des détails sur le calcul de logarithme discret avec l'algorithme NFS-DL dans le corps premier $\mathbb{F}_{p_{180}}$, où p_{180} est un nombre premier de 180 chiffres décimaux [26]. Ce calcul a été réalisé avec P. Gaudry, L. Imbert, H. Jeljeli et E. Thomé et terminé en juin 2014. Au moment de l'écriture de cette thèse, c'est le calcul record de logarithme discret dans un corps premier, le record précédent datait de février 2007 pour un nombre premier de 160 chiffres [80].

L'entier p_{180} de 180 chiffres décimaux (596 bits) qui a été utilisé est le plus petit entier p supérieur à RSA-180 tel que p et $(p-1)/2$ sont premiers.

$$\begin{aligned} p_{180} &= \text{RSA-180} + 625942 \\ &= 191147927718986609689229466631454649812986246276667354864188503638 \\ &\quad 807260703436799058776201365135161278134258296128109200046702912984 \\ &\quad 568752800330221777752773957404540495707852046983, \\ \ell &= \frac{p_{180}-1}{2}. \end{aligned}$$

L'élément $5 \in \mathbb{F}_{p_{180}}$ est un générateur du groupe multiplicatif et sera utilisé comme base pour tous les logarithmes donnés dans cette section. L'élément cible «aléatoire» dont nous avons calculé le logarithme est :

$$\begin{aligned} w &= \text{RSA-1024} \bmod p_{180} \\ &= 681880801095823308798688613309985061517748546004037006257972999 \\ &\quad 275589951627403211122609736386197579226462423021048854375367450 \\ &\quad 802992488520650800083583097358751924807244965303259 27. \end{aligned}$$

La majorité du code utilisé pour ce calcul record est disponible dans `cado-nfs` [33]. Comme `cado-nfs` est une implémentation de l'algorithme NFS pour la factorisation, certaines modifications ont été nécessaires.

La sélection polynomiale a été faite en utilisant l'algorithme de Kleinjung [81] implémenté dans `cado-nfs`, sans qu'aucun changement n'ait été nécessaire. Après l'équivalent de 2 mois de calcul sur un cœur d'un processeur Intel E5-2650 à 2 GHz, la meilleure paire de polynômes qui a été trouvée, et donc utilisée dans la suite du calcul, est la suivante [4] :

$$\begin{aligned} f_1 &= 633287365084897327346023x - 256683250895227560765113615087202 91 \\ f_2 &= 17153280x^5 + 55645402596756x^4 + 28964242910035546694 5x^3 \\ &\quad - 583903418367235648170825362 8x^2 - 348919545982234412735036794146466 0x \\ &\quad - 247746689873713970845286181645074189 28. \end{aligned}$$

Le résultant de f_1 et f_2 est égal à p_{180}.

4. En utilisant le nouvel algorithme d'optimisation de la taille présenté dans le chapitre 3, une meilleure paire de polynômes a été trouvée, avec une valeur E de Murphy plus grande d'environ 7 %, en deux fois moins de temps.

L'étape de crible a été faite en utilisant le crible par réseaux implémenté dans `cado-nfs`, sans qu'aucun changement n'ait été nécessaire. Les bornes de friabilité B_1 et B_2 étaient identiques des deux côtés et valaient $B_1 = B_2 = 80\,000\,000$. Du côté rationnel, deux grands premiers inférieurs à 2^{29} étaient autorisés et du côté algébrique, trois grands premiers inférieurs à 2^{30} étaient autorisés. Des spécial-`q` ont été utilisés du côté algébrique, car c'est le côté dont la norme est la plus grande pour les valeurs de a et b considérées. Les idéaux premiers de degré 1 du côté algébrique dont la norme est comprise dans l'intervalle $[80\,000\,000, 380\,000\,000]$ ont été utilisés comme spécial-`q` et pour chaque spécial-`q`, 2^{31} paires (a, b) ont été considérées. Au total, un peu plus de 245 millions de relations ont été trouvées en l'équivalent de 49,5 années de calcul sur un cœur d'un processeur Intel E5-2650 à 2 GHz. Une fois les doublons éliminés, il restait un peu moins de 175 millions de relations uniques.

Pour l'étape de filtrage, nous avons utilisé le code spécifique pour le calcul de logarithme discret disponible dans `cado-nfs`. Ce code est l'implémentation des algorithmes décrits dans le chapitre 5. De plus, la section 5.5.1 contient des détails supplémentaires sur cette étape de filtrage. La matrice produite par l'étape de filtrage était presque une matrice carrée avec 7,287 millions de lignes. L'étape de filtrage a pris l'équivalent de 5 heures de calcul sur un cœur d'un processeur Intel E5-2650 à 2 GHz.

Il n'existait pas de code pour le calcul des fonctions de Schirokauer dans `cado-nfs`, il a donc été écrit pour ce record et intégré dans `cado-nfs`. Le calcul des fonctions de Schirokauer a pris l'équivalent de 0,9 année sur un cœur d'un processeur Intel E5-2650 à 2 GHz.

L'étape d'algèbre linéaire a été faite à l'aide de l'implémentation de Jeljeli de l'algorithme de Wiedemann par bloc décrite dans [71]. L'algèbre linéaire a été effectué sur la matrice produite par l'étape de filtrage, qui possédait en moyenne 150 coefficients non nuls par ligne, à laquelle ont été ajoutées 4 colonnes [5] denses correspondant à la contribution des fonctions de Schirokauer du côté algébrique. Le noyau à droite de cette nouvelle matrice a été calculé modulo ℓ en utilisant l'algorithme de Wiedemann par bloc avec les paramètres de bloc $m = 24$ et $n = 12$. Le calcul a été fait sur un cluster de 48 nœuds, possédant chacun 2 processeurs de 8 cœurs Intel Xeon E5-2650 à 2 GHz et reliés par Infiniband. Les 12 séquences ont été lancées en parallèle sur 4 nœuds chacune. La première et la dernière étapes, constituées principalement de produits matrice–vecteur, ont pris 38 jours (soit environ l'équivalent de 80 années pour un cœur). La deuxième étape a été effectuée sur 144 cœurs uniquement et a pris 15 heures (soit environ l'équivalent de 0,25 année pour un cœur).

Une fois l'étape d'algèbre linéaire terminée, les logarithmes virtuels de presque 99 % des idéaux premiers non nuls de degré 1 dont la norme est inférieure à 2^{29} pour le côté rationnel et 2^{30} pour le côté algébrique étaient connus, par exemple :

$$\begin{aligned}\log(2) = {} & 14394742424980404689468652122583501155340452982569859689394995\\ & 37509189519718986652049683275189725501776470006513329773475176 6\\ & 54387676076061308411099885253085259407173106476434760 8\\ \log(3) = {} & 12540255374709186945948836756152071692814462540757959805173613 9\\ & 49252707487386035786690693592163692301618098936460400547559095 2\\ & 63524577946074538124684456888597268322428333393912658 4\end{aligned}$$

Ensuite, l'élément w, dont le logarithme doit être calculé, a été réécrit de la façon suivante :

$$w \equiv \frac{\Pi_{\text{num}} \times 1100000227 \times 4515841907 \times 184894709723 \times 10396717489297 \times 21553462907467 \times 1362405950935927}{\Pi_{\text{den}} \times 958874499397 \times 3517522326737 \times 24482357467423 \times 1050023653153247} \pmod{p_{180}},$$

5. Nous aurions pu utiliser uniquement 3 colonnes de fonctions de Schirokauer car f_2 possède 3 racines réelles et 2 racines complexes et donc $r_2 = 3 + 1 - 1 = 3$.

où Π_{num} et Π_{den} sont des entiers non nuls dont tous les facteurs premiers sont inférieurs à 2^{29} et dont les logarithmes sont donc connus. Il reste donc à calculer le logarithme des 10 facteurs premiers dont la taille est comprise en 2^{29} et 2^{51}. En utilisant la technique de descente par spécial-$\mathfrak{q}$, les logarithmes de ces 10 éléments ont pu être exprimés en fonction de logarithmes virtuels connus. Pour cela, quelques modifications au code utilisé pour la génération de relations ont été nécessaires. Au final, après quelques heures de calcul, le logarithme de w a été calculé :

$$\begin{aligned}\log(w) = {} & 13867056612682358487962586132633332631236394382562103922021558 3\\ & 34615378333627255995552197035730130291204631078290865945075854 9\\ & 10809291833135221575134605475521667300593993318639777 7.\end{aligned}$$

4.4 L'algorithme FFS pour le calcul de logarithme discret en petite caractéristique

L'algorithme FFS (pour *Function Field Sieve*) est un algorithme de calcul de logarithme discret pour les corps finis $\mathbb{F}_{p^n}$ de petite caractéristique, qui a été inventé par Adleman en 1994 [2]. Il est similaire à NFS-DL mais utilise des corps de fonctions à la place des corps de nombres. Les outils et les idées à la base de l'algorithme FFS sont les mêmes que ceux de l'algorithme NFS-DL, où l'anneau des entiers $\mathbb{Z}$ est remplacé par l'anneau de polynômes $\mathbb{F}_p[t]$ et le corps des rationnels $\mathbb{Q}$ par le corps des fractions $\mathbb{F}_p(t)$.

L'algorithme FFS commence par sélectionner deux polynômes $f_1, f_2 \in \mathbb{F}_p[t][X]$ irréductibles et premiers entre eux tels qu'il existe un polynôme irréductible $\varphi \in \mathbb{F}_p[t]$ de degré n divisant le résultant par rapport à la variable X de f_1 et f_2. Dans la suite, les polynômes f_1 et f_2 seront toujours considérés comme des polynômes en X dont les coefficients sont des polynômes en t et donc lorsque nous parlerons de degré ou de résultant, ce sera toujours par rapport à la variable X. Pour des raisons de simplicité, nous considérerons dans la suite que les polynômes f_1 et f_2 sont unitaires. Le polynôme φ est utilisé comme polynôme de définition de $\mathbb{F}_{p^n}$ comme extension de degré n de $\mathbb{F}_p$. Comme toutes les extensions de degré n de $\mathbb{F}_p$ sont isomorphes, n'importe quel polynôme irréductible de degré n peut être utilisé, ce qui permet d'avoir plus de choix lors de l'étape de sélection polynomiale. Le résultant de f_1 et f_2 étant un multiple du polynôme φ, il existe un polynôme $m \in \mathbb{F}_p[t]$ qui est une racine commune de f_1 et f_2 modulo φ.

Ensuite, de façon similaire à l'algorithme NFS-DL, nous définissons, pour $i \in \{1, 2\}$, le corps de fonctions $K_i = \mathbb{F}_p(t)[X]/f_i$, la racine α_i de f_i dans ce corps de fonctions et l'anneau des entiers $\mathcal{O}_{K_i}$. Les idéaux premiers non nuls de degré 1 de K_i sont en bijection avec l'ensemble des paires (π, ρ), où π et ρ sont deux polynômes de $\mathbb{F}_p[t]$ tels que π est irréductible et unitaire, $\deg(\rho) < \deg(\pi)$ et ρ est une racine de f_i modulo π. La base de facteurs $\mathcal{B}_i$ est composée de tous les idéaux $\mathfrak{p}$ premiers non nuls de degré 1 au-dessus des polynômes irréductibles et unitaires de $\mathbb{F}_p[t]$ de degré inférieur ou égal à la borne de friabilité B_i. Par analogie au cas des corps de nombres, si a et b sont deux polynômes de $\mathbb{F}_p[t]$ premiers entre eux, la norme de $a - b\alpha_i$, notée $\mathcal{N}_i(a - b\alpha_i)$, est définie par $\mathcal{N}_i(a - b\alpha_i) = \text{Res}(a - bX, f_i)$. La figure 4.2 est l'adaptation du diagramme commutatif de la figure 4.1 pour l'algorithme FFS.

Une relation est une paire (a, b) de polynômes de $\mathbb{F}_p[t]$ premiers entre eux tels que, pour $i \in \{1, 2\}$, les idéaux premiers divisant l'idéal principal de $\mathcal{O}_{K_i}$ engendré par $a - b\alpha_i$ appartiennent tous à la base de facteurs $\mathcal{B}_i$, c'est-à-dire tous les polynômes irréductibles de $\mathbb{F}_p[t]$ divisant $\mathcal{N}_i(a - b\alpha_i)$ sont de degré inférieur ou égal à B_i. Une relation est interprétée comme une égalité modulo ℓ entre les logarithmes virtuels des idéaux des bases de facteurs. Comme pour NFS-DL, pour donner un sens à ces logarithmes virtuels, il faut prendre en compte les unités, ce qui peut

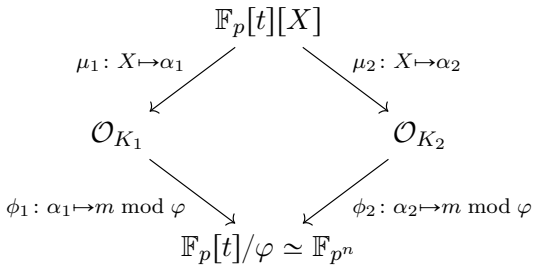

FIGURE 4.2 – Diagramme commutatif à la base de l'algorithme FFS.

être fait en considérant les valuations en les places à l'infini [3]. Tout d'abord, les unités de torsion de l'anneau des entiers $\mathcal{O}_{K_i}$ sont isomorphes à $\mathbb{F}_p^*$, et donc, comme pour NFS-DL, pour que le logarithme virtuel de ces unités de torsion soit nul, nous supposerons dans la suite que ℓ est un diviseur de $(p^n - 1)/(p - 1)$ premier avec $p - 1$. En pratique, ce n'est pas très contraignant car, comme p est petit, le calcul de logarithme discret dans le sous-groupe d'ordre $p - 1$ est facile. De plus, si le corps de fonctions K_i possède $r_i + 1$ places à l'infini, le rang du groupe des unités de $\mathcal{O}_{K_i}$ est r_i [144, p. 89] et il est donc nécessaire, pour que les logarithmes virtuels soient correctement définis, de considérer les valuations en r_i places à l'infini. Dans [101], Matsumoto propose d'utiliser des corps de fonctions provenant de courbes C_{ab} qui possèdent une seule place à l'infini, éliminant donc le besoin de considérer les valuations à l'infini.

Comme pour NFS-DL, l'anneau des entiers de $\mathcal{O}_{K_i}$ est en général strictement plus grand que $\mathbb{F}_p[t][X]/f_i$, il existe donc des mauvais idéaux. Ils sont ajoutés à la base de facteurs et comme pour NFS-DL, ils seront traités une fois les relations calculées et avant l'étape de filtrage. Mais cela est plus facile à gérer dans le cas de FFS car l'anneau $\mathcal{O}_{K_i}$ et l'indice $[\mathcal{O}_{K_i} : \mathbb{F}_p[t][X]/f_i]$ peuvent être calculés en temps polynomial dans le cas des corps de fonctions alors que ce n'est pas le cas pour les corps de nombres.

Remarque 4.4.1. Par analogie avec les algorithmes NFS et NFS-DL, dans le cas où le degré de l'un de deux polynômes vaut 1, tout ce qui se rapporte à ce côté est qualifié de rationnel et tout ce qui se rapporte à l'autre côté est qualifié d'algébrique. Si le côté i est rationnel, alors le corps de fonctions K_i est égal à $\mathbb{F}_p(t)$, l'anneau des entiers $\mathcal{O}_{K_i}$ est égal à $\mathbb{F}_p[t]$, les idéaux premiers non nuls correspondent aux polynômes irréductibles et unitaires de $\mathbb{F}_p[t]$ et, si a et b sont deux polynômes de $\mathbb{F}_p[t]$, $\mathcal{N}_i(a - b\alpha_i) = a - bm$, où m est la racine commune de f_1 et f_2. Le corps de fonctions $\mathbb{F}_p(t)$ n'a qu'une seule place à l'infini, il n'est donc pas nécessaire de considérer les valuations de cette place à l'infini et si $\mathfrak{p}$ est l'idéal premier non nul de $\mathbb{F}_p[t]$ correspondant à un polynôme irréductible et unitaire π alors $\operatorname{vlog} \mathfrak{p} \equiv \log \phi_i(\pi) \pmod{\ell}$, où $\phi_i(\pi)$ est l'image de la réduction de π modulo φ vue comme un élément de $\mathbb{F}_{p^n}$.

L'étape de sélection polynomiale de l'algorithme FFS est plus facile que dans le cas des algorithmes NFS ou NFS-DL. En effet, l'équivalent pour FFS du problème de sélection polynomiale dans le cas de NFS ou NFS-DL serait de trouver deux polynômes dont le résultant est un multiple d'un polynôme irréductible de $\mathbb{F}_p[t]$ donné, or dans le cas de FFS, le résultant de f_1 et f_2 peut être un multiple de *n'importe quel* polynôme irréductible de $\mathbb{F}_p[t]$ de degré n. Ceci permet de trouver des polynômes dont le degré en t des coefficients est petit, ce qui permet d'obtenir des normes de faible degré en t. Dans le cas de NFS et NFS-DL, le théorème de Canfield–Erdős–Pomerance permet d'estimer la probabilité de friabilité d'un entier de taille donnée pour une borne de friabilité donnée. Dans le cas de FFS, le théorème de Panario–Gourdon–Flajolet [119] permet, de façon similaire, d'estimer, étant donnée une borne sur le degré des facteurs, la probabilité de friabilité

d'un polynôme de degré donné. Plus de détails sur l'étape de sélection polynomiale pour FFS peuvent être trouvés dans [9].

Les différentes méthodes de crible utilisées dans NFS et NFS-DL peuvent être adaptées au cas de FFS, comme décrit dans [49]. De plus, la partie de l'étape de crible consistant à tester la friabilité de plusieurs paires (a, b) sélectionnées par le crible est grandement simplifiée par le fait qu'il existe des algorithmes en temps polynomial pour la factorisation des polynômes dans les corps finis.

Une fois que suffisamment de relations ont été calculées, c'est-à-dire que le nombre de relations uniques est supérieur à $\#\mathcal{B}_1 + \#\mathcal{B}_2$, et que les mauvais idéaux ont été gérés, la matrice peut être construite comme pour l'algorithme NFS-DL. L'étape de filtrage pour FFS est strictement identique à celle de NFS-DL. Pour l'étape d'algèbre linéaire, il existe une différence entre FFS et NFS-DL. Dans le cas de FFS, il est toujours possible de n'avoir aucune colonne à ajouter en choisissant des corps de fonctions avec une seule place à l'infini, et même lorsque de tels corps de fonctions ne sont pas utilisés et qu'il est nécessaire d'ajouter des colonnes pour prendre en compte la valuation des places à l'infini, la taille de leurs coefficients est similaire à la taille des coefficients du reste de la matrice, contrairement à l'étape d'algèbre linéaire pour NFS-DL où les coefficients des colonnes ajoutées sont compris dans l'intervalle $[0, \ell - 1]$. À part cette différence, le reste de l'étape d'algèbre linéaire est identique pour FFS et NFS-DL.

L'étape de logarithme individuel est aussi similaire entre NFS-DL et FFS. L'objectif est d'exprimer le logarithme de l'élément cible en fonction de logarithmes virtuels d'idéaux au-dessus de polynômes irréductibles et unitaires dont le degré est de plus en plus petit, jusqu'à pouvoir l'exprimer uniquement en fonction de logarithmes virtuels d'idéaux des bases de facteurs.

Remarque 4.4.2. Le temps de calcul de l'étape de logarithme individuel est négligeable pour l'algorithme FFS et cette étape n'a donc pas besoin d'être beaucoup optimisée. Cependant, pour les nouveaux algorithmes découverts en 2013 pour le calcul de logarithme discret dans les corps finis de petite caractéristique, dont l'algorithme quasi-polynomial [15], l'étape de logarithme individuel devient l'étape critique et a donc été énormément étudiée et améliorée.

La complexité de l'algorithme FFS, pour un choix optimal des différents paramètres, est $\mathrm{L}_{p^n}\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$. La constante dans la complexité de FFS est meilleure que celle de NFS-DL et identique à celle de SNFS. Cela est dû au fait que n'importe quel polynôme irréductible de $\mathbb{F}_p[t]$ de degré n peut être utilisé pour représenter $\mathbb{F}_{p^n}$, ce qui permet d'obtenir des polynômes f_1 et f_2 avec des coefficients de petit degré en t et donc des normes dont le degré en t est plus petit, ce qui augmente les probabilités de friabilité.

4.5 Calcul de logarithme discret dans $\mathbb{F}_{2^{809}}$ avec FFS

Nous allons maintenant donner des détails sur le calcul, terminé en avril 2013, de logarithme discret avec FFS dans l'extension première $\mathbb{F}_{2^{809}}$ [12]. Ce calcul a été réalisé avec R. Barbulescu, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau et P. Zimmermann. Ce calcul était le record de calcul de logarithme discret dans une extension première de $\mathbb{F}_2$ jusqu'en octobre 2014 et le calcul de logarithme discret dans $\mathbb{F}_{2^{1279}}$ par Kleinjung [82]. Lorsque le calcul dans $\mathbb{F}_{2^{809}}$ a été terminé, nous avons entrepris le calcul de logarithme discret dans $\mathbb{F}_{2^{1039}}$, dont quelques temps de calcul seront donnés pour illustrer que les nouveaux algorithmes de calcul de logarithme discret en petite caractéristique découverts en 2013 sont non seulement asymptotiquement plus rapides mais le sont aussi en pratique pour les extensions premières dès un millier de bits environ.

Nous nous sommes intéressés au calcul de logarithme discret dans le sous-groupe d'ordre

premier de $\mathbb{F}_{2^{809}}^*$, correspondant au facteur premier de $2^{809}-1$ suivant :

$$\ell = 41483867312606056475251865474888423964616257742413275679781 37.$$

L'entier ℓ est un nombre premier de 202 bits, et l'autre facteur de $2^{809}-1$ est un nombre premier de 608 bits. Nous avons choisi ce facteur car la taille de ℓ correspond aux recommandations faites pour l'utilisation de corps finis pour des applications cryptographiques [115]. Le calcul dans l'autre sous-groupe de $\mathbb{F}_{2^{809}}^*$ aurait été identique jusqu'à l'étape d'algèbre linéaire, pour laquelle l'utilisation d'un module quatre fois plus grand aurait entraîné un temps de calcul environ quatre fois supérieur.

Remarque 4.5.1. Afin d'avoir une notation plus compacte, les polynômes de $\mathbb{F}_2[t]$ seront représentés par l'écriture hexadécimale de l'entier obtenu en considérant les polynômes sur $\mathbb{Z}$ et en les évaluant en $t = 2$. Par exemple, l'entier `0x11` représente le polynôme $t^4 + 1$ et l'entier `0xb00001` représente le polynôme $t^{23} + t^{21} + t^{20} + 1$.

La méthode décrite dans [9] a été utilisée pour la sélection polynomiale. En utilisant les critères de cet article, il est apparu que le meilleur choix était de prendre deux polynômes unitaires de degré $\deg(f_1) = 1$ et $\deg(f_2) = 6$. Pour trouver le polynôme f_2, nous avons calculé la grandeur α — dont la définition est similaire à celle dans le cas des polynômes à coefficients entiers des algorithmes NFS et NFS-DL — pour tous les polynômes irréductibles de degré 6 tels que le degré en t du coefficient de X^i est, au plus, $12-2i$. Des techniques de crible ont été utilisées pour accélérer ce calcul, qui a pris l'équivalent de 2760 heures sur un cœur d'un processeur Intel Core i5-2500 à 3,3 GHz. Ce calcul peut être considéré comme un précalcul, car le polynôme f_2 peut être utilisé avec différents polynômes f_1 pour des calculs de logarithme discret dans $\mathbb{F}_{2^n}$ pour n allant de 700 à 900 environ [48]. Le polynôme suivant a été utilisé :

$$f_2 = X^6 + \texttt{0x7}X^5 + \texttt{0x6b}X^3 + \texttt{0x1ab}X^2 + \texttt{0x326}X + \texttt{0x19b3}.$$

Avec ces paramètres, le degré en t du résultant de f_1 et f_2 est 6 fois le degré en t du coefficient constant de f_1. Comme nous voulons que ce résultant possède un facteur irréductible de degré 809, cela impose que le degré en t du coefficient constant de f_1 soit de degré au moins 135. Le choix de f_1 n'a pas beaucoup d'impact sur la suite des calculs et peut donc se faire très rapidement. Pour notre calcul, le polynôme suivant, qui est unitaire, de degré 1 et dont le coefficient constant est de degré 135 en t, a été utilisé :

$$f_1 = X + \texttt{0x80000000000000000000000001e7eaa}.$$

Le résultant $\mathrm{Res}(f_1, f_2)$ est égal à $(t + 1)$ fois le polynôme irréductible suivant :

$$\begin{aligned}\varphi = \ & \texttt{0x3fffffffffffffffffffffffffffffe8000000000000000000000}\\ & \texttt{0cc0cfaeb0e0000000000000000000000000004dfffffffffffc}\\ & \texttt{3c3ffc3c3c3fce3c2133fffffffffffe9697fe96804c84c97e0b0000c0}\\ & \texttt{0cf9b0f7675354e79f4cf7c97e29},\end{aligned}$$

dont le degré est 809 et qui est utilisé comme polynôme de définition pour $\mathbb{F}_{2^{809}}$.

La collecte des relations a été faite en utilisant l'implémentation décrite dans [49]. Les bornes de friabilité étaient identiques des deux côtés et valaient $B_1 = B_2 = 23$, c'est-à-dire que la base de facteurs du côté rationnel contenait tous les polynômes irréductibles et unitaires de $\mathbb{F}_2[t]$ de degré inférieur ou égal à 23 et la base de facteurs du côté algébrique contenait tous les idéaux

premiers non nuls de degré 1 au-dessus d'un polynôme irréductible et unitaire de $\mathbb{F}_2[t]$ de degré inférieur ou égal à 23. Des spécial-$\mathfrak{q}$ ont été utilisés du côté rationnel, car c'est le côté pour lequel le degré de la norme est le plus grand pour les valeurs des degrés de a et b considérées. Deux ensembles de relations, notés $\mathcal{R}_{27}$ et $\mathcal{R}_{28}$ dans la suite, ont été calculés pour étudier l'influence de certains paramètres sur l'étape de filtrage.

Pour l'ensemble de relations $\mathcal{R}_{27}$, nous avons autorisé 3 grands facteurs irréductibles jusqu'au degré 27 compris des deux côtés. Tous les polynômes irréductibles et unitaires de $\mathbb{F}_2[t]$ de degré entre 24 et 27 compris ont été utilisés comme spécial-$\mathfrak{q}$ pour le crible. Pour chaque spécial-$\mathfrak{q}$, 2^{30} paires (a, b) ont été considérées. Nous avons ainsi obtenu 52 millions de relations (non uniques). Les équivalents des temps de calcul pour un cœur d'un processeur Intel Core i5-2500 à 3,3 GHz sont donnés dans la table 4.1

deg $\mathfrak{q}$	nombre de rels	secondes/rel	rels/spécial-$\mathfrak{q}$	nombre de rels total	temps total
24	6 940 249	1,48	9,93	6 940 249	2853 h
25	9 926 294	1,91	7,39	16 866 543	8119 h
26	14 516 775	2,42	5,62	31 383 318	17 877 h
27	20 645 456	3,38	4,15	52 028 774	37 260 h

TABLE 4.1 – Données sur le calcul de l'ensemble de relations $\mathcal{R}_{27}$.

Pour l'ensemble de relations $\mathcal{R}_{28}$, nous avons autorisé 3 grands facteurs irréductibles jusqu'au degré 28 compris des deux côtés. Tous les polynômes irréductibles et unitaires de $\mathbb{F}_2[t]$ de degré entre 24 et 28 compris ont été utilisés comme spécial-$\mathfrak{q}$ pour le crible. Pour chaque spécial-$\mathfrak{q}$, 2^{28} paires (a, b) ont été considérées. Nous avons ainsi obtenu 117 millions de relations (non uniques). Les équivalents des temps de calcul pour un cœur d'un processeur Intel Core i5-2500 à 3,3 GHz sont donnés dans la table 4.2

deg $\mathfrak{q}$	nombre de rels	secondes/rel	rels/spécial-$\mathfrak{q}$	nombre de rels total	temps total
24	9 515 069	0,41	13,61	9 515 069	1083 h
25	13 816 908	0,54	10,29	23 331 977	3155 h
26	20 538 387	0,65	7,95	43 870 364	6863 h
27	29 652 781	0,86	5,96	73 523 145	13 946 h
28	43 875 232	1,07	4,57	117 398 377	26 986 h

TABLE 4.2 – Données sur le calcul de l'ensemble de relations $\mathcal{R}_{28}$.

Pour l'étape de filtrage, nous avons utilisé le code spécifique pour le calcul de logarithme discret disponible dans `cado-nfs`. Ce code est l'implémentation des algorithmes décrits dans le chapitre 5. L'ensemble de relations $\mathcal{R}_{27}$ contenait 52 028 774 relations, dont 30 142 422 relations uniques (42 % de doublons). La matrice obtenue à la fin de l'étape de filtrage avait 3,68 millions de lignes et de colonnes avec en moyenne 100 coefficients non nuls par ligne. L'ensemble de relations $\mathcal{R}_{28}$ contenait 117 398 377 relations, dont 67 411 816 relations uniques (43 % de doublons). La matrice obtenue à la fin de l'étape de filtrage avait 4,85 millions de lignes et de colonnes avec en moyenne 100 coefficients non nuls par ligne.

Remarque 4.5.2. La matrice utilisée lors de l'étape d'algèbre linéaire est celle obtenue en considérant les deux ensembles de relations. Si le calcul devait être refait de la façon la plus efficace possible, seul l'ensemble de relations $\mathcal{R}_{27}$ serait calculé et utilisé, mais pour notre calcul il aurait

été dommage de ne pas utiliser toutes les relations à notre disposition. Les deux ensembles de relations contenaient 78,8 millions de relations uniques et la matrice obtenue à la fin de l'étape de filtrage avait 3 602 667 lignes et colonnes avec en moyenne 100 coefficients non nuls par ligne. Les coefficients de la matrice obtenue sont tous inférieurs, en valeur absolue, à 50 et environ 90 % des coefficients non nuls sont égaux à ± 1. Des détails supplémentaires sont donnés dans la section 5.6.1.

Pour l'étape d'algèbre linéaire, le noyau dans $\mathbb{Z}/\ell\mathbb{Z}$ de la matrice obtenue après l'étape de filtrage a été calculé en utilisant l'implémentation de Jeljeli [70] des algorithmes de Wiedemann et Wiedemann par bloc sur des cartes graphiques NVIDIA. Cette implémentation utilise le système modulaire de représentation (ou RNS, pour *Residue Number System*) pour tirer parti du parallélisme offert par les cartes graphiques. Il n'a pas été nécessaire d'ajouter de colonnes à la matrice produite par l'étape de filtrage, car ce n'est jamais nécessaire du côté rationnel et le corps de fonctions K_2 du côté algébrique n'a qu'une seule place à l'infini.

À des fins de comparaison, le calcul du noyau a été fait de deux façons différentes. Le premier calcul a été réalisé à l'aide de l'algorithme de Wiedemann sur deux cartes graphiques NVIDIA GeForce GTX 680 situées sur une même machine. Le temps d'exécution a été de 18 jours. Le second calcul a été réalisé à l'aide de l'algorithme de Wiedemann par bloc sur huit cartes graphiques NVIDIA Tesla M2050 réparties sur quatre machines. Le temps d'exécution a été de 4,5 jours. Ces deux calculs ont aussi été comparés avec une implémentation de l'algorithme de Wiedemann par bloc sur 4 nœuds contenant un processeur Intel Core i5-2500 à 3,3 GHz connectés *via* un réseau Infiniband QDR, utilisant les jeux d'instructions SSE-4.2 et AVX et la représentation RNS. Le calcul n'a pas été mené jusqu'au bout mais nous avons estimé que le temps d'exécution du calcul complet aurait été de 68,4 jours, soit l'équivalent de 26 267 heures sur un seul processeur.

Une fois le noyau calculé, les logarithmes virtuels de 39 319 911 idéaux premiers non nuls de degré 1 étaient connus, dont 98,6 % des idéaux au-dessus de polynômes irréductibles et unitaires de degré 28.

L'étape de logarithme individuel a été faite en utilisant les mêmes méthodes que pour l'algorithme NFS-DL, en particulier l'utilisation de la technique de crible par spécial-$\mathfrak{q}$, dans le but d'exprimer le logarithme de l'élément cible en fonction des logarithmes calculés par l'étape d'algèbre linéaire. Le temps de calcul de cette étape est de moins d'une heure.

L'élément correspondant au polynôme t dans $\mathbb{F}_{2^{809}}$ est un générateur du sous-groupe d'ordre ℓ de $\mathbb{F}_{2^{809}}^*$, il est donc utilisé comme base pour exprimer les logarithmes. Le logarithme des éléments correspondant aux idéaux de la base de facteurs était donné par le résultat de l'étape d'algèbre linéaire, par exemple :

$$\log(t+1) \equiv 107082105171602535431582987436798986525914273094868488570257 4 \pmod{\ell}.$$

L'élément cible choisi pour l'étape de logarithme individuel est le polynôme dont l'écriture en hexadécimal *via* notre notation est RSA-1024, réduit modulo φ, ce qui donne

$$\begin{aligned} w = {} & \texttt{0x1f9a3921ce3a6242e78639ca23caf27e12a006365601c433c63981941} \\ & \texttt{f64d96e863383e119347a75bbaddf2e29e3957219e9f9417e390826915b} \\ & \texttt{07e2d0cae4130f2a6099563cbb36afc90cca828dab388b2096c0a72fe04} \\ & \texttt{8767c3d42902db899e1521f2ae1f}. \end{aligned}$$

Son logarithme est

$$\log(w) \equiv 29997870719116434854500200834208349771549879083381254164707 96 \pmod{\ell}.$$

Après avoir terminé le calcul pour $\mathbb{F}_{2^{809}}$, nous avons voulu nous attaquer au logarithme discret dans $\mathbb{F}_{2^{1039}}$. Après trois mois de calcul sur un cluster composé de 768 cœurs de processeurs Intel E5-2650 à 2,0 GHz, nous avons obtenu environ 2 milliards de relations non uniques, dont 37,7 % de doublons. À la fin de l'étape de filtrage, nous avons obtenu une matrice d'environ 60 millions de lignes et colonnes. Nous avons ensuite estimé que l'étape d'algèbre linéaire nécessiterait, en considérant le facteur de 265 bits de $2^{1039}-1$, environ 22 mois sur le même cluster que celui utilisé pour la collecte de relations. Le calcul n'a pas été effectué mais cela montre, qu'avec un peu plus de puissance de calcul, le calcul de logarithme discret avec FFS dans une extension première de $\mathbb{F}_2$ de plus d'un millier de bits est possible. Tous les algorithmes de calcul de logarithme discret découverts durant l'année 2013 tirent parti de l'existence de sous-corps pour parvenir à améliorer la complexité. Pour les utiliser dans le cas des extensions premières, il faut plonger l'extension première dans un corps plus grand. La limite pratique entre ces nouveaux algorithmes et l'algorithme FFS pour les extensions premières n'était donc pas claire. En 2014, Kleinjung a calculé un logarithme discret dans l'extension première $\mathbb{F}_{2^{1279}}$ [82] en utilisant ces nouveaux algorithmes. Le temps de calcul nécessaire est l'équivalent d'environ 3,5 années sur un cœur, soit plus de 30 fois moins que le temps nécessaire à l'étape de collecte des relations avec FFS pour $\mathbb{F}_{2^{1039}}$. Ceci montre bien que les nouveaux algorithmes pour le calcul de logarithme discret dans les corps finis de petite caractéristique découverts en 2013 sont non seulement asymptotiquement meilleurs que l'algorithme FFS pour des extensions premières mais sont aussi meilleurs en pratique pour des extensions premières de plus d'un millier de bits.

Chapitre 5

L'étape de filtrage des algorithmes de crible : NFS, NFS-DL et FFS

Les algorithmes de crible désignent, dans le cas du calcul de logarithme discret, les algorithmes basés sur la méthode de calcul d'indices, comme les algorithmes NFS-DL et FFS, et, dans le cas de la factorisation, les algorithmes combinant des relations pour produire des congruences de carrés, comme les algorithmes QS, MPQS et NFS. L'étape de filtrage des algorithmes de crible se situe entre l'étape de collecte des relations et l'étape d'algèbre linéaire. Dans les sections 5.1 et 5.2, nous étudierons l'étape de filtrage dans le cas général des algorithmes de crible. Une partie de l'étape de filtrage consiste à choisir des lignes «lourdes» dans une matrice. Dans la section 5.3, nous proposerons différentes méthodes pour évaluer le poids d'une ligne de la matrice et dans le reste du chapitre nous évaluerons quelle méthode est la plus adaptée dans le cas de NFS, NFS-DL et FFS.

5.1 Les objectifs de l'étape de filtrage

L'étape d'algèbre linéaire des algorithmes de crible calcule le noyau d'une matrice construite à partir des relations calculées précédemment. L'étape de crible produit une quantité importante de relations, de quelques milliers à plusieurs milliards pour les calculs records. Pour accélérer l'étape d'algèbre linéaire, il est nécessaire de réduire la taille de la matrice qui sera traitée, sans toutefois perdre l'information contenue dans l'ensemble des relations.

Pour illustrer l'importance de l'étape de filtrage et dans quelle proportion le filtrage permet de réduire la taille de la matrice, nous présentons des données issues de deux calculs records : la factorisation de RSA-768 et le calcul de logarithme discret dans un corps premier. Le record actuel de factorisation avec NFS est la factorisation de RSA-768 [83], un entier de 768 bits (232 chiffres décimaux). Au début de l'étape de filtrage, les dimensions de la matrice étaient de 48 milliards de lignes et 35 milliards de colonnes. À la fin du filtrage, les dimensions n'étaient plus que d'environ 193 millions de lignes et de colonnes, soit une réduction de plus de 99 %. Le record actuel de calcul de logarithme discret avec NFS-DL dans un corps premier a été réalisé pour un premier de 180 chiffres décimaux (voir chapitre 4). Au début de l'étape de filtrage, les dimensions de la matrice étaient de 175 millions de lignes et 78 millions de colonnes. À la fin du filtrage, les dimensions n'étaient plus que d'environ $7,28$ millions de lignes et de colonnes, soit une réduction de 96 % environ.

L'étape de filtrage diffère suivant qu'elle est utilisée dans un algorithme de factorisation ou

de calcul de logarithme discret. Dans le reste de la section, nous allons étudier les particularités du filtrage dans le cas de la factorisation et dans le cas du calcul de logarithme discret pour mettre en avant les différences et les points communs.

5.1.1 Le cas de la factorisation

Tous les algorithmes de factorisation basés sur la méthode de congruence de carrés ont une étape de filtrage similaire. L'algorithme génère des relations, qui sont des égalités entre objets (entiers, idéaux dans un corps de nombres, *etc*) dont la factorisation est connue dans une base de facteurs. Il faut ensuite trouver un sous-ensemble de ces relations qui, multipliées entre elles, donne une congruence de carrés. En remarquant qu'un carré est caractérisé par le fait que tous ses facteurs ont un exposant pair, le problème peut être reformulé en un problème d'algèbre linéaire modulo 2. Cela revient à calculer le noyau d'une matrice définie modulo 2, dont les coefficients correspondent à la parité des exposants des éléments de la base de facteurs dans chaque relation. Chaque vecteur du noyau, calculé lors de l'étape d'algèbre linéaire, fournit une congruence de carrés qui permet de trouver, avec une certaine probabilité, un facteur non trivial. Il faut donc s'assurer que le noyau ait une dimension suffisamment grande (entre 50 et 150 en pratique) pour obtenir un facteur non trivial avec une probabilité proche de 1. Dans le cas de l'algorithme NFS, une description plus précise du contexte est donnée dans le chapitre 2.

Le but du filtrage est donc, une fois la matrice construite à partir de relations, de diminuer le plus possible sa taille tout en s'assurant que le noyau ait une dimension suffisamment grande.

5.1.2 Le cas du calcul de logarithme discret

Comme nous l'avons vu dans le chapitre 4, dans le cas du calcul de logarithme discret, une relation est vue comme une égalité

$$\sum_i e_i \operatorname{vlog}(\mathfrak{p}_i) = 0, \tag{5.1}$$

où les $\mathfrak{p}_i$ sont des éléments de la base de facteurs et $\operatorname{vlog}(\mathfrak{p}_i)$ représente son logarithme virtuel. La gestion des unités, comme expliqué dans le chapitre 4, *via* le calul d'unités fondamentales, l'utilisation de fonctions de Schirokauer ou le calcul des valuations des places à l'infini, est faite après l'étape de filtrage, donc nous ne nous en occuperons pas dans ce chapitre et nous supposerons que l'égalité (5.1) est vérifiée pour chaque relation.

Calculer les logarithmes virtuels se ramène à un problème d'algèbre linéaire modulo ℓ, où ℓ est l'ordre du groupe dans lequel les logarithmes sont calculés (avec les standards de sécurité actuels, la taille de ℓ varie entre 160 et 500 bits environ [23]). Le module ℓ est généralement un nombre premier, mais rien lors de l'étape de filtrage n'impose qu'il le soit. Le calcul des logarithmes virtuels revient à calculer le noyau d'une matrice définie modulo ℓ dont les coefficients correspondent à la contribution des logarithmes virtuels des éléments de la base de facteurs dans chaque relation. Pour être sûr qu'une seule solution sera obtenue, il faut s'assurer que le nombre de relations est supérieur ou égal au nombre d'éléments dans la base de facteurs.

Le but du filtrage est donc, une fois la matrice construite à partir de relations, de diminuer le plus possible sa taille sans en modifier le noyau.

5.1.3 Points communs et différences

Dans le cas de la factorisation et du calcul de logarithme discret, l'objectif principal de l'étape de filtrage est de diminuer le plus possible les dimensions d'une matrice formée par les relations

calculées par les étapes précédentes de l'algorithme, pour pouvoir calculer son noyau.

Dans le cas de la factorisation, la matrice est définie modulo 2, alors que dans le cas du calcul de logarithme discret la matrice est définie modulo ℓ, pour un entier ℓ de quelques centaines de bits.

En faisant correspondre les relations aux lignes de la matrice et les éléments de la base de facteurs aux colonnes, nous remarquons que dans le cas de la factorisation il faut calculer le noyau à gauche, alors que dans le cas du logarithme discret il faut calculer le noyau à droite.

L'excès est défini comme la différence entre le nombre de lignes et le nombre de colonnes de la matrice. Au début de l'étape de filtrage, l'excès doit être positif. Dans le cas de la factorisation, nous pouvons remarquer que l'excès à la fin de l'étape de filtrage est une borne inférieure sur la dimension du noyau à gauche. Il faut donc s'assurer qu'à la fin de l'étape de filtrage l'excès soit au moins égal à la dimension souhaitée pour le noyau. Dans le cas du calcul de logarithme discret, pour ne pas modifier le noyau, il faut s'assurer que l'excès ne devienne pas négatif. Comme il n'est pas nécessaire d'avoir un excès strictement positif, il peut être plus facile d'avoir un excès nul à la fin de l'étape de filtrage pour avoir une matrice carrée dans l'étape d'algèbre linéaire.

De plus, par construction, les matrices au début de l'étape de filtrage sont très creuses (environ 25 coefficients non nuls par ligne) et lors de l'étape d'algèbre linéaire, des algorithmes spécifiquement conçus pour les matrices creuses sont utilisés pour pouvoir gérer le cas de matrices de plusieurs centaines de milliers, voire plusieurs millions de lignes et colonnes. Le coût de ces algorithmes dépend de la dimension de la matrice et du nombre moyen de coefficients non nuls par ligne. Il faut donc conserver le caractère creux de la matrice lors de l'étape de filtrage. En pratique, à la fin de l'étape de filtrage, les matrices ont entre 100 et 200 coefficients non nuls par ligne en moyenne.

Pour résumer, l'étape de filtrage effectue des opérations sur une matrice définie sur un corps fini, dont l'objectif est de réduire la taille de la matrice tout en gardant son caractère creux et en contrôlant son noyau. Dans la section suivante, nous décrirons les différentes opérations effectuées lors de l'étape de filtrage.

5.2 Description de l'étape de filtrage

L'état de l'art de l'étape de filtrage, dans le cas de la factorisation, est décrit dans la thèse de Cavallar [35, chap. 3]. La description donnée dans cette section est une adaptation de cet état de l'art pour donner une description de l'étape de filtrage unifiée des cas de la factorisation et du logarithme discret.

Différentes implémentations du filtrage pour la factorisation sont disponibles dans des outils de factorisation tels que `GGNFS` [59], `Msieve` [111] et `cado-nfs` [33]. Une description de l'étape de filtrage de `Msieve` peut être trouvée dans [120]. À notre connaissance, la seule implémentation disponible du filtrage dans le cas du logarithme discret est celle de `cado-nfs`, développée durant cette thèse.

L'étape de filtrage est généralement divisée en 4 sous-étapes. Le première sous-étape consiste à supprimer les relations qui apparaissent plus d'une fois. La présence ou non de doublons dépend de la méthode utilisée lors de l'étape de crible. Nous n'étudierons pas cette sous-étape dans ce chapitre et nous supposerons toujours que l'ensemble de relations au début de l'étape de filtrage est un ensemble de relations uniques. La deuxième sous-étape, étudiée dans la section 5.2.1, consiste à supprimer les singletons (les colonnes de la matrice avec un seul coefficient non nul) pour réduire la taille de la matrice. La troisième sous-étape, étudiée dans la section 5.2.2, réduit encore la taille de la matrice en supprimant des lignes «lourdes». Cette sous-étape diminue

l'excès de la matrice. Enfin, la dernière sous-étape, étudiée dans la section 5.2.3, est un début d'élimination de Gauss. Cette sous-étape réduit la taille de la matrice en combinant des lignes, mais rend la matrice moins creuse. Les deuxième et troisièmes sous-étapes de l'étape de filtrage consistent à supprimer des lignes et des colonnes dans la matrice et sont identiques pour la factorisation et le logarithme discret. En revanche, la dernière sous-étape, où des lignes sont combinées, dépend du fait que les coefficients sont considérés modulo 2 ou modulo ℓ.

Les deuxième et troisième sous-étapes de l'étape de filtrage sont principalement une *élimination de Gauss structurée* (SGE pour *structured gaussian elimination*) [88, 127]. L'élimination de Gauss structurée commence par supprimer des colonnes contenant uniquement des coefficients nuls, qui n'existe pas dans le cas du filtrage par construction de la matrice (voir ci-dessous). Ensuite, les colonnes avec un seul coefficient, ce qui correspond à la deuxième sous-étape du filtrage, et des lignes «lourdes» sont supprimées, ce qui correspond à la troisième sous-étape du filtrage. La différence entre l'élimination de Gauss structurée et l'étape de filtrage est l'existence de la quatrième sous-étape de l'étape de filtrage, la combinaison des relations, qui fait que la meilleure stratégie dans les deux cas peut être différente.

Avant de décrire en détails les différentes sous-étapes de l'étape de filtrage, nous allons expliquer comment la matrice est construite à partir des relations. Tout d'abord les relations uniques sont numérotées, ainsi que les éléments de la base de facteurs qui apparaissent dans au moins une relation. Ensuite le coefficient de coordonnée (i, j) dans la matrice correspond à la valuation du $j^{\text{ème}}$ élément de la base de facteurs dans la $i^{\text{ème}}$ relation. Dans le cas de la factorisation, les coefficients de la matrice sont réduits modulo 2. Dans le cas du logarithme discret, les valuations étant très petites par rapport à ℓ, il n'est pas nécessaire de réduire la matrice modulo ℓ. Il existe donc une correspondance entre les lignes de la matrice et les relations, ce qui nous amènera dans la suite à utiliser indifféremment les termes *relation* et *ligne*. Il existe aussi une correspondance entre les colonnes de la matrice et les éléments de la base de facteurs. Dans les cas auxquels nous allons nous intéresser (NFS, NFS-DL et FFS), les éléments de la base de facteurs sont des idéaux premiers dans un corps de nombres (NFS et NFS-DL) ou un corps de fonctions (FFS), donc nous utiliserons indifféremment les termes *idéal* et *colonne*.

Enfin, rappelons la définition de l'excès et donnons la définition du poids des colonnes et des lignes de la matrice.

Définition 5.2.1 (Excès)**.** L'excès est la différence entre le nombre de lignes et de colonnes de la matrice. De façon équivalente, c'est la différence entre le nombre de relations et le nombre d'éléments de la base de facteurs apparaissant dans au moins une relation. L'excès relatif est le rapport entre l'excès et le nombre de colonnes de la matrice.

Définition 5.2.2. Le poids d'une ligne (resp. d'une colonne) de la matrice est le nombre de coefficients non nuls dans la ligne (resp. la colonne). Par extension, nous parlerons du poids d'une relation ou d'un idéal.

Le poids total d'une matrice est le nombre total de coefficients non nuls de la matrice.

5.2.1 La suppression des singletons

Dans cette section, nous allons étudier la deuxième partie de l'étape de filtrage : la suppression des singletons. Commençons tout d'abord par donner la définition d'un singleton.

Définition 5.2.3 (Singleton)**.** Un singleton est une colonne de la matrice de poids 1.

Une colonne qui est un singleton et la ligne correspondant au seul coefficient non nul de la colonne peuvent être supprimées sans perte d'information. En effet, dans le cas de la factorisation,

une relation contenant un singleton ne pourra jamais contribuer à former un carré car la valuation de l'idéal correspondant à la colonne qui est un singleton sera toujours impaire. Le singleton peut donc être supprimé sans modifier le noyau à gauche de la matrice. Et dans le cas du logarithme discret, un singleton correspond à un idéal dont le logarithme virtuel peut être exprimé en fonction des logarithmes virtuels des autres idéaux apparaissant dans la relation. Dans ce cas, la relation peut être supprimée de la matrice, mais doit être mise de côté pour permettre, une fois l'étape d'algèbre linéaire terminée, le calcul du logarithme virtuel de l'idéal correspondant à la colonne qui était un singleton.

Lors de la suppression d'un singleton, une colonne et une ligne sont supprimées, donc l'excès n'est pas modifié (sauf dans le cas très rare où une ligne contient plus d'un singleton et où l'excès augmente).

Exemple 5.2.4. Considérons la matrice suivante, qui a un excès de 2 :

$$M_0 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

La première colonne est un singleton et peut donc être supprimée, ainsi que la deuxième ligne. En faisant cela, un nouveau singleton est créé : la deuxième colonne. La deuxième colonne et la première ligne peuvent donc aussi être supprimées. La matrice ainsi obtenue ne contient plus de singleton et a toujours un excès de 2 :

$$M_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

D'un point de vue de l'implémentation, la suppression des singletons est identique dans le cas de la factorisation et dans le cas du logarithme discret. En effet la valeur des coefficients n'a pas d'importance, seul la nullité ou non des coefficients compte. Le seul changement est que dans le cas du logarithme discret, il faut garder à la fin la liste des relations qui ont été supprimées, ce qui n'est pas nécessaire pour la factorisation.

Enfin, comme le montre l'exemple 5.2.4, supprimer un singleton peut en créer un ou plusieurs autres, il est donc nécessaire de parcourir plusieurs fois toutes les colonnes de la matrice pour s'assurer que tous les singletons ont bien été supprimés.

5.2.2 La suppression des cliques

Tant que l'excès est supérieur à ce qui est voulu, c'est-à-dire 0 pour le logarithme discret et entre 50 et 150 pour la factorisation, des lignes peuvent être supprimées de la matrice. Le choix des lignes à supprimer est primordial pour obtenir la matrice la plus petite et la plus creuse possible.

En particulier, s'il existe une colonne de poids 2 et que l'une des lignes correspondant à l'un des coefficients non nuls de cette colonne est supprimée, alors la colonne devient un singleton et sera supprimée (ainsi que l'autre ligne correspondant à l'autre coefficient non nul). Au final, deux lignes et une colonne auront été supprimées, réduisant l'excès de 1. Pour généraliser cette idée, nous allons définir ce qu'est une clique dans le contexte de l'étape de filtrage.

Définition 5.2.5 (Clique)**.** Considérons le graphe où les nœuds sont les lignes et les arêtes sont les colonnes de poids 2, connectant les deux lignes correspondant aux coefficients non nuls de la colonne. Une clique est une composante connexe de ce graphe.

Remarque 5.2.6. Bien qu'une composante connexe ne soit pas une clique dans le sens de la théorie des graphes, dans le contexte de l'étape de filtrage c'est le terme qui s'est imposé (voir, par exemple, [35, p. 52] et [83, p. 339]), nous avons donc gardé cette terminologie dans cette thèse.

Exemple 5.2.7. La matrice M_1 de l'exemple 5.2.4 contient 3 cliques : une contenant 3 lignes (la première, la deuxième et la troisième), une contenant deux lignes (la quatrième et la cinquième) et une contenant seulement la sixième ligne. Le graphe obtenu à partir de cette matrice est représenté dans la figure 5.1.

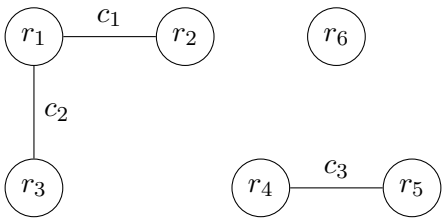

FIGURE 5.1 – Un graphe avec 3 cliques, construit à partir de la matrice M_1 de l'exemple 5.2.4 (r_i correspond à la $i^{\text{ème}}$ ligne et c_j à la $j^{\text{ème}}$ colonne).

Supprimer n'importe laquelle de ces trois cliques réduit l'excès de 1. Supprimer la clique contenant trois lignes connecte les deux autres cliques *via* la dernière colonne qui devient une colonne de poids 2.

Si n'importe quelle ligne d'une clique est supprimée, alors cela va créer par définition un singleton, qui à son tour lorsqu'il sera supprimé, créera un autre singleton, jusqu'à ce que toutes les lignes appartenant à la clique soient supprimées. Donc supprimer n'importe quelle ligne d'une clique conduit à supprimer toutes les lignes formant cette clique, ainsi que toutes les colonnes de poids 2 correspondant aux arêtes de la clique. Pour comprendre comment évolue l'excès lorsqu'une clique est supprimée, il faut distinguer deux cas. Soit la clique est un graphe acyclique, alors comme elle est connexe, elle est formée d'exactement un nœud (c'est-à-dire une relation) de plus que d'arêtes (c'est-à-dire d'idéaux de poids 2) et supprimer la clique fait diminuer l'excès d'exactement 1. Soit la clique n'est pas acyclique, alors comme elle est connexe, elle est formée d'au moins autant de nœuds que d'idéaux de poids 2 et supprimer la clique laisse l'excès inchangé ou l'augmente. Le cas où la clique n'est pas acyclique est le plus intéressant, car cela permet de diminuer la taille et le poids total de la matrice sans diminuer l'excès, mais malheureusement, en pratique dans les cas auxquels nous nous intéresserons (NFS, NFS-DL et FFS), une clique est presque toujours acyclique et dans la suite nous considérerons qu'une clique contient toujours exactement une relation de plus que d'idéaux de poids 2.

La suppression des cliques consiste, une fois tous les singletons supprimés, à calculer l'ensemble des cliques et à en choisir certaines qui seront supprimées. Comme n'importe quelle clique peut être supprimée, le choix des cliques supprimées a des conséquences importantes sur la taille et la densité de la matrice obtenue à la fin de l'étape de filtrage. Lors de la suppression des cliques, un poids est donné à chaque clique et les plus lourdes sont supprimées, par rapport à cette fonction de poids. Le choix de la fonction de poids est crucial et sera discuté dans la section 5.3.

Du point de vue de l'implémentation, si la fonction de poids utilisée ne dépend pas de la valeur des coefficients, alors seule la nullité ou non des coefficients compte. Et, comme pour la suppression des singletons, la seule différence, entre les cas de la factorisation et du logarithme discret provient du fait qu'il faut garder la trace des relations supprimées dans le cas du logarithme discret. Lorsque des cliques sont supprimées, cela peut créer de nouveaux singletons et il peut alors être nécessaire de refaire une sous-étape de suppression des singletons, une fois qu'un nombre significatif de cliques a été supprimé.

Finalement, remarquons que tout algorithme de suppression de relations peut être écrit comme un algorithme de suppression de cliques en utilisant une fonction de poids appropriée et en effectuant une suppression des singletons après la suppression des relations. En effet, chaque relation appartient à une clique (qui peut être une clique contenant une seule relation) et, comme vu précédemment, supprimer une relation d'une clique entraîne la création de singletons, qui lorsqu'ils sont supprimés revient à supprimer toutes les relations appartenant à la clique. Par exemple, la stratégie consistant à supprimer la relation la plus lourde de la matrice peut être vue comme une suppression de cliques où la fonction de poids pour les cliques est le maximum des poids des relations appartenant à la clique.

5.2.3 Début d'élimination de Gauss : la combinaison des relations

La combinaison des relations est un début d'élimination de Gauss, certaines lignes vont être combinées pour faire apparaître des singletons qui pourront être supprimés. Jusqu'à présent, toutes les méthodes que nous avons décrites consistaient à supprimer des lignes et des colonnes, et diminuaient donc les dimensions de la matrice ainsi que son poids. Le but de la combinaison des relations est encore de diminuer les dimensions de la matrice mais cela se fera au détriment du poids total de la matrice, qui augmente.

Nous allons dans un premier temps décrire la combinaison des relations dans le cas de la factorisation pour ensuite voir comment cela peut être adapté au cas du logarithme discret.

La cas de la factorisation

Rappelons que dans le cas de la factorisation, la matrice est définie modulo 2 et donc le résultat de l'addition de deux coefficients non nuls est un coefficient nul.

Pour comprendre l'idée sous-jacente à la combinaison des relations, commençons par des exemples. Soit $\mathfrak{p}$ un idéal de poids 2 et r_1 et r_2 les deux lignes de la matrice correspondant aux deux coefficients non nuls de $\mathfrak{p}$. Si la ligne r_1 est remplacée par $r_1 + r_2$, alors la colonne correspondant à l'idéal $\mathfrak{p}$ devient un singleton et peut être supprimée, ainsi que la ligne r_2 qui est la seule ligne restante contenant l'idéal $\mathfrak{p}$. L'idée est que si la relation r_1 (resp. r_2) est utilisée pour former un carré, comme la valuation de $\mathfrak{p}$ doit être paire et que la seule autre relation avec une valuation impaire pour $\mathfrak{p}$ est la relation r_2 (resp. r_1), elle doit aussi être utilisée. Cette opération est appelée un 2-merge. Un 2-merge permet de supprimer une ligne et une colonne de la matrice, et donc ne change pas l'excès.

Si nous considérons maintenant que l'idéal $\mathfrak{p}$ est de poids 3, et que r_1, r_2 et r_3 sont les lignes correspondant aux 3 coefficients non nuls de l'idéal $\mathfrak{p}$, alors comme dans l'exemple précédent, si la ligne r_1 est remplacée par $r_1 + r_3$, et la ligne r_2 par $r_2 + r_3$, la colonne correspondant à $\mathfrak{p}$ devient un singleton et peut être supprimée, ainsi que la ligne r_3. Cette opération est appelée un 3-merge. Un 3-merge supprime une ligne (dans ce cas r_3) et une colonne (celle correspondant à $\mathfrak{p}$) de la matrice, et donc ne change pas l'excès. Une différence importante avec un 2-merge est qu'il existe plus d'une façon de combiner les 3 relations. En effet, nous aurions pu choisir d'ajouter r_2 à r_1 et r_3, et de supprimer r_2, ou d'ajouter r_1 à r_2 et r_3, et de supprimer r_1.

Exemple 5.2.8. Considérons la matrice M_0 du début de l'exemple 5.2.4. La deuxième colonne est de de poids 2, un 2-merge peut donc être effectué. Si la deuxième ligne est ajoutée à la première ligne, alors la deuxième colonne devient un singleton et la deuxième ligne et la deuxième colonne peuvent être supprimées.

Un exemple de 3-merge est donné par la troisième colonne qui est de poids 3. L'objectif étant de minimiser le poids total de la matrice, il est préférable d'ajouter la quatrième ligne à la première et à la troisième lignes.

La définition suivante permet de généraliser la notion de 2-merge et 3-merge vue dans les exemples précédents.

Définition 5.2.9 (k-merge)**.** Soit k un entier supérieur ou égal à 2. Soit $\mathfrak{p}$ un idéal de poids k et $r_1, \ldots, r_k$ les k lignes correspondant aux k coefficients non nuls de $\mathfrak{p}$. Un k-merge est une façon d'effectuer de manière successive des additions de lignes de la matrice de la forme $r_i \leftarrow r_i + r_j$, avec $i \neq j$ et $1 \leqslant i, j \leqslant k$, de sorte que la colonne correspondant à $\mathfrak{p}$ devienne un singleton, qui sera supprimé, ainsi que la seule ligne restante contenant un coefficient non nul dans cette colonne.

Cette définition pourrait être généralisée à $k = 1$, en remarquant qu'effectuer un 1-merge correspond à supprimer un singleton. Mais dans la suite, nous ne considérerons que les k-merges avec $k \geqslant 2$, car la suppression de tous les singletons est effectuée avant la combinaison des relations.

Effectuer un k-merge permet de supprimer une ligne et une colonne, et laisse donc l'excès inchangé (à part dans les très rares cas où l'excès peut augmenter si effectuer le k-merge permet de supprimer plus d'une colonne). Effectuer un 2-merge réduit toujours le poids total de la matrice. Si deux lignes r_1 et r_2 ont pour poids respectifs w_1 et w_2, alors le poids de $r_1 + r_2$ est au plus $w_1 + w_2 - 2$ (les 2 coefficients non nuls de la colonne de poids 2 qui a permis le 2-merge ont été supprimés). Donc un 2-merge permet de réduire le poids total de la matrice d'au moins 2. En général, pour un k-merge ($k > 2$) le poids total de la matrice augmente. Pour contrôler l'augmentation du poids total de la matrice, il faut choisir une façon de combiner les k lignes de façon à minimiser cette augmentation. Une manière de faire cela est de se représenter les k lignes de la matrice comme des nœuds d'un graphe complet (c'est-à-dire où tous les nœuds sont connectés entre eux) et pondéré. Le poids d'une arête de ce graphe est donné par le poids de l'addition des deux lignes représentées par les nœuds que cette arête relie. Trouver le meilleur moyen de combiner les k lignes est équivalent à trouver un arbre couvrant de poids minimal de ce graphe. L'exemple 5.2.10 et la figure 5.2 illustrent cette méthode.

Exemple 5.2.10. Exemple d'un 6-merge provenant de la sixième colonne de la matrice M_0 de l'exemple 5.2.4. Le tableau à gauche de la figure 5.2 liste les poids de la combinaison des lignes r_i et r_j, $i < j$. L'arbre à droite de la figure 5.2 est un arbre couvrant de poids minimal associé à ces poids. Les arêtes de cet arbre relient deux nœuds qui seront combinés pour ce 6-merge.

	r_2	r_4	r_5	r_7	r_8
r_1	4	2	4	2	3
r_2		4	2	4	3
r_4			2	2	1
r_5				2	1
r_7					1

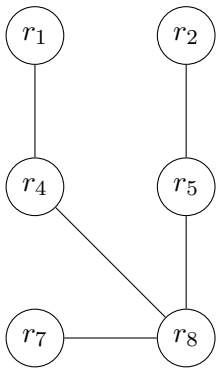

FIGURE 5.2 – Poids des combinaisons des relations et arbre couvrant de poids minimal associé.

Un k-merge effectué à l'aide d'un calcul d'arbre couvrant de poids minimum ne considère pas toutes les combinaisons possibles des k lignes. En effet, après un k-merge utilisant un calcul d'arbre couvrant, les $k-1$ lignes restantes sont la combinaison d'exactement deux lignes initiales. Or, il est aussi possible que les lignes finales soient la combinaison de quatre, six, huit,... lignes de départ. En effet, dans l'exemple 5.2.10, le poids de la combinaison des lignes 1, 4, 7 et 8 est plus petit que le poids de la combinaison des lignes 1 et 4, et donc cette combinaison devrait être utilisée à la place de celle suggérée par l'arbre couvrant. Mais, en pratique, la matrice est très différente de celle de l'exemple 5.2.10 (il n'y a pas, par exemple, de ligne ne contenant qu'un coefficient non nul comme la ligne 8) et donc combiner plus de deux lignes produit toujours des lignes plus lourdes.

Adaptation au cas du logarithme discret

D'un point de vue théorique, il y a peu de changement à faire pour adapter la combinaison des relations au cas du logarithme discret. Remarquons tout d'abord que, les coefficients n'étant plus définis modulo 2, si $\mathfrak{p}$ est un idéal et r_1 et r_2 sont deux lignes dont les coefficients dans la colonne associée à $\mathfrak{p}$ sont non nuls, alors le coefficient de $\mathfrak{p}$ n'est pas nécessairement nul dans la ligne obtenue par la combinaison $r_1 + r_2$. Pour arriver à annuler le coefficient de $\mathfrak{p}$, il faut effectuer une combinaison de la forme $\alpha_1 r_1 + \alpha_2 r_2$ où α_1 et α_2 sont des entiers non nuls calculés en fonction de la valeur des coefficients de $\mathfrak{p}$ dans les lignes r_1 et r_2. La valeur des coefficients n'intervient pas dans le poids des lignes ou des colonnes, seule la nullité ou non d'un coefficient est prise en compte, et donc la valeur des coefficients n'intervient que lors du calcul des entiers α_1 et α_2. Une fois ce changement pris en compte, le reste de l'algorithme décrit dans le cas de la factorisation reste valable dans le cas du logarithme discret.

Remarques sur l'implémentation

L'algorithme de Markowitz [100] est utilisé pour choisir quel est le prochain k-merge qui sera effectué. Tous les k-merges possibles sont dans un tas (avec $2 \leqslant k \leqslant k_{\max}$) triés par l'augmentation du poids total de la matrice correspondant à ce k-merge. Le k-merge correspondant à l'augmentation la plus faible du poids total est effectué, ensuite le tas est mis à jour si nécessaire. Le coût du calcul exact de tous les arbres couvrants de poids minimum pour tous les k-merges considérés étant trop important, seule une approximation de l'augmentation de poids engendrée par un k-merge est calculée. L'approximation se fait de la façon suivante : la ligne la plus légère est ajoutée au $k-1$ autres lignes et les annulations autres que celles de la colonne pour laquelle

le k-merge est effectuée ne sont pas prises en compte. Cela signifie que pour un k-merge dont le poids de la ligne la plus légère est noté ω, l'augmentation du poids total de la matrice est approchée par $(\omega - 2)(k - 2) - 2$. L'arbre couvrant de poids minimum n'est calculé qu'au moment où le k-merge est effectué. La combinaison des relations faisant augmenter le poids total de la matrice, l'algorithme s'arrête dès que la densité de la matrice dépasse une certaine borne.

Ensuite, à la fin de la combinaison des relations, une ligne de la matrice ne correspond plus à une relation mais à un ensemble de relations. Pour les algorithmes de calcul de logarithme discret, comme NFS et NFS-DL, pour pouvoir corriger l'erreur faite par l'omission des unités dans les relations, il est nécessaire de garder pour chaque ligne de la matrice l'ensemble des relations qui entrent dans la construction de cette ligne. Dans le cas de l'algorithme NFS, cela est aussi nécessaire pour le calcul des caractères. De plus, comme pour la suppression des singletons et la suppression des cliques, il faut, dans le cas du logarithme discret, garder la trace des lignes qui ont été supprimées pour pouvoir, une fois l'étape d'algèbre linéaire terminée, calculer le logarithme virtuel des colonnes qui ont été supprimées.

Enfin, dans le cas du logarithme discret, les coefficients de la matrice sont définis modulo ℓ et lors de la combinaison des relations, le calcul des nouveaux coefficients devrait s'effectuer modulo ℓ. Mais en pratique, il est préférable de considérer ces éléments comme des entiers relatifs. En effet, les données expérimentales montrent que les valeurs initiales des coefficients de la matrice sont très petites devant ℓ, et qu'à la fin de la combinaison des relations, plus de 99,5 % des coefficients sont compris entre -10 et 10, et plus de 90 % sont égaux à ± 1.

5.3 Les fonctions de poids lors de la suppression des cliques

Le problème que nous allons maintenant étudier est le suivant : quelle fonction de poids doit être utilisée lors la suppression de cliques (voir section 5.2.2) pour produire la meilleure matrice à la fin de l'étape de filtrage ? Le calcul du noyau de la matrice lors de l'étape d'algèbre linéaire est effectué grâce à des algorithmes spécifiques pour les matrices creuses (voir la discussion de la section 2.5). Ces algorithmes ont une complexité en temps qui, en première approximation, est proportionnelle au produit du nombre de lignes de la matrice par son poids total. Si le nombre moyen de coefficients non nuls par ligne est fixé, alors le but de l'étape de filtrage est de produire la matrice la plus petite possible. Le problème que nous cherchons à résoudre est donc le suivant : étant donné un ensemble de relations uniques, une valeur pour l'excès final et une valeur pour le nombre moyen de coefficients non nuls par ligne pour la matrice finale, quelle est la fonction de poids qui doit être utilisée lors de la suppression des cliques pour produire la matrice la plus petite possible à la fin de l'étape de filtrage ?

Ce que nous cherchons à améliorer est la taille de la matrice *à la fin de l'étape de filtrage*. Cela signifie que la fonction de poids utilisée lors de la suppression des cliques doit produire la matrice la plus petite possible *après la combinaison des relations* et pas nécessairement *après la suppression des cliques*. Pour arriver à cela, une bonne fonction de poids doit donc non seulement supprimer les relations lourdes mais aussi créer beaucoup de 2-merges qui réduiront aussi le poids total de la matrice finale.

Avant de définir les différentes fonctions de poids que nous allons étudier, il est nécessaire de définir quelques notations. La taille de la matrice sera notée N, et son poids total W. Soit c une clique, le nombre de relations contenues dans la clique sera noté $n(c)$. Les $n(c)$ relations sont liées par $n(c) - 1$ idéaux de poids 2 (cf. section 5.2.2). Si $\mathfrak{p}$ est un idéal, le poids de la colonne correspondant à cet idéal sera noté $w(\mathfrak{p})$. De plus l'algorithme de suppression des cliques suppose que tous les singletons ont été supprimés.

5.3.1 Les fonctions de poids étudiées

Les fonctions de poids que nous nous proposons d'étudier dans la suite auront toutes deux termes. Le premier terme mesurera la contribution des idéaux de poids plus grand ou égal à 3 et servira à supprimer les cliques contenant beaucoup d'idéaux et à favoriser la création de colonnes de faible poids pour encourager la création de singletons ou de 2-merges (ou de k-merges avec k petit). Le second terme mesurera la contribution des idéaux de poids 2, c'est-à-dire des idéaux qui lient les relations de la clique et servira à supprimer les grandes cliques.

Pour créer de nouveaux singletons ou de nouveaux 2-merges, la valeur associée par la fonction de poids à une colonne apparaissant dans la clique doit être une fonction décroissante de son poids. Dans la suite, sept fonctions $\Lambda_0, \ldots, \Lambda_6$ seront étudiées, elles ont toutes la forme suivante :

$$\Lambda_i = \sum_{\mathfrak{p} \in c, w(\mathfrak{p}) \geqslant 3} \lambda_i(w(\mathfrak{p})),$$

où la somme est sur tous les idéaux apparaissant dans une relation de la clique avec un poids supérieur ou égal à 3. Un idéal peut contribuer plusieurs fois, s'il appartient à plus d'une relation de la clique. Les expressions pour les λ_i sont :

$$\lambda_0(w) = 1, \quad \lambda_1(w) = \left(\frac{2}{3}\right)^{w-2}, \quad \lambda_2(w) = \left(\frac{1}{2}\right)^{w-2}, \quad \lambda_3(w) = \left(\frac{4}{5}\right)^{w-2},$$
$$\lambda_4(w) = \frac{1}{\log_2(w)}, \quad \lambda_5(w) = \frac{2}{w} \quad \text{et} \quad \lambda_6(w) = \frac{4}{w^2}\,.$$

Remarquons que pour toutes les fonctions $\lambda_0, \ldots, \lambda_6$, la valeur pour $w = 2$ est 1. Le cas où la contribution des idéaux de poids plus grand ou égal à 3 est nulle est considéré plus tard.

Les idéaux de poids 2 étant les idéaux qui lient les relations appartenant à la clique, le nombre d'idéaux de poids 2 est lié au nombre de relations de la clique. La contribution des idéaux de poids 2 de la clique est donc une fonction dépendante de $n(c)$, le nombre de relations formant la clique. Les 4 fonctions suivantes ont été considérées :

$$\nu_0(c) = 0, \quad \nu_1(c) = \frac{n(c)}{4}, \quad \nu_2(c) = \frac{n(c)}{2} \quad \text{et} \quad \nu_3(c) = n(c).$$

En utilisant les 7 formules pour Λ et les 4 formules pour ν, il est possible de construire 28 fonctions de poids différentes, $\Omega_{xy}(c) = \Lambda_x(c) + \nu_y(c)$. Par exemple,

$$\Omega_{23}(c) = \Lambda_2(c) + \nu_3(c) = \sum_{\mathfrak{p} \in c, w(\mathfrak{p}) \geqslant 3} \frac{1}{2^{w(\mathfrak{p})-2}} + n(c).$$

En plus de ces 28 fonctions de poids, 3 autres seront considérées, elles seront notées Ω_{s0}, Ω_{s1} et Ω_{s2}. Les relations de la clique sont liées grâce à des idéaux de poids 2 qui peuvent être utilisés pour effectuer des 2-merges. Pour Ω_{s0}, le poids de la clique est le poids de la relation restante une fois que tous les 2-merges possibles, correspondant aux idéaux de poids 2 de la clique, ont été effectués. Ce poids est exactement le nombre d'idéaux qui apparaît un nombre impair de fois dans la clique, donc

$$\Omega_{s0}(c) = \sum_{w(\mathfrak{p}) \geqslant 3} \frac{1 - (-1)^{w_c(\mathfrak{p})}}{2},$$

où $w_c(p)$ compte le nombre de relations de la clique c qui contiennent l'idéal $\mathfrak{p}$. Dans le cas où $w_c(\mathfrak{p}) = 0$ ou $w_c(\mathfrak{p}) = 1$ lorsque $w(\mathfrak{p}) \geqslant 3$ (ce qui est presque toujours le cas pour les "grands" idéaux), cette fonction de poids est similaire à Ω_{00}.

La fonction de poids Ω_{s1} correspond au cas où seulement le nombre de relations dans la clique est considéré :

$$\Omega_{s1}(c) = \nu_3(c) = n(c).$$

L'expression de la fonction de poids Ω_{s2} est inspirée par la dérivée logarithmique du produit $N \times W$, qui est la quantité que nous voulons minimiser :

$$\Omega_{s2}(c) = \frac{n(c)}{N} + \frac{\Lambda_0(c)}{W}.$$

Remarque 5.3.1. Dans le cas du calcul de logarithme discret, aucune des fonctions de poids décrites ne dépend de la valeur des coefficients, elles dépendent seulement du fait que les coefficients sont nuls ou non.

5.3.2 Les fonctions de poids utilisées dans les différentes implémentations disponibles

Dans sa thèse, Cavallar propose une fonction de poids [35] :

> « *The metric being used weighs the contribution from the small prime ideals by adding* 1 *for each relation in the clique and* 0.5 *for each free relation. The large prime ideals which occur more than twice in the relation table contribute* 0.5^{f-2} *where* f *is the prime ideal's frequency.* »

Comme les relations gratuites (« *free relations* ») représentent un très faible pourcentage du nombre total de relations (par exemple, moins de $0,01\%$ pour RSA-768), la fonction décrite par Cavallar peut être considérée comme étant la même que la fonction Ω_{23}.

La fonction de poids utilisée dans `Msieve` 1.50 est la fonction de poids décrite par Cavallar dans sa thèse, où seulement les idéaux de poids inférieurs ou égaux à 15 sont considérés. En pratique, cela donne les mêmes résultats. La fonction de poids utilisée dans l'outil de factorisation du CWI [47] est aussi la fonction décrite par Cavallar.

Dans `GGNFS` 0.77.1, la fonction de poids utilisée est la taille de la mémoire nécessaire pour stocker toutes les relations de la clique, ce qui correspond à une constante près à Ω_{03}.

Dans `cado-nfs` 1.0 et `cado-nfs` 1.1, la fonction de poids par défaut est, à une constante près, Ω_{s1}. La fonction de poids Ω_{s2} est aussi disponible dans le code source de `cado-nfs` 1.0 et `cado-nfs` 1.1 comme alternative.

Suite aux différentes versions de notre article [25] sur lequel s'appuie ce chapitre, les fonctions de poids utilisées par `Msieve` et `cado-nfs` ont été changées. Dans les versions 1.51 et 1.52 de `Msieve`, la fonction de poids Ω_{50} est utilisée. Dans `cado-nfs` 2.0 et `cado-nfs` 2.1, la fonction de poids par défaut est Ω_{31}.

5.3.3 Description des expériences

Les 31 fonctions de poids décrites dans cette section ont été implémentées dans `cado-nfs`. Les programmes utilisés sont `purge`, chargé de la suppression des singletons et la suppression des cliques, et `merge`, chargé de la combinaison des relations (calcul des k-merges). Les 31 fonctions de poids seront testées sur des relations provenant de différents calculs de factorisation et de logarithme discret, avec les mêmes paramètres et le même environnement.

La fonction de poids est utilisée lors de la suppression des cliques, c'est-à-dire pendant le programme `purge`, mais la qualité d'une fonction de poids sera jugée à la fin de l'étape de filtrage, c'est-à-dire après `merge`. La quantité utilisée pour évaluer les fonctions de poids est le

produit $N \times W$ à la fin de l'étape de filtrage, c'est-à-dire après `merge`, car ce produit permet de comparer, en première approximation, le temps d'exécution de l'étape d'algèbre linéaire. Pour les expériences, le ratio final W/N sera fixé, donc plus la matrice finale sera petite, meilleure sera la fonction de poids. Dans la suite, lors des comparaisons des fonctions de poids, une fonction de poids sera dite meilleure de $x\%$ lorsque le produit final $N \times W$ sera inférieur de $x\%$, et pas seulement la taille de la matrice finale.

Le temps d'exécution de l'étape de filtrage ne sera pas un critère pour évaluer la qualité d'une fonction de poids. En effet, le temps passé dans l'étape de filtrage par rapport à l'étape de crible ou d'algèbre linéaire est négligeable pour NFS, NFS-DL et FFS et les différences de temps dues aux différentes fonctions de poids sont faibles, le temps d'exécution n'est donc pas un critère pertinent.

Le programme `purge` de la version de développement[6] de `cado-nfs` utilisé pour ces expériences effectue 50 phases de suppression des singletons et de suppression des cliques. À chaque phase de suppression des singletons, tous les singletons sont supprimés. À chaque phase de suppression des cliques, l'excès est diminué par $1/50^{\text{ième}}$ de la différence entre l'excès initial et l'excès final désiré. Comme `cado-nfs` 1.1, `Msieve` 1.50 et `GGNFS` 0.77.1 combinent différemment les phases de suppression de singletons et de suppression de cliques, l'implémentation dans `cado-nfs` de leurs fonctions de poids (respectivement Ω_{s1}, Ω_{23} et Ω_{03}) permet d'avoir une comparaison fiable où seulement les fonctions de poids changent.

Les résultats des expériences sur les 31 fonctions de poids sont donnés dans la section 5.4, pour le cas de NFS, dans la section 5.5 pour le cas de NFS-DL et dans la section 5.6 pour le cas de FFS. Dans la section 5.7, nous étudierons l'influence de l'excès sur la matrice finale pour différentes fonctions de poids.

5.4 Comparaison des fonctions de poids avec NFS

Pour comparer les fonctions de poids décrites dans la section 5.3 dans le cas de l'étape de filtrage pour NFS, nous considérons les ensembles de relations issues de trois factorisations : celle de RSA-155, de B200 et de RSA-704.

5.4.1 Expériences : RSA-155, B200 et RSA-704

RSA-155 est un entier de 155 chiffres décimaux (512 bits) factorisé en 1999 [36] en utilisant NFS. Il faisait partie des challenges RSA. B200 est le $200^{\text{ième}}$ nombre de Bernoulli. Lors de la factorisation de son numérateur, un entier de 204 chiffres décimaux (677 bits) a été factorisé à l'aide de NFS en août 2012 [69]. RSA-704 est un entier de 212 chiffres décimaux (704 bits) factorisé en juillet 2012 [7] en utilisant `cado-nfs`. Il faisait partie des challenges RSA [87]. Pour RSA-704 et B200, les relations provenant des calculs de factorisation originaux ont été utilisées. Pour RSA-155, des relations ont été recalculées avec `cado-nfs`. Les paramètres utilisés et les données relatives aux trois expériences de comparaison de fonctions de poids avec NFS sont présentés dans la table 5.1.

5.4.2 Résultats

Dans cette section, uniquement des résultats partiels sont présentés, les résultats complets sont donnés à la fin du chapitre. Seuls les résultats pour les fonctions de poids suivantes sont

6. C'est également vrai pour les versions 2.0 et 2.1 de `cado-nfs`.

		RSA-155	B200	RSA-704
Avant `purge`	Nombre de relations	97,0M	702M	833M
	Nombre d'idéaux	89,7M	667M	745M
Après la première suppression des singletons	Nombre de relations	62,8M	453M	650M
	Nombre d'idéaux	51,9M	385M	552M
	Excès	10,8M	67,4M	98,5M
	Excès relatif	20,8 %	17,5 %	17,9 %
Après `purge`	Excès	160	160	160
Après `merge`	k-merge pour k de	2 à 30	2 à 30	2 à 50
	W/N	100	120	187

TABLE 5.1 – Données et paramètres pour les trois expériences de comparaison avec NFS.

donnés ici : les deux meilleures, la plus mauvaise, celle qui donne la plus petite matrice après `purge`, et celles de Cavallar (Ω_{23}), de `cado-nfs` 1.1 (Ω_{s1}) et de `GGNFS` 0.77.1 (Ω_{03}), si elles ne sont pas déjà présentes dans la table. Les résumés des résultats pour RSA-155, B200 et RSA-704 sont donnés, respectivement, dans les tables 5.2, 5.3 et 5.4. Les données complètes sont présentées, respectivement, dans les tables 5.12, 5.13 et 5.14 à la fin du chapitre.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	11	20 801 141	6 576 610	$4{,}33 \times 10^{15}$	
($2^{\text{ième}}$ meilleure)	21	20 812 812	6 586 358	$4{,}34 \times 10^{15}$	+0,30 %
(meilleure après purge)	23	20 300 078	6 689 249	$4{,}47 \times 10^{15}$	+3,45 %
	03	20 737 766	6 841 292	$4{,}68 \times 10^{15}$	+8,21 %
(plus mauvaise)	s1	20 560 066	6 961 546	$4{,}85 \times 10^{15}$	+12,05 %

TABLE 5.2 – Résultats partiels pour RSA-155 ; pour les résultats complets, voir table 5.12 à la fin du chapitre.

Remarquons tout d'abord que dans les trois cas, la meilleure fonction de poids pour l'étape de filtrage est Ω_{11}. Dans les trois cas, Ω_{11} produit une matrice environ 3,5 % meilleure que la matrice produite par Ω_{23}, la fonction de poids de Cavallar, de 6 % à 8 % meilleure que la matrice produite par Ω_{03}, la fonction de poids de `GGNFS` 0.77.1, et de 11 % à 13 % meilleure que la matrice produite par Ω_{s1}, la fonction de poids de `cado-nfs` 1.1. De plus, cette dernière est toujours celle qui produit la plus mauvaise matrice.

Ces trois expériences montrent que la meilleure fonction de poids Ω_{11} pour l'étape de filtrage complète n'est pas celle qui produit la plus petite matrice à la fin de `purge`, qui est dans ces trois expériences celle de Cavallar Ω_{23}.

Les matrices produites avec Ω_{00} et Ω_{s0} sont presque identiques, ce qui confirme le fait que presque toujours pour un "grand" idéal de poids plus grand ou égal à 3, s'il apparaît dans une clique, alors il n'y apparaît qu'une seule fois.

Les meilleures fonctions de poids sont celles pour lesquelles la contribution du nombre de relations dans la clique au poids de la clique est faible ou nulle (par exemple Ω_{11}, Ω_{21}, Ω_{10}, Ω_{30}). Cela signifie que le nombre de relations dans la clique devrait avoir une contribution faible au

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	11	154 087 021	44 202 262	$2{,}34 \times 10^{17}$	
($2^{\text{ième}}$ meilleure)	21	153 346 744	44 237 278	$2{,}35 \times 10^{17}$	+0,16 %
(meilleure après purge)	23	150 426 986	45 029 911	$2{,}43 \times 10^{17}$	+3,78 %
	03	153 261 960	45 536 067	$2{,}49 \times 10^{17}$	+6,13 %
(plus mauvaise)	s1	154 029 290	47 030 544	$2{,}65 \times 10^{17}$	+13,21 %

TABLE 5.3 – Résultats partiels pour B200 ; pour les résultats complets, voir table 5.13 à la fin du chapitre.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	11	312 721 893	81 239 661	$1{,}23 \times 10^{18}$	
($2^{\text{ième}}$ meilleure)	10	323 519 043	81 263 553	$1{,}23 \times 10^{18}$	+0,06 %
(meilleure après purge)	23	305 215 508	82 669 415	$1{,}28 \times 10^{18}$	+3,55 %
	03	312 257 402	83 835 119	$1{,}31 \times 10^{18}$	+6,49 %
(plus mauvaise)	s1	307 415 189	85 540 465	$1{,}37 \times 10^{18}$	+10,87 %

TABLE 5.4 – Résultats partiels pour RSA-704 ; pour les résultats complets, voir table 5.14 à la fin du chapitre.

poids de la clique.

5.5 Comparaison des fonctions de poids avec NFS-DL

Pour comparer les fonctions de poids décrites dans la section 5.3 dans le cas de l'étape de filtrage pour NFS-DL, nous considérons les ensembles de relations issues de deux calculs de logarithme discret dans $\mathbb{F}_p$: l'un avec un premier p de 155 chiffres décimaux et l'autre avec un premier p de 180 chiffres décimaux.

5.5.1 Expériences : p155 et p180

Le calcul de logarithme discret dans $\mathbb{F}_p$, avec un premier p de 180 chiffres décimaux (596 bits) est, au moment où cette thèse est écrite, le record de calcul de logarithme discret dans $\mathbb{F}_p$, avec p premier [26]. Plus de détails sont donnés dans le chapitre 4. Les relations utilisées pour l'expérience sont celles utilisées lors du calcul original. Dans la suite, ce nombre premier sera appelé p180. Le calcul de logarithme discret dans $\mathbb{F}_p$, avec un premier p de 155 chiffres est un calcul effectué pour tester le logiciel `cado-nfs` en prévision du calcul dans $\mathbb{F}_p$ avec l'entier p180. L'entier p de 155 chiffres (512 bits) qui a été utilisé est le plus petit entier p supérieur à RSA-155 tel que p et $(p-1)/2$ sont premiers. Dans la suite, ce nombre premier sera appelé p155.

Les paramètres utilisés et les données relatives aux deux expériences de comparaison de fonctions de poids avec NFS-DL sont présentés dans la table 5.5.

		p155	p180
Avant `purge`	Nombre de relations	26,9M	175M
	Nombre d'idéaux	14,9M	78M
Après la première suppression des singletons	Nombre de relations	26,0M	171M
	Nombre d'idéaux	14,0M	78M
	Excès	12M	93M
	Excès relatif	85,6 %	119 %
Après `purge`	Excès	5	5
Après `merge`	k-merge pour k de	2 à 25	2 à 30
	W/N	100	150

TABLE 5.5 – Données et paramètres pour les deux expériences de comparaison avec NFS-DL.

5.5.2 Résultats

Comme dans la section précédente, seuls des résultats partiels sont présentés ici, les résultats complets sont donnés à la fin du chapitre. Les résumés des résultats pour p155 et p180 sont donnés, respectivement, dans les tables 5.6 et 5.7. Les données complètes sont présentées, respectivement, dans les tables 5.15 et 5.16 à la fin du chapitre.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	30	6 714 244	2 536 409	$6{,}43 \times 10^{14}$	
($2^{\text{ième}}$ meilleure)	31	6 463 195	2 537 567	$6{,}44 \times 10^{14}$	+0,09 %
(meilleure après purge)	63	6 061 559	2 723 351	$7{,}42 \times 10^{14}$	+15,28 %
	23	6 066 632	2 711 534	$7{,}35 \times 10^{14}$	+14,29 %
	03	6 885 122	2 841 000	$8{,}07 \times 10^{14}$	+25,46 %
(plus mauvaise)	s1	7 659 426	3 396 493	$1{,}15 \times 10^{15}$	+79,32 %

TABLE 5.6 – Résultats partiels pour p155 ; pour les résultats complets, voir table 5.15 à la fin du chapitre.

Tout d'abord remarquons que les différences entre les fonctions de poids sont plus grandes que dans le cas de NFS. Comme le montre la section 5.7, cela peut être expliqué par le fait que l'excès relatif initial est plus grand ; de 85 % à 119 % pour les expériences avec NFS-DL contre 17 % à 21 % pour les expériences avec NFS.

Pour p180, la meilleure fonction de poids est Ω_{31}, pour p155, la meilleure est Ω_{30} (avec Ω_{31} en deuxième, 0,09 % moins bonne). La fonction de poids Ω_{31} est environ 15 % meilleure que Ω_{23}, de 25 % à 58 % meilleure que Ω_{03} et de 79 % à 118 % meilleure que Ω_{s1}. La meilleure fonction de poids après `purge` est Ω_{23} pour p180 et Ω_{63} pour p155 (avec Ω_{23} en seconde meilleure). Pour ces deux fonctions de poids la matrice produite à la fin de l'étape de filtrage est environ 15 % moins bonne que la meilleure fonction de poids globale.

Le fait que la meilleure fonction de poids après `purge` ne soit pas la meilleure fonction de poids après l'étape de filtrage complète et le fait que les meilleures fonctions de poids soient celles avec peu ou pas de contribution du nombre de relations dans les cliques se vérifient encore dans

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	31	21 468 306	7 288 100	$7{,}97 \times 10^{15}$	
($2^{\text{ième}}$ meilleure)	30	22 280 496	7 301 643	$8{,}00 \times 10^{15}$	+0,37 %
(meilleure après purge)	63	20 394 199	7 861 739	$9{,}27 \times 10^{15}$	+16,36 %
	23	20 395 070	7 866 604	$9{,}28 \times 10^{15}$	+16,51 %
	03	25 676 095	9 163 369	$1{,}26 \times 10^{16}$	+58,08 %
(plus mauvaise)	s1	28 940 807	10 769 526	$1{,}74 \times 10^{16}$	+118,36 %

TABLE 5.7 – Résultats partiels pour p180 ; pour les résultats complets, voir table 5.16 à la fin du chapitre.

le cas de NFS-DL.

5.6 Comparaison des fonctions de poids avec FFS

Pour comparer les fonctions de poids décrites dans la section 5.3 dans le cas de l'étape de filtrage pour FFS, nous considérons les ensembles de relations issues de trois calculs de logarithme discret : dans $\mathbb{F}_{2^{619}}$, $\mathbb{F}_{2^{809}}$ et $\mathbb{F}_{2^{1039}}$.

5.6.1 Expériences : $\mathbb{F}_{2^{619}}$, $\mathbb{F}_{2^{809}}$ et $\mathbb{F}_{2^{1039}}$

Les calculs de logarithme discret dans $\mathbb{F}_{2^{619}}$ en 2013 et dans $\mathbb{F}_{2^{809}}$ en 2014 étaient, au moment respectif où ils ont été terminés, le record de calcul de logarithme discret dans $\mathbb{F}_{2^p}$, avec p premier. Quelques détails sur le calcul dans $\mathbb{F}_{2^{619}}$ peuvent être trouvés dans [11]. Une description complète du calcul dans $\mathbb{F}_{2^{809}}$ est donnée dans le chapitre 4 et dans [12]. Le calcul dans $\mathbb{F}_{2^{1039}}$ a été entrepris après le calcul dans $\mathbb{F}_{2^{809}}$, mais n'a jamais pu être terminé, l'étape d'algèbre linéaire demandant trop de temps. Il a ensuite été abandonné, car en octobre 2014, Kleinjung a utilisé les nouveaux algorithmes découverts récemment (voir chapitre 4) pour effectuer un calcul de logarithme discret dans $\mathbb{F}_{2^{1279}}$ [82].

Les relations utilisées dans ces trois expériences ont été calculées avec l'outil de crible pour FFS disponible dans la version de développement de `cado-nfs` (cf. [49] pour plus de détails sur son implémentation). Pour $\mathbb{F}_{2^{619}}$ et $\mathbb{F}_{2^{809}}$, ce sont les relations utilisées lors des calculs originaux qui sont utilisées.

Les paramètres utilisés et les données relatives aux deux expériences de comparaison de fonctions de poids avec FFS sont présentés dans la table 5.8.

5.6.2 Résultats

Comme dans les sections précédentes, seuls des résultats partiels sont présentés ici, les résultats complets sont donnés à la fin du chapitre. Les résumés des résultats pour $\mathbb{F}_{2^{619}}$, $\mathbb{F}_{2^{809}}$ et $\mathbb{F}_{2^{1039}}$ sont donnés, respectivement, dans les tables 5.9, 5.10 et 5.11. Les données complètes sont présentées, respectivement, dans les tables 5.17, 5.18 et 5.19 à la fin du chapitre.

Tout d'abord remarquons que, comme pour NFS-DL, les différences entre les fonctions de poids sont plus grandes que dans le cas de NFS. Comme pour NFS-DL, cela peut être expliqué par le fait que l'excès relatif initial est plus grand ; de 45 % à 111 % pour les expériences avec FFS

$\mathbb{F}_{2^p}$ avec $p =$		619	809	1039
Avant `purge`	Nombre de relations	20,5M	81,0M	1306M
	Nombre d'idéaux	10,5M	39,4M	986M
Après la première suppression des singletons	Nombre de relations	19,7M	79,1M	1080M
	Nombre d'idéaux	9,7M	37,5M	746M
	Excès	10M	41,6M	334M
	Excès relatif	103 %	111 %	44,7 %
Après `purge`	Excès	0	0	0
Après `merge`	k-merge pour k de	2 à 30	2 à 30	2 à 30
	W/N	100	100	100

TABLE 5.8 – Données et paramètres pour les trois expériences de comparaison avec FFS.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	31	1 978 752	648 476	$4{,}21 \times 10^{13}$	
($2^{\text{ième}}$ meilleure)	30	2 083 125	650 626	$4{,}23 \times 10^{13}$	+0,66 %
(meilleure après purge)	62	1 893 132	671 975	$4{,}52 \times 10^{13}$	+7,38 %
	23	1 934 876	718 629	$5{,}16 \times 10^{13}$	+22,81 %
	03	2 483 373	887 541	$7{,}88 \times 10^{13}$	+87,32 %
(plus mauvaise)	s1	2 548 952	949 242	$9{,}01 \times 10^{13}$	+114,27 %

TABLE 5.9 – Résultats partiels pour $\mathbb{F}_{2^{619}}$; pour les résultats complets, voir table 5.17 à la fin du chapitre.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	31	9 644 786	3 484 569	$1{,}21 \times 10^{15}$	
($2^{\text{ième}}$ meilleure)	30	10 126 436	3 493 773	$1{,}22 \times 10^{15}$	+0,53 %
(meilleure après purge)	62	9 153 959	3 596 069	$1{,}29 \times 10^{15}$	+6,50 %
	23	9 226 205	3 822 940	$1{,}46 \times 10^{15}$	+20,36 %
	03	12 363 963	4 651 334	$2{,}16 \times 10^{15}$	+78,18 %
(plus mauvaise)	s1	22 190 421	9 075 847	$8{,}24 \times 10^{15}$	+578,38 %

TABLE 5.10 – Résultats partiels pour $\mathbb{F}_{2^{809}}$; pour les résultats complets, voir table 5.18 à la fin du chapitre.

		Après `purge`	Après `merge`		
	Ω	N	N	$N \times W$	
(meilleure)	11	188 580 425	65 138 845	$4{,}24 \times 10^{17}$	
($2^{\text{ième}}$ meilleure)	61	185 399 885	65 501 515	$4{,}29 \times 10^{17}$	+1,12 %
(meilleure après purge)	23	182 939 672	67 603 362	$4{,}57 \times 10^{17}$	+7,71 %
	03	197 703 703	74 570 015	$5{,}56 \times 10^{17}$	+31,05 %
(plus mauvaise)	s1	203 255 785	78 239 129	$6{,}12 \times 10^{17}$	+44,27 %

TABLE 5.11 – Résultats partiels pour $\mathbb{F}_{2^{1039}}$; pour les résultats complets, voir table 5.19 à la fin du chapitre.

contre 17 % à 21 % pour les expériences avec NFS. Pour les calculs dans $\mathbb{F}_{2^{1039}}$, les résultats sont similaires aux résultats dans le cas de NFS car l'excès relatif est similaire. Pour les calculs dans $\mathbb{F}_{2^{619}}$ et $\mathbb{F}_{2^{809}}$, l'excès relatif est très grand (plus de 100 %), les résultats sont donc différents et les différences plus marquées. Pour les calculs dans $\mathbb{F}_{2^{619}}$ et $\mathbb{F}_{2^{809}}$, la meilleure fonction de poids est Ω_{31}, qui est de 20 % à 23 % meilleure que Ω_{23}, de 78 % à 87 % meilleure que Ω_{03} et de 115 % à 578 % meilleure que Ω_{s1}. Le fait que Ω_{31} est la meilleure fonction de poids quand l'excès relatif est élevé est cohérent avec les résultats de la section 5.7. La meilleure fonction de poids après `purge` n'est pas Ω_{23} mais Ω_{62}. Cela est probablement dû à l'excès élevé, comme le montrent les expériences de la section 5.7 qui suggèrent que cela peut aussi être le cas pour NFS avec plus d'excès. La fonction de poids Ω_{23} produit toujours des petites matrices après `purge` mais n'est pas toujours compétitive après l'ensemble de l'étape de filtrage.

Le fait que la meilleure fonction de poids après `purge` ne soit pas la meilleure fonction de poids après l'étape de filtrage complète et le fait que les meilleures fonctions de poids soient celles avec peu ou pas de contribution du nombre de relations dans les cliques se vérifient encore dans le cas de FFS.

5.7 Influence de l'excès initial sur l'efficacité du filtrage

Dans cette section nous allons étudier l'effet de l'excès initial sur la taille de la matrice finale pour différentes fonctions de poids. Les calculs ont été effectués pour RSA-155 avec les mêmes paramètres que ceux décrits dans la section 5.4 sur 104 ensembles de relations initiales (entre 77,5 et 232 millions de relations uniques). Les résultats sont regroupés dans la figure 5.3, où la valeur du produit $N \times W$ de la matrice finale et l'excès relatif sont donnés, en fonction du nombre de relations uniques initiales, pour les fonctions de poids suivantes : Ω_{31} et Ω_{11} (les deux meilleures), Ω_{23} (Cavallar), Ω_{s1} (`cado-nfs` 1.1), Ω_{03} (`GGNFS` 0.77.1) et Ω_{62}.

La figure 5.3 montre que la meilleure fonction de poids peut changer avec l'excès. En effet lorsque l'excès relatif est plus petit que 63 % pour cette expérience, Ω_{11} est meilleure que Ω_{31}, alors que le contraire est vrai lorsque l'excès relatif est plus grand que 63%.

La figure 5.3 montre aussi que plus le nombre de relations initiales est important plus la différence entre les fonctions de poids augmente. Par exemple, la fonction de poids Ω_{11} est environ 0,2 % meilleure que Ω_{23} lorsque l'excès relatif vaut 0,6 % mais peut être jusqu'à 20 % meilleure quand l'excès relatif vaut 133 %. Cela montre que le choix des cliques à supprimer fait par la fonction de poids Ω_{11} est meilleur, car plus il y a d'excès (et donc de choix à faire), plus Ω_{11} est meilleure.

Enfin, la figure 5.3 montre que, dans la majorité des cas, plus le nombre de relations initiales

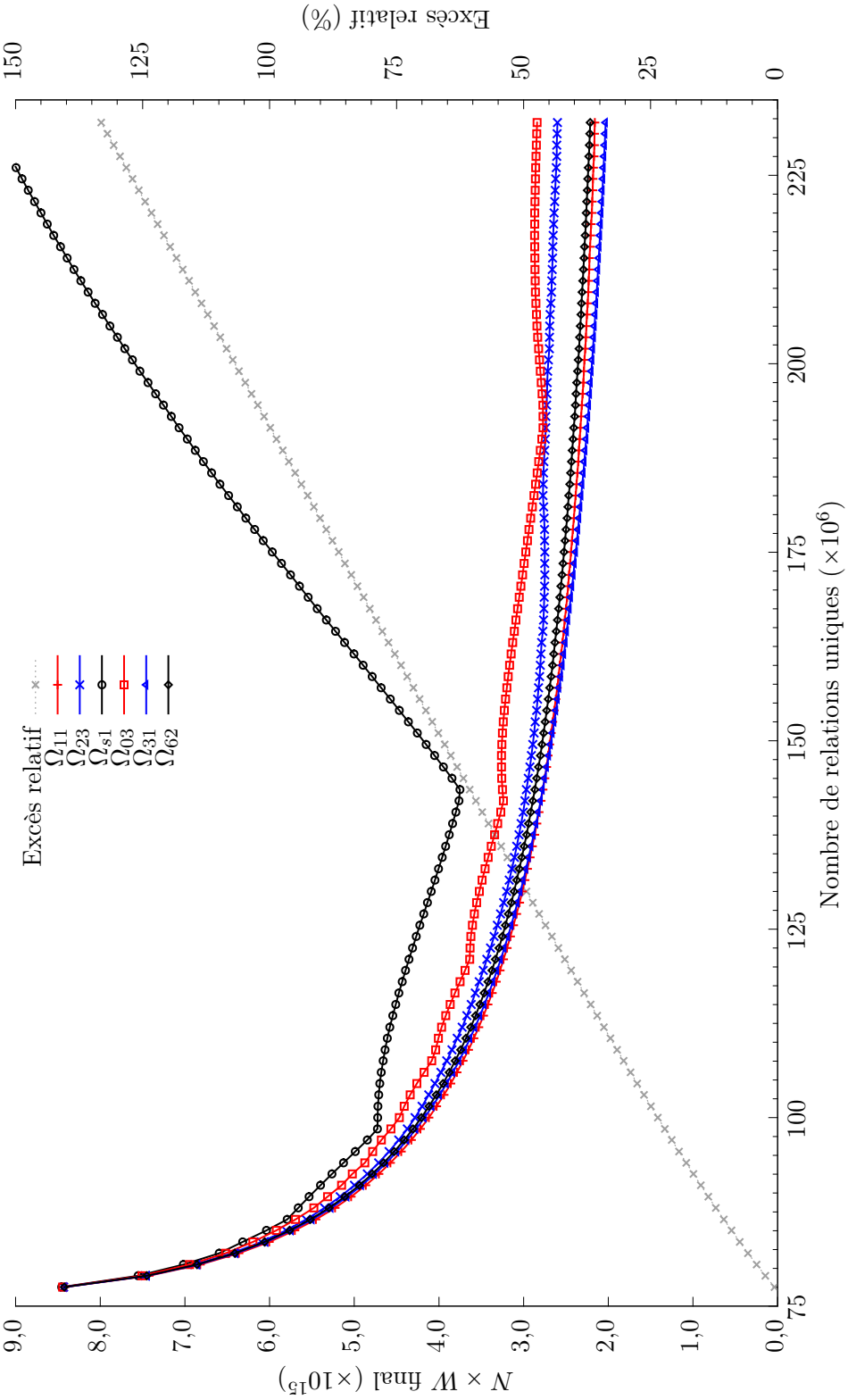

FIGURE 5.3 – Comparaisons des fonctions de poids en fonction de l'excès relatif.

est important, plus le produit $N \times W$ de la matrice finale est petit. La fonction de poids Ω_{s1} est un contre-exemple. En effet, lorsque l'excès relatif dépasse 60 %, le produit $N \times W$ augmente avec le nombre de relations initiales. Des calculs supplémentaires ont permis de montrer que cela est dû au fait que lorsque l'excès est trop important, des cliques formées d'une seule relation sont supprimées. Or la fonction de poids Ω_{s1} dépend uniquement de la taille de la clique et donc toutes les cliques formées d'une seule relation ont le même poids et le choix des cliques à supprimer parmi celles-ci est fait au hasard. Cela peut entraîner la suppression de relation légère ce qui peut augmenter le poids des k-merges.

5.8 Conclusion et perspectives

Dans ce chapitre, nous avons étudié l'étape de filtrage et décrit comment l'adapter dans le contexte du calcul de logarithme discret. Nous avons ensuite proposé plusieurs fonctions de poids pour l'étape de suppression des cliques, et nous les avons comparées entre elles et avec des fonctions de poids déjà décrites dans la littérature, en utilisant `cado-nfs`, avec les mêmes paramètres. Ces comparaisons ont été effectuées pour 3 ensembles de relations provenant de calcul de factorisation avec NFS, pour 2 ensembles de relations provenant de calcul de logarithme discret avec NFS-DL et pour 3 ensembles de relations provenant de calcul de logarithme avec FFS. Ces comparaisons nous ont permis de montrer que les meilleures fonctions de poids étaient celles dont la contribution des idéaux de poids 2, ou de façon équivalente du nombre de relations dans la clique, était faible ou nulle. Nous avons aussi montré que la meilleure fonction de poids pour `purge` (la suppression des singletons et des cliques, là où la fonction de poids est utilisée) ne donne pas la meilleure fonction de poids pour l'étape de filtrage. De plus des nouvelles fonctions de poids, meilleures que celles précédemment décrites dans la littérature, ont été identifiées, ce qui résulte en un gain de temps dans l'étape d'algèbre linéaire suivant l'étape de filtrage. Les expériences de comparaisons dans le cas de NFS, NFS-DL et FFS ont permis d'identifier Ω_{11} et Ω_{31} comme les meilleures fonctions de poids. L'expérience sur l'influence de l'excès initial a permis de montrer que Ω_{11} est la meilleure lorsque l'excès relatif initial est faible, tandis que Ω_{31} est la meilleure lorsque l'excès relatif initial est élevé.

Des expériences similaires pourraient être menées pour étudier dans quel ordre les k-merges devraient être effectués. Actuellement, seule la différence de poids total due au k-merge est prise en compte lors du `merge`, peut-être faudrait-il aussi considérer le poids des idéaux impliqués dans le k-merge. Il serait peut-être aussi possible d'entreprendre une analyse statistique de l'étape de filtrage, similaire à celle de [16] pour l'élimination de Gauss structurée. Cette analyse s'appuierait sur le fait qu'un idéal $\mathfrak{p}$ au-dessus d'un premier p apparaît avec valuation e dans une relation avec une probabilité $1/p^e$ (en considérant que tout ce qui touche aux relations se comporte comme des entiers aléatoires).

Résultats complets des comparaisons des sections 5.4, 5.5 et 5.6

L'annexe contient les résultats complets des expériences de comparaison des fonctions de poids. Les résultats complets sont donnés dans la table 5.12 pour RSA-155, dans la table 5.13 pour B200, dans la table 5.14 pour RSA-704, dans la table 5.15 pour p155, dans la table 5.16 pour p180, dans la table 5.17 pour $\mathbb{F}_{2^{619}}$, dans la table 5.18 pour $\mathbb{F}_{2^{809}}$ et dans la table 5.19 pour $\mathbb{F}_{2^{1039}}$.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
11	20 801 141	$8{,}71 \times 10^{15}$	6 576 610	$4{,}33 \times 10^{15}$	
21	20 812 812	$8{,}71 \times 10^{15}$	6 586 358	$4{,}34 \times 10^{15}$	+0,30 %
61	20 591 172	$8{,}53 \times 10^{15}$	6 592 609	$4{,}35 \times 10^{15}$	+0,49 %
10	21 677 978	$9{,}46 \times 10^{15}$	6 597 020	$4{,}35 \times 10^{15}$	+0,62 %
30	21 066 248	$8{,}93 \times 10^{15}$	6 598 208	$4{,}35 \times 10^{15}$	+0,66 %
12	20 525 758	$8{,}48 \times 10^{15}$	6 605 260	$4{,}36 \times 10^{15}$	+0,87 %
50	20 940 957	$8{,}83 \times 10^{15}$	6 607 958	$4{,}37 \times 10^{15}$	+0,96 %
60	21 672 863	$9{,}45 \times 10^{15}$	6 611 637	$4{,}37 \times 10^{15}$	+1,07 %
31	20 732 159	$8{,}65 \times 10^{15}$	6 614 847	$4{,}38 \times 10^{15}$	+1,17 %
22	20 437 176	$8{,}40 \times 10^{15}$	6 616 157	$4{,}38 \times 10^{15}$	+1,21 %
51	20 632 230	$8{,}57 \times 10^{15}$	6 630 136	$4{,}40 \times 10^{15}$	+1,63 %
32	20 570 710	$8{,}52 \times 10^{15}$	6 638 018	$4{,}41 \times 10^{15}$	+1,88 %
62	20 377 654	$8{,}35 \times 10^{15}$	6 640 471	$4{,}41 \times 10^{15}$	+1,95 %
40	20 855 869	$8{,}76 \times 10^{15}$	6 657 267	$4{,}43 \times 10^{15}$	+2,47 %
52	20 493 661	$8{,}45 \times 10^{15}$	6 659 152	$4{,}43 \times 10^{15}$	+2,53 %
13	20 361 903	$8{,}34 \times 10^{15}$	6 668 262	$4{,}45 \times 10^{15}$	+2,81 %
41	20 628 073	$8{,}57 \times 10^{15}$	6 676 570	$4{,}46 \times 10^{15}$	+3,06 %
33	20 433 585	$8{,}40 \times 10^{15}$	6 686 196	$4{,}47 \times 10^{15}$	+3,36 %
23	20 300 078	$8{,}29 \times 10^{15}$	6 689 249	$4{,}47 \times 10^{15}$	+3,45 %
42	20 514 904	$8{,}47 \times 10^{15}$	6 701 793	$4{,}49 \times 10^{15}$	+3,84 %
63	20 321 565	$8{,}30 \times 10^{15}$	6 712 215	$4{,}51 \times 10^{15}$	+4,17 %
53	20 400 561	$8{,}37 \times 10^{15}$	6 713 693	$4{,}51 \times 10^{15}$	+4,21 %
43	20 434 660	$8{,}40 \times 10^{15}$	6 750 781	$4{,}56 \times 10^{15}$	+5,37 %
20	22 876 961	$1{,}05 \times 10^{16}$	6 751 559	$4{,}56 \times 10^{15}$	+5,39 %
02	20 858 120	$8{,}76 \times 10^{15}$	6 818 390	$4{,}65 \times 10^{15}$	+7,49 %
01	20 860 148	$8{,}76 \times 10^{15}$	6 818 411	$4{,}65 \times 10^{15}$	+7,49 %
s0	21 048 884	$8{,}92 \times 10^{15}$	6 831 805	$4{,}67 \times 10^{15}$	+7,91 %
00	21 048 891	$8{,}92 \times 10^{15}$	6 831 875	$4{,}67 \times 10^{15}$	+7,91 %
03	20 737 766	$8{,}65 \times 10^{15}$	6 841 292	$4{,}68 \times 10^{15}$	+8,21 %
s2	20 496 848	$8{,}44 \times 10^{15}$	6 883 627	$4{,}74 \times 10^{15}$	+9,55 %
s1	20 560 066	$8{,}49 \times 10^{15}$	6 961 546	$4{,}85 \times 10^{15}$	+12,05 %

TABLE 5.12 – Données complètes pour RSA-155.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
11	154 087 021	$5{,}23 \times 10^{17}$	44 202 262	$2{,}34 \times 10^{17}$	
21	153 346 744	$5{,}18 \times 10^{17}$	44 237 278	$2{,}35 \times 10^{17}$	+0,16 %
61	152 480 272	$5{,}12 \times 10^{17}$	44 295 368	$2{,}35 \times 10^{17}$	+0,42 %
30	158 109 764	$5{,}51 \times 10^{17}$	44 344 210	$2{,}36 \times 10^{17}$	+0,64 %
10	161 480 932	$5{,}74 \times 10^{17}$	44 359 892	$2{,}36 \times 10^{17}$	+0,71 %
50	156 737 711	$5{,}41 \times 10^{17}$	44 359 936	$2{,}36 \times 10^{17}$	+0,71 %
31	154 254 136	$5{,}24 \times 10^{17}$	44 370 581	$2{,}36 \times 10^{17}$	+0,76 %
60	161 056 386	$5{,}71 \times 10^{17}$	44 386 107	$2{,}36 \times 10^{17}$	+0,83 %
12	151 985 472	$5{,}08 \times 10^{17}$	44 398 656	$2{,}37 \times 10^{17}$	+0,89 %
51	153 312 434	$5{,}18 \times 10^{17}$	44 455 926	$2{,}37 \times 10^{17}$	+1,15 %
22	151 171 217	$5{,}03 \times 10^{17}$	44 504 976	$2{,}38 \times 10^{17}$	+1,37 %
32	152 617 177	$5{,}13 \times 10^{17}$	44 519 799	$2{,}38 \times 10^{17}$	+1,44 %
40	155 854 758	$5{,}35 \times 10^{17}$	44 624 257	$2{,}39 \times 10^{17}$	+1,92 %
62	150 943 964	$5{,}01 \times 10^{17}$	44 631 783	$2{,}39 \times 10^{17}$	+1,95 %
52	151 973 891	$5{,}08 \times 10^{17}$	44 647 196	$2{,}39 \times 10^{17}$	+2,02 %
41	153 252 776	$5{,}17 \times 10^{17}$	44 718 043	$2{,}40 \times 10^{17}$	+2,35 %
13	150 807 090	$5{,}00 \times 10^{17}$	44 822 261	$2{,}41 \times 10^{17}$	+2,82 %
33	151 379 703	$5{,}04 \times 10^{17}$	44 835 865	$2{,}41 \times 10^{17}$	+2,89 %
42	152 135 118	$5{,}09 \times 10^{17}$	44 879 381	$2{,}42 \times 10^{17}$	+3,09 %
53	151 103 756	$5{,}02 \times 10^{17}$	44 990 483	$2{,}43 \times 10^{17}$	+3,60 %
23	150 426 986	$4{,}98 \times 10^{17}$	45 029 911	$2{,}43 \times 10^{17}$	+3,78 %
20	167 572 608	$6{,}18 \times 10^{17}$	45 145 579	$2{,}45 \times 10^{17}$	+4,31 %
43	151 365 155	$5{,}04 \times 10^{17}$	45 160 515	$2{,}45 \times 10^{17}$	+4,38 %
63	150 505 303	$4{,}98 \times 10^{17}$	45 168 479	$2{,}45 \times 10^{17}$	+4,42 %
02	154 067 402	$5{,}22 \times 10^{17}$	45 495 013	$2{,}48 \times 10^{17}$	+5,93 %
01	155 395 404	$5{,}32 \times 10^{17}$	45 505 293	$2{,}48 \times 10^{17}$	+5,98 %
03	153 261 960	$5{,}17 \times 10^{17}$	45 536 067	$2{,}49 \times 10^{17}$	+6,13 %
00	156 540 087	$5{,}40 \times 10^{17}$	45 550 198	$2{,}49 \times 10^{17}$	+6,19 %
s0	156 540 092	$5{,}40 \times 10^{17}$	45 550 331	$2{,}49 \times 10^{17}$	+6,19 %
s2	151 983 844	$5{,}08 \times 10^{17}$	45 897 498	$2{,}53 \times 10^{17}$	+7,82 %
s1	154 029 290	$5{,}21 \times 10^{17}$	47 030 544	$2{,}65 \times 10^{17}$	+13,21 %

TABLE 5.13 – Données complètes pour B200.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
11	312 721 893	$2{,}18 \times 10^{18}$	81 239 661	$1{,}23 \times 10^{18}$	
10	323 519 043	$2{,}34 \times 10^{18}$	81 263 553	$1{,}23 \times 10^{18}$	+0,06 %
21	312 473 810	$2{,}18 \times 10^{18}$	81 379 271	$1{,}24 \times 10^{18}$	+0,34 %
30	317 826 394	$2{,}26 \times 10^{18}$	81 411 698	$1{,}24 \times 10^{18}$	+0,42 %
60	324 041 496	$2{,}34 \times 10^{18}$	81 418 192	$1{,}24 \times 10^{18}$	+0,44 %
61	309 891 857	$2{,}14 \times 10^{18}$	81 427 075	$1{,}24 \times 10^{18}$	+0,46 %
50	316 478 268	$2{,}24 \times 10^{18}$	81 480 389	$1{,}24 \times 10^{18}$	+0,59 %
12	308 837 595	$2{,}13 \times 10^{18}$	81 584 492	$1{,}24 \times 10^{18}$	+0,85 %
31	312 574 204	$2{,}18 \times 10^{18}$	81 589 561	$1{,}24 \times 10^{18}$	+0,86 %
51	311 175 033	$2{,}16 \times 10^{18}$	81 724 892	$1{,}25 \times 10^{18}$	+1,20 %
22	307 395 363	$2{,}11 \times 10^{18}$	81 774 400	$1{,}25 \times 10^{18}$	+1,32 %
32	309 900 307	$2{,}14 \times 10^{18}$	81 854 176	$1{,}25 \times 10^{18}$	+1,52 %
40	315 549 828	$2{,}22 \times 10^{18}$	81 971 998	$1{,}26 \times 10^{18}$	+1,81 %
62	306 584 663	$2{,}10 \times 10^{18}$	81 998 217	$1{,}26 \times 10^{18}$	+1,88 %
52	308 695 518	$2{,}13 \times 10^{18}$	82 055 985	$1{,}26 \times 10^{18}$	+2,02 %
41	311 251 786	$2{,}16 \times 10^{18}$	82 176 315	$1{,}26 \times 10^{18}$	+2,32 %
13	306 186 056	$2{,}09 \times 10^{18}$	82 336 697	$1{,}27 \times 10^{18}$	+2,72 %
33	307 411 911	$2{,}11 \times 10^{18}$	82 425 057	$1{,}27 \times 10^{18}$	+2,94 %
42	309 026 229	$2{,}13 \times 10^{18}$	82 465 478	$1{,}27 \times 10^{18}$	+3,04 %
20	335 993 975	$2{,}52 \times 10^{18}$	82 558 900	$1{,}27 \times 10^{18}$	+3,27 %
23	305 215 508	$2{,}08 \times 10^{18}$	82 669 415	$1{,}28 \times 10^{18}$	+3,55 %
53	306 704 487	$2{,}10 \times 10^{18}$	82 735 374	$1{,}28 \times 10^{18}$	+3,72 %
63	305 410 102	$2{,}08 \times 10^{18}$	82 943 841	$1{,}29 \times 10^{18}$	+4,24 %
43	307 120 972	$2{,}10 \times 10^{18}$	83 091 811	$1{,}29 \times 10^{18}$	+4,61 %
01	315 512 310	$2{,}22 \times 10^{18}$	83 775 945	$1{,}31 \times 10^{18}$	+6,34 %
02	314 374 563	$2{,}21 \times 10^{18}$	83 783 791	$1{,}31 \times 10^{18}$	+6,36 %
03	312 257 402	$2{,}18 \times 10^{18}$	83 835 119	$1{,}31 \times 10^{18}$	+6,49 %
s0	318 482 486	$2{,}27 \times 10^{18}$	83 863 260	$1{,}32 \times 10^{18}$	+6,56 %
00	318 482 490	$2{,}27 \times 10^{18}$	83 863 378	$1{,}32 \times 10^{18}$	+6,56 %
s2	307 155 720	$2{,}10 \times 10^{18}$	85 114 020	$1{,}35 \times 10^{18}$	+9,77 %
s1	307 415 189	$2{,}11 \times 10^{18}$	85 540 465	$1{,}37 \times 10^{18}$	+10,87 %

TABLE 5.14 – Données complètes pour RSA-704.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
30	6 714 244	$1{,}24 \times 10^{15}$	2 536 409	$6{,}43 \times 10^{14}$	
31	6 463 195	$1{,}15 \times 10^{15}$	2 537 567	$6{,}44 \times 10^{14}$	+0,09 %
50	6 614 987	$1{,}20 \times 10^{15}$	2 549 072	$6{,}50 \times 10^{14}$	+1,00 %
12	6 253 286	$1{,}07 \times 10^{15}$	2 554 065	$6{,}52 \times 10^{14}$	+1,40 %
11	6 475 031	$1{,}15 \times 10^{15}$	2 554 168	$6{,}52 \times 10^{14}$	+1,41 %
32	6 322 765	$1{,}10 \times 10^{15}$	2 554 989	$6{,}53 \times 10^{14}$	+1,47 %
51	6 379 357	$1{,}12 \times 10^{15}$	2 559 608	$6{,}55 \times 10^{14}$	+1,84 %
61	6 300 121	$1{,}09 \times 10^{15}$	2 565 604	$6{,}58 \times 10^{14}$	+2,32 %
52	6 257 702	$1{,}07 \times 10^{15}$	2 586 312	$6{,}69 \times 10^{14}$	+3,97 %
62	6 127 174	$1{,}03 \times 10^{15}$	2 592 776	$6{,}72 \times 10^{14}$	+4,49 %
60	6 842 617	$1{,}28 \times 10^{15}$	2 601 691	$6{,}77 \times 10^{14}$	+5,21 %
40	6 638 218	$1{,}21 \times 10^{15}$	2 602 679	$6{,}77 \times 10^{14}$	+5,29 %
22	6 207 899	$1{,}05 \times 10^{15}$	2 603 690	$6{,}78 \times 10^{14}$	+5,38 %
10	6 940 275	$1{,}32 \times 10^{15}$	2 603 719	$6{,}78 \times 10^{14}$	+5,38 %
33	6 189 728	$1{,}05 \times 10^{15}$	2 610 061	$6{,}81 \times 10^{14}$	+5,89 %
41	6 443 679	$1{,}14 \times 10^{15}$	2 613 977	$6{,}83 \times 10^{14}$	+6,21 %
21	6 441 819	$1{,}13 \times 10^{15}$	2 618 263	$6{,}86 \times 10^{14}$	+6,56 %
13	6 099 883	$1{,}02 \times 10^{15}$	2 620 156	$6{,}87 \times 10^{14}$	+6,71 %
42	6 335 640	$1{,}10 \times 10^{15}$	2 637 605	$6{,}96 \times 10^{14}$	+8,14 %
53	6 155 631	$1{,}04 \times 10^{15}$	2 657 331	$7{,}06 \times 10^{14}$	+9,76 %
43	6 240 426	$1{,}06 \times 10^{15}$	2 692 926	$7{,}25 \times 10^{14}$	+12,72 %
20	7 035 463	$1{,}35 \times 10^{15}$	2 694 055	$7{,}26 \times 10^{14}$	+12,82 %
23	6 066 632	$1{,}00 \times 10^{15}$	2 711 534	$7{,}35 \times 10^{14}$	+14,29 %
63	6 061 559	$1{,}00 \times 10^{15}$	2 723 351	$7{,}42 \times 10^{14}$	+15,28 %
03	6 885 122	$1{,}30 \times 10^{15}$	2 841 000	$8{,}07 \times 10^{14}$	+25,46 %
02	6 904 290	$1{,}31 \times 10^{15}$	2 841 499	$8{,}07 \times 10^{14}$	+25,50 %
01	6 906 046	$1{,}31 \times 10^{15}$	2 841 956	$8{,}08 \times 10^{14}$	+25,54 %
s2	6 811 474	$1{,}27 \times 10^{15}$	2 846 664	$8{,}10 \times 10^{14}$	+25,96 %
00	7 258 620	$1{,}45 \times 10^{15}$	2 880 372	$8{,}30 \times 10^{14}$	+28,96 %
s0	7 259 181	$1{,}45 \times 10^{15}$	2 880 491	$8{,}30 \times 10^{14}$	+28,97 %
s1	7 659 426	$1{,}59 \times 10^{15}$	3 396 493	$1{,}15 \times 10^{15}$	+79,32 %

TABLE 5.15 – Données complètes pour p155.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
31	21 468 306	$1{,}33 \times 10^{16}$	7 288 100	$7{,}97 \times 10^{15}$	
30	22 280 496	$1{,}43 \times 10^{16}$	7 301 643	$8{,}00 \times 10^{15}$	+0,37 %
32	21 049 740	$1{,}28 \times 10^{16}$	7 340 742	$8{,}08 \times 10^{15}$	+1,45 %
50	21 513 630	$1{,}34 \times 10^{16}$	7 341 660	$8{,}08 \times 10^{15}$	+1,48 %
12	20 844 152	$1{,}25 \times 10^{16}$	7 384 306	$8{,}18 \times 10^{15}$	+2,66 %
11	21 546 475	$1{,}33 \times 10^{16}$	7 400 557	$8{,}22 \times 10^{15}$	+3,11 %
51	21 029 612	$1{,}28 \times 10^{16}$	7 417 761	$8{,}25 \times 10^{15}$	+3,59 %
61	20 910 310	$1{,}26 \times 10^{16}$	7 417 887	$8{,}25 \times 10^{15}$	+3,59 %
62	20 433 422	$1{,}20 \times 10^{16}$	7 494 805	$8{,}43 \times 10^{15}$	+5,75 %
60	22 511 205	$1{,}46 \times 10^{16}$	7 508 127	$8{,}46 \times 10^{15}$	+6,13 %
52	20 816 381	$1{,}25 \times 10^{16}$	7 528 444	$8{,}50 \times 10^{15}$	+6,70 %
33	20 732 666	$1{,}24 \times 10^{16}$	7 530 027	$8{,}51 \times 10^{15}$	+6,75 %
22	20 701 498	$1{,}23 \times 10^{16}$	7 544 135	$8{,}54 \times 10^{15}$	+7,15 %
13	20 419 942	$1{,}20 \times 10^{16}$	7 571 375	$8{,}60 \times 10^{15}$	+7,92 %
10	23 265 489	$1{,}55 \times 10^{16}$	7 589 181	$8{,}64 \times 10^{15}$	+8,43 %
21	21 362 438	$1{,}31 \times 10^{16}$	7 589 706	$8{,}64 \times 10^{15}$	+8,45 %
40	21 700 996	$1{,}36 \times 10^{16}$	7 656 009	$8{,}79 \times 10^{15}$	+10,35 %
41	21 431 724	$1{,}33 \times 10^{16}$	7 726 688	$8{,}96 \times 10^{15}$	+12,40 %
53	20 703 467	$1{,}23 \times 10^{16}$	7 737 456	$8{,}98 \times 10^{15}$	+12,71 %
42	21 297 239	$1{,}31 \times 10^{16}$	7 797 696	$9{,}12 \times 10^{15}$	+14,47 %
20	23 138 229	$1{,}53 \times 10^{16}$	7 849 460	$9{,}24 \times 10^{15}$	+16,00 %
63	20 394 199	$1{,}19 \times 10^{16}$	7 861 739	$9{,}27 \times 10^{15}$	+16,36 %
23	20 395 070	$1{,}19 \times 10^{16}$	7 866 604	$9{,}28 \times 10^{15}$	+16,51 %
43	21 211 343	$1{,}29 \times 10^{16}$	7 884 619	$9{,}33 \times 10^{15}$	+17,04 %
03	25 676 095	$1{,}91 \times 10^{16}$	9 163 369	$1{,}26 \times 10^{16}$	+58,08 %
s2	25 808 563	$1{,}93 \times 10^{16}$	9 170 864	$1{,}26 \times 10^{16}$	+58,34 %
02	25 809 523	$1{,}93 \times 10^{16}$	9 170 929	$1{,}26 \times 10^{16}$	+58,34 %
01	25 811 279	$1{,}93 \times 10^{16}$	9 171 292	$1{,}26 \times 10^{16}$	+58,36 %
s0	26 188 557	$1{,}98 \times 10^{16}$	9 215 630	$1{,}27 \times 10^{16}$	+59,89 %
00	26 190 035	$1{,}98 \times 10^{16}$	9 215 701	$1{,}27 \times 10^{16}$	+59,89 %
s1	28 940 807	$2{,}37 \times 10^{16}$	10 769 526	$1{,}74 \times 10^{16}$	+118,36 %

TABLE 5.16 – Données complètes pour p180.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
31	1 978 752	$8{,}51 \times 10^{13}$	648 476	$4{,}21 \times 10^{13}$	
30	2 083 125	$9{,}43 \times 10^{13}$	650 626	$4{,}23 \times 10^{13}$	+0,66 %
50	1 982 615	$8{,}55 \times 10^{13}$	653 358	$4{,}27 \times 10^{13}$	+1,51 %
32	1 938 808	$8{,}16 \times 10^{13}$	655 589	$4{,}30 \times 10^{13}$	+2,21 %
12	1 916 454	$7{,}96 \times 10^{13}$	657 308	$4{,}32 \times 10^{13}$	+2,74 %
11	1 984 857	$8{,}54 \times 10^{13}$	657 967	$4{,}33 \times 10^{13}$	+2,95 %
61	1 926 763	$8{,}05 \times 10^{13}$	661 554	$4{,}38 \times 10^{13}$	+4,07 %
51	1 936 270	$8{,}15 \times 10^{13}$	663 028	$4{,}40 \times 10^{13}$	+4,54 %
62	1 893 132	$7{,}77 \times 10^{13}$	671 975	$4{,}52 \times 10^{13}$	+7,38 %
22	1 906 373	$7{,}87 \times 10^{13}$	675 500	$4{,}56 \times 10^{13}$	+8,51 %
52	1 926 009	$8{,}05 \times 10^{13}$	677 077	$4{,}58 \times 10^{13}$	+9,02 %
21	1 971 971	$8{,}42 \times 10^{13}$	678 108	$4{,}60 \times 10^{13}$	+9,35 %
60	2 125 923	$9{,}80 \times 10^{13}$	678 718	$4{,}61 \times 10^{13}$	+9,54 %
33	1 925 359	$8{,}04 \times 10^{13}$	678 894	$4{,}61 \times 10^{13}$	+9,60 %
13	1 901 372	$7{,}83 \times 10^{13}$	679 181	$4{,}61 \times 10^{13}$	+9,69 %
10	2 182 072	$1{,}03 \times 10^{14}$	683 442	$4{,}67 \times 10^{13}$	+11,08 %
40	1 980 510	$8{,}53 \times 10^{13}$	692 468	$4{,}80 \times 10^{13}$	+14,03 %
41	1 973 283	$8{,}46 \times 10^{13}$	707 211	$5{,}00 \times 10^{13}$	+18,94 %
20	2 227 058	$1{,}07 \times 10^{14}$	713 659	$5{,}09 \times 10^{13}$	+21,11 %
53	1 948 261	$8{,}22 \times 10^{13}$	714 369	$5{,}10 \times 10^{13}$	+21,36 %
23	1 934 876	$8{,}09 \times 10^{13}$	718 629	$5{,}16 \times 10^{13}$	+22,81 %
42	1 975 296	$8{,}46 \times 10^{13}$	723 109	$5{,}23 \times 10^{13}$	+24,34 %
63	1 957 756	$8{,}28 \times 10^{13}$	737 947	$5{,}45 \times 10^{13}$	+29,50 %
43	1 984 188	$8{,}52 \times 10^{13}$	745 938	$5{,}56 \times 10^{13}$	+32,32 %
03	2 483 373	$1{,}34 \times 10^{14}$	887 541	$7{,}88 \times 10^{13}$	+87,32 %
01	2 490 814	$1{,}35 \times 10^{14}$	887 905	$7{,}88 \times 10^{13}$	+87,48 %
02	2 490 830	$1{,}35 \times 10^{14}$	887 956	$7{,}88 \times 10^{13}$	+87,50 %
s2	2 480 183	$1{,}34 \times 10^{14}$	888 767	$7{,}90 \times 10^{13}$	+87,84 %
s0	2 504 747	$1{,}37 \times 10^{14}$	890 097	$7{,}92 \times 10^{13}$	+88,40 %
00	2 504 738	$1{,}37 \times 10^{14}$	890 107	$7{,}92 \times 10^{13}$	+88,41 %
s1	2 548 952	$1{,}40 \times 10^{14}$	949 242	$9{,}01 \times 10^{13}$	+114,27 %

TABLE 5.17 – Données complètes pour $\mathbb{F}_{2^{619}}$.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
31	9 644 786	$2{,}10 \times 10^{15}$	3 484 569	$1{,}21 \times 10^{15}$	
30	10 126 436	$2{,}32 \times 10^{15}$	3 493 773	$1{,}22 \times 10^{15}$	+0,53 %
50	9 761 906	$2{,}15 \times 10^{15}$	3 515 248	$1{,}24 \times 10^{15}$	+1,77 %
32	9 424 672	$2{,}00 \times 10^{15}$	3 517 593	$1{,}24 \times 10^{15}$	+1,90 %
12	9 313 137	$1{,}95 \times 10^{15}$	3 528 628	$1{,}25 \times 10^{15}$	+2,54 %
11	9 670 420	$2{,}11 \times 10^{15}$	3 530 193	$1{,}25 \times 10^{15}$	+2,64 %
51	9 454 582	$2{,}02 \times 10^{15}$	3 548 367	$1{,}26 \times 10^{15}$	+3,70 %
61	9 360 098	$1{,}97 \times 10^{15}$	3 550 649	$1{,}26 \times 10^{15}$	+3,83 %
62	9 153 959	$1{,}89 \times 10^{15}$	3 596 069	$1{,}29 \times 10^{15}$	+6,50 %
52	9 336 976	$1{,}97 \times 10^{15}$	3 607 460	$1{,}30 \times 10^{15}$	+7,18 %
22	9 238 788	$1{,}92 \times 10^{15}$	3 614 219	$1{,}31 \times 10^{15}$	+7,58 %
60	10 265 165	$2{,}37 \times 10^{15}$	3 616 670	$1{,}31 \times 10^{15}$	+7,73 %
33	9 284 572	$1{,}94 \times 10^{15}$	3 623 881	$1{,}31 \times 10^{15}$	+8,16 %
21	9 544 594	$2{,}05 \times 10^{15}$	3 631 009	$1{,}32 \times 10^{15}$	+8,58 %
13	9 159 235	$1{,}89 \times 10^{15}$	3 634 841	$1{,}32 \times 10^{15}$	+8,81 %
10	10 569 620	$2{,}51 \times 10^{15}$	3 644 690	$1{,}33 \times 10^{15}$	+9,40 %
40	9 838 799	$2{,}19 \times 10^{15}$	3 660 460	$1{,}34 \times 10^{15}$	+10,35 %
41	9 634 081	$2{,}10 \times 10^{15}$	3 696 603	$1{,}37 \times 10^{15}$	+12,54 %
42	9 535 985	$2{,}05 \times 10^{15}$	3 747 073	$1{,}40 \times 10^{15}$	+15,63 %
53	9 298 354	$1{,}95 \times 10^{15}$	3 753 044	$1{,}41 \times 10^{15}$	+16,00 %
20	10 617 233	$2{,}53 \times 10^{15}$	3 764 939	$1{,}42 \times 10^{15}$	+16,74 %
23	9 226 205	$1{,}91 \times 10^{15}$	3 822 940	$1{,}46 \times 10^{15}$	+20,36 %
43	9 477 803	$2{,}02 \times 10^{15}$	3 846 167	$1{,}48 \times 10^{15}$	+21,83 %
63	9 238 544	$1{,}92 \times 10^{15}$	3 862 733	$1{,}49 \times 10^{15}$	+22,88 %
s2	12 231 926	$3{,}38 \times 10^{15}$	4 649 193	$2{,}16 \times 10^{15}$	+78,02 %
03	12 363 963	$3{,}46 \times 10^{15}$	4 651 334	$2{,}16 \times 10^{15}$	+78,18 %
02	12 442 274	$3{,}50 \times 10^{15}$	4 652 580	$2{,}16 \times 10^{15}$	+78,27 %
01	12 445 627	$3{,}50 \times 10^{15}$	4 652 845	$2{,}16 \times 10^{15}$	+78,29 %
00	12 735 446	$3{,}67 \times 10^{15}$	4 703 312	$2{,}21 \times 10^{15}$	+82,18 %
s0	12 735 441	$3{,}67 \times 10^{15}$	4 703 398	$2{,}21 \times 10^{15}$	+82,19 %
s1	22 190 421	$1{,}07 \times 10^{16}$	9 075 847	$8{,}24 \times 10^{15}$	+578,38 %

TABLE 5.18 – Données complètes pour $\mathbb{F}_{2^{809}}$.

	Après `purge`		Après `merge`		
Ω	N	$N \times W$	N	$N \times W$	
11	188 580 425	$8{,}49 \times 10^{17}$	65 138 845	$4{,}24 \times 10^{17}$	
61	185 399 885	$8{,}20 \times 10^{17}$	65 501 515	$4{,}29 \times 10^{17}$	+1,12 %
30	192 816 556	$8{,}88 \times 10^{17}$	65 513 540	$4{,}29 \times 10^{17}$	+1,15 %
12	185 046 190	$8{,}17 \times 10^{17}$	65 600 990	$4{,}30 \times 10^{17}$	+1,42 %
21	188 480 429	$8{,}47 \times 10^{17}$	65 646 657	$4{,}31 \times 10^{17}$	+1,57 %
31	188 302 437	$8{,}47 \times 10^{17}$	65 800 281	$4{,}33 \times 10^{17}$	+2,04 %
10	200 713 508	$9{,}61 \times 10^{17}$	65 884 992	$4{,}34 \times 10^{17}$	+2,30 %
50	188 136 760	$8{,}46 \times 10^{17}$	65 964 477	$4{,}35 \times 10^{17}$	+2,55 %
22	183 764 221	$8{,}05 \times 10^{17}$	66 030 150	$4{,}36 \times 10^{17}$	+2,76 %
60	198 153 114	$9{,}37 \times 10^{17}$	66 037 579	$4{,}36 \times 10^{17}$	+2,78 %
32	186 410 816	$8{,}29 \times 10^{17}$	66 263 200	$4{,}39 \times 10^{17}$	+3,48 %
62	183 311 722	$8{,}01 \times 10^{17}$	66 413 060	$4{,}41 \times 10^{17}$	+3,95 %
51	185 963 237	$8{,}26 \times 10^{17}$	66 521 544	$4{,}43 \times 10^{17}$	+4,29 %
13	183 552 821	$8{,}03 \times 10^{17}$	66 820 482	$4{,}46 \times 10^{17}$	+5,23 %
52	185 187 044	$8{,}19 \times 10^{17}$	67 039 863	$4{,}49 \times 10^{17}$	+5,92 %
33	185 193 044	$8{,}18 \times 10^{17}$	67 106 628	$4{,}50 \times 10^{17}$	+6,13 %
23	182 939 672	$7{,}98 \times 10^{17}$	67 603 362	$4{,}57 \times 10^{17}$	+7,71 %
53	184 758 976	$8{,}14 \times 10^{17}$	67 766 411	$4{,}59 \times 10^{17}$	+8,23 %
63	183 169 307	$8{,}00 \times 10^{17}$	67 792 554	$4{,}60 \times 10^{17}$	+8,31 %
40	187 520 013	$8{,}41 \times 10^{17}$	68 079 911	$4{,}63 \times 10^{17}$	+9,23 %
41	186 937 457	$8{,}35 \times 10^{17}$	68 363 238	$4{,}67 \times 10^{17}$	+10,15 %
42	186 640 959	$8{,}32 \times 10^{17}$	68 571 814	$4{,}70 \times 10^{17}$	+10,82 %
43	186 442 611	$8{,}30 \times 10^{17}$	68 797 554	$4{,}73 \times 10^{17}$	+11,55 %
20	212 519 429	$1{,}08 \times 10^{18}$	69 376 064	$4{,}81 \times 10^{17}$	+13,43 %
s2	197 233 222	$9{,}30 \times 10^{17}$	74 451 493	$5{,}54 \times 10^{17}$	+30,64 %
03	197 703 703	$9{,}34 \times 10^{17}$	74 570 015	$5{,}56 \times 10^{17}$	+31,05 %
01	197 944 839	$9{,}37 \times 10^{17}$	74 585 164	$5{,}56 \times 10^{17}$	+31,11 %
02	197 940 661	$9{,}37 \times 10^{17}$	74 585 212	$5{,}56 \times 10^{17}$	+31,11 %
00	198 444 888	$9{,}42 \times 10^{17}$	74 694 800	$5{,}58 \times 10^{17}$	+31,49 %
s0	198 444 886	$9{,}42 \times 10^{17}$	74 694 852	$5{,}58 \times 10^{17}$	+31,49 %
s1	203 255 785	$9{,}84 \times 10^{17}$	78 239 129	$6{,}12 \times 10^{17}$	+44,27 %

TABLE 5.19 – Données complètes pour $\mathbb{F}_{2^{1039}}$.

Bibliographie

[1] L. M. Adleman. « Factoring numbers using singular integers ». In : *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. ACM. 1991, p. 64–71 (cité p. 42).

[2] L. M. Adleman. « The function field sieve ». In : *Algorithmic Number Theory*. Sous la dir. de L. M. Adleman et M.-D. Huang. T. 877. Lecture Notes in Computer Science. Springer-Verlag, 1994, p. 108–121 (cité p. 2, 76).

[3] L. M. Adleman et M.-D. A. Huang. « Function Field Sieve Method for Discrete Logarithms over Finite Fields ». In : *Information and Computation* 151.1–2 (1999), p. 5–16. issn : 0890-5401. doi : 10.1006/inco.1998.2761. url : http://www.sciencedirect.com/science/article/pii/S0890540198927614 (cité p. 77).

[4] A. O. L. Atkin et F. Morain. « Finding suitable curves for the elliptic curve method of factorization ». In : *Mathematics of Computation* 60.201 (1993), p. 399–405 (cité p. 17).

[5] S. Bai, C. Bouvier, A. Kruppa et P. Zimmermann. « Better polynomials for GNFS ». In : *Mathematics of Computation* (2015). doi : 10.1090/mcom3048. url : https://hal.inria.fr/hal-01089507 (cité p. 4, 44, 54).

[6] S. Bai, R. P. Brent et E. Thomé. « Root optimization of polynomials in the number field sieve ». In : *Mathematics of Computation* (2015). doi : 10.1090/S0025-5718-2015-02926-3 (cité p. 48).

[7] S. Bai, E. Thomé et P. Zimmermann. *Factorisation of RSA-704 with CADO-NFS*. Cryptology ePrint Archive, Report 2012/369. 2012. url : http://eprint.iacr.org/ (cité p. 95).

[8] R. Barbulescu. « Familles de courbes adaptées à la factorisation des entiers ». Research report version 2. 2009. url : http://hal.inria.fr/inria-00419218/en/ (cité p. 31, 32).

[9] R. Barbulescu. « Selecting polynomials for the Function Field Sieve ». Preprint, 23 pages. 2013. url : http://hal.inria.fr/hal-00798386 (cité p. 78, 79).

[10] R. Barbulescu, J. W. Bos, C. Bouvier, T. Kleinjung et P. L. Montgomery. « Finding ECM-Friendly Curves through a Study of Galois Properties ». In : *ANTS X: Proceedings of the Tenth Algorithmic Number Theory Symposium*. Sous la dir. d'E. W. Howe et K. S. Kedlaya. T. 1. Open Book Series. Berkeley : Mathematical Sciences Publishers, 2013, p. 63–86. doi : 10.2140/obs.2013.1.63. url : https://hal.inria.fr/hal-00671948 (cité p. 3, 5, 22).

[11] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau et P. Zimmermann. « The relationship between some guy and cryptography ». ECC2012 rump session talk (humoristic). 2012. URL : http://ecc.2012.rump.cr.yp.to/ (cité p. 99).

[12] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau et P. Zimmermann. « Discrete logarithm in GF(2^{809}) with FFS ». In : *Public-Key Cryptography – PKC 2014*. Sous la dir. de H. Krawczyk. T. 8383. Lecture Notes in Computer Science. Springer-Verlag, 2014, p. 221–238. ISBN : 978-3-642-54630-3. DOI : 10.1007/978-3-642-54631-0_13. URL : http://hal.inria.fr/hal-00818124 (cité p. 4, 69, 78, 99).

[13] R. Barbulescu, P. Gaudry, A. Guillevic et F. Morain. *Improvements to the Number Field Sieve for non-prime finite fields*. 2014. URL : http://hal.inria.fr/hal-01052449 (cité p. 72).

[14] R. Barbulescu, P. Gaudry, A. Guillevic et F. Morain. « Improving NFS for the discrete logarithm problem in non-prime finite fields ». In : *Eurocrypt 2015*. Sous la dir. de M. Fischlin et E. Oswald. Eurocrypt 2015, 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Sofia, Bulgaria, 2015, p. 27. URL : https://hal.inria.fr/hal-01112879 (cité p. 69).

[15] R. Barbulescu, P. Gaudry, A. Joux et E. Thomé. « A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic ». In : *Advances in Cryptology – EUROCRYPT 2014*. Sous la dir. de P. Q. Nguyen et E. Oswald. T. 8441. Lecture Notes in Computer Science. Springer-Verlag, 2014, p. 1–16. ISBN : 978-3-642-55219-9. DOI : 10.1007/978-3-642-55220-5_1 (cité p. 69, 78).

[16] E. A. Bender et E. R. Canfield. « An approximate probabilistic model for Structured Gaussian Elimination ». In : *Journal of Algorithms* 31.2 (1999), p. 271–290. ISSN : 0196-6774. DOI : 10.1006/jagm.1999.1008. URL : http://www.sciencedirect.com/science/article/pii/S0196677499910088 (cité p. 103).

[17] D. J. Bernstein, P. Birkner, M. Joye, T. Lange et C. Peters. « Twisted Edwards Curves ». In : *Africacrypt*. Sous la dir. de S. Vaudenay. T. 5023. Lecture Notes in Computer Science. Springer-Verlag, 2008, p. 389–405 (cité p. 11).

[18] D. J. Bernstein, P. Birkner et T. Lange. « Starfish on Strike ». In : *Progress in Cryptology – LATINCRYPT 2010*. Sous la dir. de M. Abdalla et P. S. L. M. Barreto. T. 6212. Lecture Notes in Computer Science. Springer-Verlag, 2010, p. 61–80. ISBN : 978-3-642-14711-1. DOI : 10.1007/978-3-642-14712-8_4 (cité p. 17, 28, 30–32).

[19] D. J. Bernstein, P. Birkner, T. Lange et C. Peters. « ECM using Edwards curves ». In : *Mathematics of Computation* 82 (2013), p. 1139–1179 (cité p. 11, 17, 29, 31).

[20] D. J. Bernstein et T. Lange. « Analysis and optimization of elliptic-curve single-scalar multiplication ». In : *Contemporary Mathematics* 461 (2008), p. 1–20 (cité p. 14).

[21] D. J. Bernstein et A. K. Lenstra. « A general number field sieve implementation ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 103–126 (cité p. 38).

[22] D. J. Bernstein, T. Lange et al. *Explicit-Formulas Database*. URL : http://www.hyperelliptic.org/EFD/ (cité p. 8, 10, 11).

[23] BlueKrypt. *Cryptographic Key Length Recommendation*. URL : http://www.keylength.com (cité p. 84).

[24] H. Boender. « Factoring large integers with the quadratic sieve ». Thèse de doct. Leiden University, 1997 (cité p. 46).

[25] C. Bouvier. *The filtering step of discrete logarithm and integer factorization algorithms*. Preprint, 22 pages. 2013. URL : http://hal.inria.fr/hal-00734654 (cité p. 3, 94).

[26] C. Bouvier, P. Gaudry, L. Imbert, H. Jeljeli et E. Thomé. *Discrete logarithms in* GF(p) — *180 digits*. Archives of the Number Theory Mailing List (NMBRTHRY). 2014. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1406&L=NMBRTHRY#4 (cité p. 69, 74, 97).

[27] C. Bouvier et P. Zimmermann. « Division-Free Binary-to-Decimal Conversion ». In : *IEEE Transactions on Computers* 63.8 (2014), p. 1895–1901. ISSN : 0018-9340. DOI : 10.1109/TC.2014.2315621. URL : https://hal.inria.fr/hal-00864293 (cité p. 4).

[28] S. P. Bradley, A. C. Hax et T. L. Magnanti. *Applied mathematical programming*. Addison-Wesley Reading, MA, 1977 (cité p. 65).

[29] A. Brauer. « On addition chains ». In : *Bulletin of the American Mathematical Society* 45 (1939), p. 736–739. ISSN : 0273–0979 (cité p. 14).

[30] E. Brier et C. Clavier. « New Families of ECM Curves for Cunningham Numbers ». In : *Algorithmic Number Theory – ANTS-IX*. Sous la dir. de G. Hanrot, F. Morain et E. Thomé. T. 6197. Lecture Notes in Computer Science. Springer-Verlag, 2010, p. 96–109 (cité p. 19, 30).

[31] J. P. Buhler, éd. T. 1423. Lecture Notes in Computer Science. Springer-Verlag, 1998. ISBN : 978-3-540-64657-0. DOI : 10.1007/BFb0054849.

[32] J. P. Buhler, H. W. Lenstra Jr et C. Pomerance. « Factoring integers with the number field sieve ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 50–94 (cité p. 35–38, 42, 43).

[33] The CADO-NFS Development Team. *CADO-NFS, An Implementation of the Number Field Sieve Algorithm*. URL : http://cado-nfs.gforge.inria.fr/ (cité p. 4, 38, 58, 74, 85).

[34] E. Canfield, P. Erdős et C. Pomerance. « On a problem of Oppenheim concerning "factorisatio numerorum" ». In : *Journal of Number Theory* 17.1 (1983), p. 1–28. ISSN : 0022-314X. DOI : 10.1016/0022-314X(83)90002-1". URL : http://www.sciencedirect.com/science/article/pii/0022314X83900021 (cité p. 15).

[35] S. Cavallar. « On the Number Field Sieve Integer Factorization Algorithm ». Thèse de doct. Universiteit Leiden, 2002 (cité p. 85, 88, 94).

[36] S. Cavallar et al. « Factorization of a 512-bit RSA modulus ». In : *Advances in Cryptology – EUROCRYPT 2000*. Springer-Verlag. 2000, p. 1–18 (cité p. 95).

[37] S. D. Chowla et T. Vijayaraghavan. « On the largest prime divisors of numbers ». In : *J. Indian Math. Soc* 11 (1947), p. 31–37 (cité p. 45).

[38] H. Cohen. *A course in computational algebraic number theory*. T. 138. Springer-Verlag, 1993 (cité p. 34, 70).

[39] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen et F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Sous la dir. de H. Cohen et G. Frey. CRC Press, 2005. ISBN : 1–58488–518–1 (cité p. 14).

[40] D. Coppersmith. « Solving linear equations over GF(2): block Lanczos algorithm ». In : *Linear algebra and its applications* 192 (1993), p. 33–60. DOI : 10.1016/0024-3795(93)90235-G (cité p. 41).

[41] D. Coppersmith. « Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm ». In : *Mathematics of Computation* 62.205 (1994), p. 333–350 (cité p. 41).

[42] J.-M. Couveignes. « Computing a Square Root for the Number Field Sieve ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 95–102 (cité p. 43).

[43] N. Coxon. *On nonlinear polynomial selection for the number field sieve*. ArXiv e-prints. 2013. URL : http://arxiv.org/abs/1109.6398 (cité p. 49, 62, 63).

[44] N. Coxon. *Montgomery's method of polynomial selection for the number field sieve*. ArXiv e-prints. 2014. URL : http://arxiv.org/abs/1412.6011 (cité p. 62).

[45] R. Crandall et C. Pomerance. *Prime numbers: a computational perspective*. T. 182. Springer-Verlag, 2006 (cité p. 2).

[46] J. E. Cremona. *Elliptic Curve Data*. URL : http://homepages.warwick.ac.uk/~masgaj/ftp/data/ (cité p. 23).

[47] H. te Riele et al. *CWI NFS*. URL : https://gforge.inria.fr/frs/?group_id=4334 (cité p. 38, 94).

[48] J. Detrey. *FFS Factory: Adapting Coppersmith's "Factorization Factory" to the Function Field Sieve*. Cryptology ePrint Archive, Report 2014/419. 2014. URL : http://eprint.iacr.org/ (cité p. 79).

[49] J. Detrey, P. Gaudry et M. Videau. « Relation collection for the Function Field Sieve ». In : *Computer Arithmetic (ARITH), 2013 21st IEEE Symposium on*. Sous la dir. de P.-M. S. Alberto Nannarelli et P. T. P. Tang. IEEE, avr. 2013, p. 201–210. DOI : 10.1109/ARITH.2013.28. URL : http://hal.inria.fr/hal-00736123 (cité p. 78, 79, 99).

[50] K. Dickman. « On the frequency of numbers containing prime factors of a certain relative magnitude ». In : *Ark. Mat. Astr. Fys.* 22 (1930), p. 1–14 (cité p. 45).

[51] W. Diffie et M. E. Hellman. « New directions in cryptography ». In : *IEEE Transactions on Information Theory* 22.6 (1976), p. 644–654 (cité p. 2).

[52] J. D. Dixon. « Asymptotically Fast Factorization of Integers ». In : *Mathematics of Computation* 36.153 (1981), p. 255–260. ISSN : 00255718. URL : http://www.jstor.org/stable/2007743 (cité p. 1).

[53] H. M. Edwards. « A normal form for elliptic curves ». In : *Bulletin of the American Mathematical Society* 44 (juil. 2007), p. 393–422 (cité p. 11).

[54] D. J. Bernstein, P. Birkner, T. Lange, C. Peters et al. *EECM-MPFQ: ECM using Edwards curves*. URL : http://eecm.cr.yp.to/index.html (cité p. 16).

[55] T. ElGamal. « A public key cryptosystem and a signature scheme based on discrete logarithms ». In : *IEEE Transactions on Information Theory* 31.4 (1985), p. 469–472 (cité p. 2).

[56] M. Elkenbracht-Huizing. « An Implementation of the Number Field Sieve ». In : *Experimental Mathematics* 5.3 (1996), p. 231–253. DOI : 10.1080/10586458.1996.10504590 (cité p. 39).

[57] A. Enge, P. Gaudry et E. Thomé. « An L(1/3) Discrete Logarithm Algorithm for Low Degree Curves ». In : *Journal of Cryptology* 24.1 (2011), p. 24–41. ISSN : 0933-2790. DOI : 10.1007/s00145-010-9057-y (cité p. 2).

[58] J. Franke et T. Kleinjung. « Continued fractions and lattice sieving ». In : *Special-Purpose Hardware for Attacking Cryptographic Systems–SHARCS* (2005) (cité p. 40).

[59] C. Monico. *GGNFS*. URL : http://www.math.ttu.edu/~cmonico/software/ggnfs (cité p. 38, 85).

[60] T. Granlund et the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*. URL : http://gmplib.org/ (cité p. 16).

[61] P. Zimmermann et al. *GMP-ECM (Elliptic Curve Method for Integer Factorization)*. URL : https://gforge.inria.fr/projects/ecm/ (cité p. 4, 16).

[62] F. Göloğlu, R. Granger, G. McGuire et J. Zumbrägel. « On the Function Field Sieve and the Impact of Higher Splitting Probabilities ». In : *Advances in Cryptology – CRYPTO 2013*. Sous la dir. de R. Canetti et J. A. Garay. T. 8043. Lecture Notes in Computer Science. Springer-Verlag, 2013, p. 109–128. ISBN : 978-3-642-40083-4. DOI : 10.1007/978-3-642-40084-1_7 (cité p. 69).

[63] F. Göloğlu, R. Granger, G. McGuire et J. Zumbrägel. « Solving a 6120-bit DLP on a Desktop Computer ». In : *Selected Areas in Cryptography – SAC 2013*. Sous la dir. de T. Lange, K. Lauter et P. Lisoněk. Lecture Notes in Computer Science. Springer-Verlag, 2014, p. 136–152. ISBN : 978-3-662-43413-0. DOI : 10.1007/978-3-662-43414-7_7 (cité p. 69).

[64] D. M. Gordon. « Discrete Logarithms in GF(p) Using the Number Field Sieve ». In : *SIAM Journal on Discrete Mathematics* 6.1 (1993), p. 124–138 (cité p. 2, 70).

[65] R. Granger, T. Kleinjung et J. Zumbrägel. *Discrete Logarithms in* $\mathrm{GF}(2^{9234})$. Archives of the Number Theory Mailing List (NMBRTHRY). 2014. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1401&L=NMBRTHRY#4 (cité p. 69).

[66] R. Granger, T. Kleinjung et J. Zumbrägel. *On the Powers of 2*. Cryptology ePrint Archive, Report 2014/300. 2014. URL : http://eprint.iacr.org/ (cité p. 69).

[67] A. Granville. « Smooth numbers: computational number theory and beyond ». In : *Algorithmic number theory: lattices, number fields, curves and cryptography* 44 (2008), p. 267–323 (cité p. 45).

[68] D. Hankerson, S. Vanstone et A. Menezes. *Guide to elliptic curve cryptography*. Springer-Verlag, 2004 (cité p. 14).

[69] W. Hart. *Factorisation of B200*. Archives of the Number Theory Mailing List (NMBRTHRY). 2012. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1208&L=NMBRTHRY#12 (cité p. 95).

[70] H. Jeljeli. « Accelerating Iterative SpMV for Discrete Logarithm Problem Using GPUs ». In : *International Workshop on the Arithmetic of Finite Fields WAIFI 2014*. Gebze, Turkey, 2014. URL : https://hal.inria.fr/hal-00734975 (cité p. 81).

[71] H. Jeljeli. « Resolution of Linear Algebra for the Discrete Logarithm Problem Using GPU and Multi-core Architectures ». In : *Euro-Par 2014 Parallel Processing*. T. 8632. Lecture Notes in Computer Science. Porto, Portugal : Springer-Verlag, 2014, p. 764–775. URL : https://hal.inria.fr/hal-00946895 (cité p. 73, 75).

[72] A. Joux. *A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic*. Cryptology ePrint Archive, Report 2013/095. 2013. URL : http://eprint.iacr.org/ (cité p. 69).

[73] A. Joux. *Discrete logarithms in a 1425-bit finite field*. Archives of the Number Theory Mailing List (NMBRTHRY). 2013. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1301&L=NMBRTHRY#2 (cité p. 69).

[74] A. Joux. « Faster Index Calculus for the Medium Prime Case Application to 1175-bit and 1425-bit Finite Fields ». In : *Advances in Cryptology – EUROCRYPT 2013*. Sous la dir. de T. Johansson et P. Q. Nguyen. T. 7881. Lecture Notes in Computer Science. Springer-Verlag, 2013, p. 177–193. ISBN : 978-3-642-38347-2. DOI : 10.1007/978-3-642-38348-9_11 (cité p. 69).

[75] A. Joux et R. Lercier. « Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the gaussian integer method ». In : *Mathematics of Computation* 72 (2003), p. 953–967. DOI : 10.1090/S0025-5718-02-01482-5 (cité p. 73).

[76] A. Joux, R. Lercier, N. Smart et F. Vercauteren. « The Number Field Sieve in the Medium Prime Case ». In : *Advances in Cryptology – CRYPTO 2006*. Sous la dir. de C. Dwork. T. 4117. Lecture Notes in Computer Science. Springer-Verlag, 2006, p. 326–344. ISBN : 978-3-540-37432-9. DOI : 10.1007/11818175_19 (cité p. 69).

[77] A. Joux et C. Pierrot. *Discrete logarithm record in characteristic 3, $\mathrm{GF}(3^{(5\times 479)})$ a 3796-bit field*. Archives of the Number Theory Mailing List (NMBRTHRY). 2014. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1409&L=NMBRTHRY#4 (cité p. 69).

[78] T. Kleinjung. « Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024 bit integers ». In : *Proceedings of SHARCS* (2006) (cité p. 61).

[79] T. Kleinjung. « On polynomial selection for the general number field sieve ». In : *Mathematics of Computation* 75.256 (2006), p. 2037–2047. DOI : 10.1090/S0025-5718-06-01870-9 (cité p. 39, 47, 49).

[80] T. Kleinjung. *Discrete logarithms in $\mathrm{GF}(p)$ — 160 digits*. Archives of the Number Theory Mailing List (NMBRTHRY). 2007. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind0702&L=NMBRTHRY#5 (cité p. 74).

[81] T. Kleinjung. *Polynomial Selection*. Slides presented at the CADO workshop on integer factorization, Nancy, France. 2008. URL : http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf (cité p. 39, 47, 50, 74).

[82] T. Kleinjung. *Discrete Logarithms in* GF(2^{1279}). Archives of the Number Theory Mailing List (NMBRTHRY). 2014. URL : https://listserv.nodak.edu/cgi-bin/wa.exe?A1=ind1410&L=NMBRTHRY#1 (cité p. 69, 78, 82, 99).

[83] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev et P. Zimmermann. « Factorization of a 768-bit RSA modulus ». In : *Advances in Cryptology – CRYPTO 2010*. Sous la dir. de T. Rabin. T. 6223. Lecture Notes in Computer Science. Springer-Verlag, 2010, p. 333–350 (cité p. 38, 61, 83, 88).

[84] N. Koo, G. H. Jo et S. Kwon. *On Nonlinear Polynomial Selection and Geometric Progression (mod N) for Number Field Sieve*. Cryptology ePrint Archive, Report 2011/292. 2011. URL : http://eprint.iacr.org/ (cité p. 63).

[85] M. Kraitchik. *Théorie des nombres*. Gauthier–Villards, 1922 (cité p. 2).

[86] A. Kruppa. « Speeding up Integer Multiplication and Factorization ». Thèse de doct. Université Henri Poincaré - Nancy I, jan. 2010. URL : http://tel.archives-ouvertes.fr/tel-00477005/en/ (cité p. 31).

[87] R. Laboratories. *The RSA factoring challenge*. 1991 – 2007. URL : http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-factoring-challenge.htm (cité p. 38, 58, 95).

[88] B. A. LaMacchia et A. M. Odlyzko. « Solving Large Sparse Linear Systems Over Finite Fields ». In : *Advances in Cryptology - CRYPTO' 90*. Sous la dir. d'A. Menezes et S. Vanstone. T. 537. Lecture Notes in Computer Science. Springer-Verlag, 1991, p. 109–133 (cité p. 86).

[89] C. Lanczos. « An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators ». In : *Journal of Research of the National Bureau of Standards* 45.4 (1950) (cité p. 41).

[90] J. B. Lasserre. *Moments, positive polynomials and their applications*. Imperial College Press, 2009. ISBN : 978-1-84816-445-1 (cité p. 65).

[91] D. H. Lehmer et R. E. Powers. « On factoring large numbers ». In : *Bulletin of the American Mathematical Society* 37.10 (1931), p. 770–776. ISSN : 02730979. URL : http://www.ams.org/journals/bull/1931-37-10/S0002-9904-1931-05271-X/S0002-9904-1931-05271-X.pdf (cité p. 1).

[92] A. K. Lenstra et H. W. Lenstra Jr, éds. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993 (cité p. 35).

[93] A. K. Lenstra, H. W. Lenstra Jr et L. Lovász. « Factoring polynomials with rational coefficients ». In : *Mathematische Annalen* 261.4 (1982), p. 515–534. ISSN : 0025-5831. DOI : 10.1007/BF01457454 (cité p. 56).

[94] A. K. Lenstra, H. W. Lenstra Jr, M. S. Manasse et J. M. Pollard. « The factorization of the ninth Fermat number ». In : *Mathematics of Computation* 61.203 (1993), p. 319–349 (cité p. 39).

[95] A. K. Lenstra, H. W. Lenstra Jr, M. S. Manasse et J. M. Pollard. « The number field sieve ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 11–42 (cité p. 35).

[96] A. K. Lenstra, E. Tromer, A. Shamir, W. Kortsmit, B. Dodson, J. Hughes et P. Leyland. « Factoring Estimates for a 1024-Bit RSA Modulus ». In : *Advances in Cryptology - ASIACRYPT 2003*. Sous la dir. de C.-S. Laih. T. 2894. Lecture Notes in Computer Science. Springer-Verlag, 2003, p. 55–74. isbn : 978-3-540-20592-0. doi : 10.1007/978-3-540-40061-5_4 (cité p. 61).

[97] H. W. Lenstra Jr. « Factoring integers with elliptic curves ». In : *Annals of Mathematics*. Second Series 126.3 (1987), p. 649–673. url : http://www.jstor.org/stable/1971363 (cité p. 1, 12, 15, 23).

[98] H. W. Lenstra Jr. « The number field sieve: an annotated bibliography ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 1–3 (cité p. 35).

[99] W. Bosma, J. Cannon et C. Playoust. « The Magma algebra system. I. The user language ». In : *Journal of Symbolic Computation* 24.3-4 (1997). Computational algebra and number theory (London, 1993), p. 235–265. issn : 0747-7171. doi : 10.1006/jsco.1996.0125 (cité p. 23).

[100] H. M. Markowitz. « The elimination form of the inverse and its application to linear programming ». In : *Management Science* 3.3 (1957), p. 255–269. doi : 10.1287/mnsc.3.3.255 (cité p. 91).

[101] R. Matsumoto. « Using C_{ab} curves in the function field sieve ». In : *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 82.3 (1999), p. 551–552 (cité p. 77).

[102] P. L. Montgomery. « Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains. » Unpublished. Déc. 1983 (cité p. 14).

[103] P. L. Montgomery. « Speeding the Pollard and Elliptic Curve Methods of Factorization ». In : *Mathematics of Computation* 48.177 (1987), p. 243–264. url : http://www.jstor.org/stable/2007888 (cité p. 10, 14–16, 54).

[104] P. L. Montgomery. « An FFT extension of the elliptic curve method of factorization ». Thèse de doct. University of California, 1992 (cité p. 15, 16).

[105] P. L. Montgomery. « Small geometric progressions modulo n ». Unpublished note of 2 pages. 1993 (cité p. 62).

[106] P. L. Montgomery. « Square roots of products of algebraic numbers ». In : *Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematic*. Sous la dir. de W. Gautschi. T. 48. Proceedings of Symposia in Applied Mathematics. American Mathematical Society, 1994, p. 567–571 (cité p. 43, 119).

[107] P. L. Montgomery. « A Block Lanczos Algorithm for Finding Dependencies over GF(2) ». In : *Advances in Cryptology — EUROCRYPT '95*. Sous la dir. de L. C. Guillou et J.-J. Quisquater. T. 921. Lecture Notes in Computer Science. Springer-Verlag, 1995, p. 106–120. isbn : 978-3-540-59409-3. doi : 10.1007/3-540-49264-X_9 (cité p. 41).

[108] P. L. Montgomery. « Square roots of products of algebraic numbers ». Article non publié, significativement différent de la version publiée [106]. 16 mai 1997 (cité p. 43).

[109] M. A. Morrison et J. Brillhart. « A Method of Factoring and the Factorization of F_7 ». In : *Mathematics of Computation* 29.129 (1975), p. 183–205. issn : 00255718. url : http://www.jstor.org/stable/2005475 (cité p. 1).

[110] P. Gaudry et E. Thomé. mp$\mathbb{F}_q$ *: a finite field library*. URL : http://mpfq.org (cité p. 16).

[111] J. Papadopoulos. *Msieve*. URL : http://sourceforge.net/projects/msieve/ (cité p. 38, 58, 85).

[112] B. A. Murphy. « Modelling the yield of number field sieve polynomials ». In : *Algorithmic Number Theory*. Sous la dir. de J. P. Buhler. T. 1423. Lecture Notes in Computer Science. Springer-Verlag, 1998, p. 137–150. ISBN : 978-3-540-64657-0. DOI : 10.1007/BFb0054858 (cité p. 46).

[113] B. A. Murphy. « Polynomial selection for the number field sieve integer factorisation algorithm ». Thèse de doct. Australian National University, 1999, p. 144 (cité p. 39, 46).

[114] B. A. Murphy et R. P. Brent. « On Quadratic Polynomials for the Number Field Sieve ». In : *Proceedings of the CATS '98*. T. 20. Australian Computer Science Communications. Springer-Verlag, 1998, p. 199–214 (cité p. 46).

[115] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. 2013. URL : http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (cité p. 79).

[116] J. Neukirch. *Class field theory*. T. 280. Springer-Verlag, 1986 (cité p. 17).

[117] P. Q. Nguyen, B. Vallée et al. *The LLL Algorithm. Survey and Applications*. Sous la dir. de P. Q. Nguyen et B. Vallée. Information Security and Cryptography 1. Springer-Verlag, 2010. ISBN : 978-3-642-02295-1. DOI : 10.1007/978-3-642-02295-1 (cité p. 56).

[118] P. Nguyen. « A Montgomery-like square root for the number field sieve ». In : *Algorithmic Number Theory*. Sous la dir. de J. P. Buhler. T. 1423. Lecture Notes in Computer Science. Springer-Verlag, 1998, p. 151–168. ISBN : 978-3-540-64657-0. DOI : 10.1007/BFb0054859 (cité p. 43).

[119] D. Panario, X. Gourdon et P. Flajolet. « An analytic approach to smooth polynomials over finite fields ». In : *Algorithmic Number Theory*. Sous la dir. de J. P. Buhler. T. 1423. Lecture Notes in Computer Science. Springer-Verlag, 1998, p. 226–236. ISBN : 978-3-540-64657-0. DOI : 10.1007/BFb0054865 (cité p. 77).

[120] J. Papadopoulos. *A Self Tuning Filtering Implementation for the Number Field Sieve*. Slides presented at the CADO workshop on integer factorization, Nancy, France. 2008. URL : http://cado.gforge.inria.fr/workshop/slides/papadopoulos.pdf (cité p. 85).

[121] S. C. Pohlig et M. E. Hellman. « An improved algorithm for computing logarithms over GF(p) and its cryptographic significance ». In : *Information Theory, IEEE Transactions on* 24.1 (1978), p. 106–110. ISSN : 0018-9448. DOI : 10.1109/TIT.1978.1055817 (cité p. 2, 68).

[122] J. M. Pollard. « Theorems on factorization and primality testing ». In : *Mathematical Proceedings of the Cambridge Philosophical Society* 76 (nov. 1974), p. 521–528. ISSN : 1469-8064. DOI : 10.1017/S0305004100049252. URL : http://journals.cambridge.org/article_S0305004100049252 (cité p. 1, 13).

[123] J. M. Pollard. « Monte Carlo methods for index computation (mod p) ». In : *Mathematics of computation* 32.143 (1978), p. 918–924 (cité p. 2).

[124] J. M. Pollard. « Factoring with cubic integers ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 4–10 (cité p. 1, 35).

[125] J. M. Pollard. « The lattice sieve ». In : *The development of the number field sieve*. Sous la dir. d'A. K. Lenstra et H. W. Lenstra Jr. T. 1554. Lecture Notes in Computer Science. Springer-Verlag, 1993, p. 43–49 (cité p. 40).

[126] C. Pomerance. « The quadratic sieve factoring algorithm ». In : *Advances in cryptology: EUROCRYPT '84*. Sous la dir. de T. Beth, N. Cot et I. Ingemarsson. T. 209. Lecture Notes in Computer Science. Springer-Verlag, 1985, p. 169–182. isbn : 3–540–16076–0 (cité p. 1).

[127] C. Pomerance et J. W. Smith. « Reduction of huge, sparse matrices over finite fields via created catastrophes ». In : *Experimental Mathematics* 1.2 (1992), p. 89–94. doi : 10.1080/10586458.1992.10504250 (cité p. 86).

[128] T. Prest et P. Zimmermann. « Non-linear polynomial selection for the number field sieve ». In : *Journal of Symbolic Computation* 47.4 (2012). Special Issue for Joachim von zur Gathen at 60, p. 401–409. doi : 10.1016/j.jsc.2011.09.004 (cité p. 63).

[129] R. L. Rivest, A. Shamir et L. Adleman. « A method for obtaining digital signatures and public-key cryptosystems ». In : *Communications of the ACM* 21.2 (1978), p. 120–126 (cité p. 1).

[130] W. A. Stein et al. *Sage Mathematics Software*. The Sage Development Team. url : http://www.sagemath.org (cité p. 20, 30).

[131] O. Schirokauer. « Discrete Logarithms and Local Units ». In : *Philosophical Transactions of the Royal Society of London*. Series A: Physical and Engineering Sciences 345.1676 (1993), p. 409–423. doi : 10.1098/rsta.1993.0139 (cité p. 72).

[132] O. Schirokauer. « Using number fields to compute logarithms in finite fields ». In : *Mathematics of Computation of the American Mathematical Society* 69.231 (2000), p. 1267–1283 (cité p. 70).

[133] O. Schirokauer. « Virtual logarithms ». In : *Journal of Algorithms* 57.2 (2005), p. 140–147. issn : 0196-6774. doi : 10.1016/j.jalgor.2004.11.004. url : http://www.sciencedirect.com/science/article/pii/S0196677404001579 (cité p. 71, 72).

[134] K. Fujisawa, M. Fukuda, K. Kobayashi, M. Kojima, K. Nakata, M. Nakata et M. Yamashita. *SDPA (SemiDefinite Programming Algorithm)*. url : http://sourceforge.net/projects/sdpa/ (cité p. 65).

[135] J.-P. Serre. « Propriétés galoisiennes des points d'ordre fini des courbes elliptiques ». In : *Inventiones mathematicae* 15.4 (1971), p. 259–331 (cité p. 22).

[136] J.-P. Serre. « Quelques applications du théeorèeme de densité de Chebotarev ». In : *Publications Mathématiques de l'Institut des Hautes Études Scientifiques* 54.1 (1981), p. 123–201. doi : 10.1007/BF02698692 (cité p. 23).

[137] D. Shanks. « Analysis and improvement of the continued fraction method of factorization ». In : *American Mathematical Society Notices* 22 (1975). Typed by Stephen McMath (cité p. 1).

[138] J. H. Silverman. *The arithmetic of elliptic curves*. T. 106. Springer-Verlag, 2009 (cité p. 5–8, 10, 18).

[139] R. D. Silverman. « The multiple polynomial quadratic sieve ». In : *Mathematics of Computation* 48.177 (1987), p. 329–339 (cité p. 1, 57).

[140] A. Sutherland. *Computing the image of Galois*. CNTA XII. 2012. URL : http://math.mit.edu/~drew/CNTA12.pdf (cité p. 22).

[141] H. Suyama. *Informal preliminary report (8)*. Oct. 1985 (cité p. 11, 16).

[142] E. Thomé. « Square root algorithms for the number field sieve ». In : *Arithmetic of Finite Fields 4th International Workshop, WAIFI 2012*. Sous la dir. de F. Özbudak et F. Rodríguez-Henríquez. T. 7369. Springer-Verlag, 2012, p. 208–224. DOI : 10.1007/978-3-642-31662-3_15 (cité p. 43).

[143] E. G. Thurber. « On addition chains $1(mn) \leqslant 1(n) - b$ and lower bounds for $c(r)$ ». In : *Duke Mathematical Journal* 40.4 (déc. 1973), p. 907–913. DOI : 10.1215/S0012-7094-73-04085-4 (cité p. 14).

[144] A. Weil. *Basic Number Theory*. 3e éd. T. 144. Classics in Mathematics. Springer-Verlag, 1974 (cité p. 77).

[145] D. H. Wiedemann. « Solving sparse linear equations over finite fields ». In : *IEEE Transactions on Information Theory* 32.1 (1986), p. 54–62 (cité p. 41).

[146] H. C. Williams. « A $p+1$ Method of Factoring ». In : *Mathematics of Computation* 39.159 (1982), p. 225–234. ISSN : 00255718. URL : http://www.jstor.org/stable/2007633 (cité p. 1, 13).

[147] R. S. Williams Jr. « Cubics Polynomials in the Number Field Sieve ». Mém.de mast. Texas Tech University, 2010. URL : http://www.math.ttu.edu/~cmonico/research/Williams_Ronnie_Thesis.pdf (cité p. 63).

[148] P. Zimmermann et B. Dodson. « 20 Years of ECM ». In : *Algorithmic Number Theory – ANTS-VII*. Sous la dir. de F. Hess, S. Pauli et M. E. Pohst. T. 4076. Lecture Notes in Computer Science. Springer-Verlag, 2006, p. 525–542 (cité p. 15, 16).

[149] D. Zywina. *On the surjectivity of mod ℓ representations associated to elliptic curves*. 2011. URL : http://www.math.cornell.edu/~zywina/papers/EffectiveModl.pdf (cité p. 23).

Résumé

Dans cette thèse, nous étudions les problèmes de la factorisation d'entier et de calcul de logarithme discret dans les corps finis. Dans un premier temps, nous nous intéressons à l'algorithme de factorisation d'entier ECM et présentons une méthode pour analyser les courbes elliptiques utilisées dans cet algorithme en étudiant les propriétés galoisiennes des polynômes de division.

Ensuite, nous présentons en détail l'algorithme de factorisation d'entier NFS, et nous nous intéressons en particulier à l'étape de sélection polynomiale pour laquelle des améliorations d'algorithmes existants sont proposées.

Puis, nous présentons les algorithmes NFS-DL et FFS pour le calcul de logarithme discret dans les corps finis. Nous donnons aussi des détails sur deux calculs de logarithme discret effectués durant cette thèse, l'un avec NFS-DL et l'autre avec FFS.

Enfin, nous étudions une étape commune à l'algorithme NFS pour la factorisation et aux algorithmes NFS-DL et FFS pour le calcul de logarithme discret : l'étape de filtrage. Nous l'étudions en détail et nous présentons une amélioration dont nous validons l'impact en utilisant des données provenant de plusieurs calculs de factorisation et de logarithme discret.

Abstract

In this thesis, we study the problems of integer factorization and discrete logarithm computation in finite fields. First, we study the ECM algorithm for integer factorization and present a method to analyze the elliptic curves used in this algorithm by studying the Galois properties of division polynomials.

Then, we present in detail the NFS algorithm for integer factorization and we study in particular the polynomial selection step for which we propose improvements of existing algorithms.

Next, we present two algorithms for computing discrete logarithms in finite fields: NFS-DL and FFS. We also give some details of two computations of discrete logarithms carried out during this thesis, one with NFS-DL and the other with FFS.

Finally, we study a common step of the NFS algorithm for integer factorization and the NFS-DL and FFS algorithms for discrete logarithm computations: the filtering step. We study this step thoroughly and present an improvement for which we study the impact using data from several computations of discrete logarithms and factorizations.