



# Polynomial systems solving and elliptic curve cryptography

Louise Huot

## ► To cite this version:

Louise Huot. Polynomial systems solving and elliptic curve cryptography. Symbolic Computation [cs.SC]. Université Pierre et Marie Curie - Paris VI, 2013. English. NNT : . tel-00925271

**HAL Id: tel-00925271**

**<https://theses.hal.science/tel-00925271>**

Submitted on 7 Jan 2014

**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.



UNIVERSITÉ PIERRE ET MARIE CURIE



École doctorale Informatique,  
Télécommunications et Électronique (Paris)  
ED130

## THÈSE DE DOCTORAT

Pour obtenir le grade de  
**DOCTEUR EN SCIENCES**  
de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité **Informatique**

# Résolution de systèmes polynomiaux et cryptologie sur les courbes elliptiques

Thèse dirigée par Jean-Charles Faugère, Pierrick Gaudry et Guénaél Renault  
préparée au Laboratoire d'informatique de Paris 6 (LIP6).

Présentée et soutenue publiquement par

**Louise Huot**

le **vendredi 13 décembre 2013**

après avis des **rapporteurs**

|                    |                                                   |
|--------------------|---------------------------------------------------|
| M. Reynald LERCIER | Chercheur associé IRMAR, Ingénieur DGA MI         |
| M. Éric SCHOST     | Associate Professor University of Western Ontario |

devant le **jury** composé de

|                         |                                                                                 |
|-------------------------|---------------------------------------------------------------------------------|
| M. Jean-Charles FAUGÈRE | Directeur de Recherche INRIA Paris-Rocquencourt                                 |
| M. Pierrick GAUDRY      | Directeur de Recherche CNRS                                                     |
| M. Antoine JOUX         | Titulaire de la Chaire de Cryptologie de la fondation<br>partenariale de l'UPMC |
| M. Reynald LERCIER      | Chercheur associé IRMAR, Ingénieur DGA MI                                       |
| M. Guénaél RENAULT      | Maitre de Conférences UPMC                                                      |
| M. Mohab SAFEY EL DIN   | Professeur UPMC                                                                 |
| M. Éric SCHOST          | Associate Professor University of Western Ontario                               |
| M. Benjamin SMITH       | Chargé de Recherche INRIA Saclay-Île-de-France                                  |



# Contents

|          |                                                                                                  |           |
|----------|--------------------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                                              | <b>1</b>  |
| <b>I</b> | <b>Gröbner Bases and Polynomial Systems Solving</b>                                              | <b>21</b> |
| <b>2</b> | <b>Gröbner bases</b>                                                                             | <b>23</b> |
| 2.1      | Preliminaries . . . . .                                                                          | 24        |
| 2.1.1    | Ideals and varieties . . . . .                                                                   | 24        |
| 2.1.2    | Gröbner bases: definition and general properties . . . . .                                       | 26        |
| 2.1.3    | Properties of degree reverse lexicographical Gröbner bases . . . . .                             | 31        |
| 2.1.4    | Properties of lexicographical Gröbner bases . . . . .                                            | 37        |
| 2.1.5    | What means solving? . . . . .                                                                    | 41        |
| 2.2      | Gröbner bases algorithms . . . . .                                                               | 41        |
| 2.2.1    | Lazard's algorithm . . . . .                                                                     | 41        |
| 2.2.2    | Efficient algorithms for Gröbner bases: $F_4$ and $F_5$ . . . . .                                | 43        |
| 2.3      | Change of ordering algorithms . . . . .                                                          | 45        |
| 2.3.1    | The FGLM algorithm . . . . .                                                                     | 45        |
| 2.3.2    | Sparse change of ordering for Shape Position ideals: the probabilistic algorithm . . . . .       | 48        |
| 2.3.3    | Sparse change of ordering for Shape Position ideals: the deterministic algorithm . . . . .       | 50        |
| 2.3.4    | Computation of $T_n$ . . . . .                                                                   | 53        |
| 2.4      | Complexity . . . . .                                                                             | 53        |
| 2.4.1    | Gröbner bases algorithms . . . . .                                                               | 53        |
| 2.4.2    | Change of ordering . . . . .                                                                     | 58        |
| 2.4.3    | Polynomial systems solving . . . . .                                                             | 61        |
| <b>3</b> | <b>Solving structured polynomial systems</b>                                                     | <b>65</b> |
| 3.1      | Systems admitting a polynomial change of variables . . . . .                                     | 67        |
| 3.1.1    | An algorithm for solving polynomial systems admitting a polynomial change of variables . . . . . | 67        |
| 3.1.2    | Complexity of $F_5$ steps . . . . .                                                              | 69        |
| 3.1.3    | Complexity of change of ordering steps . . . . .                                                 | 71        |
| 3.1.4    | Comparison with the usual algorithm . . . . .                                                    | 72        |
| 3.2      | Application to polynomial systems invariant under a linear group . . . . .                       | 72        |
| 3.2.1    | Preliminaries on invariant theory . . . . .                                                      | 73        |
| 3.2.2    | Solving systems pointwise invariant under a pseudo-reflection group $\mathbb{G}$ . . . . .       | 74        |

|           |                                                                                                      |            |
|-----------|------------------------------------------------------------------------------------------------------|------------|
| 3.2.3     | Particular case: some examples of groups in semi-direct product with $\mathfrak{S}_n$                | 77         |
| <b>4</b>  | <b>Change of ordering</b>                                                                            | <b>79</b>  |
| 4.1       | Computing the LEX Gröbner basis given the multiplication matrices . . . . .                          | 83         |
| 4.1.1     | Triangular set . . . . .                                                                             | 83         |
| 4.1.2     | Shape Position case . . . . .                                                                        | 85         |
| 4.2       | Computing the multiplication matrices using fast linear algebra . . . . .                            | 86         |
| 4.3       | Polynomial equations with fixed degree: the tame case . . . . .                                      | 88         |
| 4.3.1     | General Complexity analysis . . . . .                                                                | 89         |
| 4.3.2     | Complexity for regular systems . . . . .                                                             | 91         |
| 4.4       | A worst case ultimately not so bad . . . . .                                                         | 93         |
| 4.5       | Polynomial equations with non-fixed degree: the wild case . . . . .                                  | 94         |
| 4.5.1     | Reading directly $T_n$ from the Gröbner basis . . . . .                                              | 94         |
| 4.5.2     | Another algorithm for polynomial systems solving . . . . .                                           | 96         |
| 4.6       | Impact of Algorithm 16 on the practical solving of PoSSo in the worst case . . .                     | 99         |
| <b>II</b> | <b>Algebraic Cryptanalysis of the Elliptic Curves Discrete Logarithm</b>                             | <b>101</b> |
| <b>5</b>  | <b>Elliptic curves</b>                                                                               | <b>103</b> |
| 5.1       | Definitions . . . . .                                                                                | 104        |
| 5.2       | Elliptic curves representations . . . . .                                                            | 106        |
| 5.2.1     | Short Weierstrass form . . . . .                                                                     | 106        |
| 5.2.2     | Twisted Jacobi intersection curves . . . . .                                                         | 107        |
| 5.2.3     | Twisted Edwards curves . . . . .                                                                     | 108        |
| 5.2.4     | Universal Edwards model of elliptic curves . . . . .                                                 | 109        |
| 5.3       | Discrete logarithm problem and generic algorithms . . . . .                                          | 111        |
| 5.3.1     | Pohlig Hellman reduction . . . . .                                                                   | 111        |
| 5.3.2     | Baby step giant step . . . . .                                                                       | 112        |
| 5.3.3     | Pollard $\rho$ method . . . . .                                                                      | 113        |
| 5.4       | Semaev summation polynomials . . . . .                                                               | 114        |
| 5.4.1     | Computing summation polynomials . . . . .                                                            | 115        |
| 5.4.2     | Twisted Jacobi intersection curves . . . . .                                                         | 116        |
| 5.4.3     | Twisted Edwards curves . . . . .                                                                     | 117        |
| 5.4.4     | Universal Edwards model of elliptic curves . . . . .                                                 | 117        |
| 5.5       | Gaudry's index calculus attack for ECDLP solving . . . . .                                           | 117        |
| 5.5.1     | Presentation of the algorithm . . . . .                                                              | 117        |
| 5.5.2     | Complexity analysis . . . . .                                                                        | 121        |
| 5.5.3     | Balancing relation search and linear algebra using the <i>double large prime variation</i> . . . . . | 122        |
| 5.5.4     | Variant " $n - 1$ " . . . . .                                                                        | 123        |
| 5.5.5     | Diem's variant of the index calculus attack . . . . .                                                | 124        |
| 5.6       | Using symmetries to improve the ECDLP solving . . . . .                                              | 124        |
| 5.6.1     | Solving the point decomposition problem . . . . .                                                    | 125        |
| 5.6.2     | Computation of summation polynomials . . . . .                                                       | 127        |

|          |                                                                                                           |            |
|----------|-----------------------------------------------------------------------------------------------------------|------------|
| <b>6</b> | <b>Point decomposition problem in high characteristic</b>                                                 | <b>129</b> |
| 6.1      | Impact of the elliptic curve representation on the PDP solving . . . . .                                  | 132        |
| 6.2      | Impact of a 2-torsion subgroup on the PDP solving . . . . .                                               | 133        |
| 6.2.1    | Action of the 2-torsion on the solutions of the PDP . . . . .                                             | 133        |
| 6.2.2    | Action of the 2-torsion on the polynomial systems modelling the PDP . . . . .                             | 136        |
| 6.3      | Action of the 4-torsion on the PDP . . . . .                                                              | 139        |
| 6.3.1    | Twisted Edwards curve . . . . .                                                                           | 139        |
| 6.3.2    | Universal Edwards model of elliptic curves . . . . .                                                      | 139        |
| 6.3.3    | Twisted Jacobi intersection curve . . . . .                                                               | 139        |
| 6.4      | Experimental results and security estimates . . . . .                                                     | 141        |
| 6.4.1    | Experiments with $n = 4$ . . . . .                                                                        | 141        |
| 6.4.2    | Experiments for $n = 5$ and $n = 6$ . . . . .                                                             | 143        |
| 6.4.3    | Security level estimates . . . . .                                                                        | 145        |
| <b>7</b> | <b>Summation polynomials in characteristic 2</b>                                                          | <b>149</b> |
| 7.1      | Compact representation of summation polynomials in characteristic two . . . . .                           | 153        |
| 7.1.1    | Symmetries . . . . .                                                                                      | 153        |
| 7.1.2    | Density . . . . .                                                                                         | 158        |
| 7.2      | Compact summation polynomials by resultant and Gröbner bases . . . . .                                    | 159        |
| 7.3      | Outline of sparse multivariate polynomial interpolation algorithm . . . . .                               | 161        |
| 7.3.1    | Description of Zippel's sparse multivariate polynomial interpolation algorithm . . . . .                  | 161        |
| 7.3.2    | Complexity and probability of success of Zippel's algorithm . . . . .                                     | 164        |
| 7.4      | Summation polynomials by implicit sparse multivariate interpolation . . . . .                             | 165        |
| 7.4.1    | Evaluation of summation polynomials using factorization and resultant of univariate polynomials . . . . . | 166        |
| 7.4.2    | Sparing factorizations . . . . .                                                                          | 169        |
| 7.4.3    | Degree of summation polynomials . . . . .                                                                 | 171        |
| 7.4.4    | Computation of the eighth summation polynomial . . . . .                                                  | 172        |
| 7.4.5    | Discussion about the computation of the ninth summation polynomial . . . . .                              | 173        |
| 7.5      | Application to the Discrete Logarithm Problem . . . . .                                                   | 175        |
| 7.5.1    | Using symmetries to speed up the PDP solving in characteristic two . . . . .                              | 175        |
| 7.5.2    | Benchmarks on the PDP solving . . . . .                                                                   | 176        |
|          | <b>List of Tables</b>                                                                                     | <b>179</b> |
|          | <b>List of Figures</b>                                                                                    | <b>181</b> |
|          | <b>List of Algorithms</b>                                                                                 | <b>183</b> |
|          | <b>Bibliography</b>                                                                                       | <b>185</b> |



## CHAPTER 1

# Introduction

La cryptologie à clé publique (ou asymétrique) repose sur l'existence de fonctions à *sens unique*. Une fonction à sens unique est une fonction facile à évaluer mais dont l'application inverse est difficile à évaluer. La première mise en pratique de la cryptologie asymétrique est proposée par Diffie et Hellman dans [DH76] où ils introduisent un protocole d'échange de clés. La sécurité de leur protocole, repose sur la difficulté de résoudre le problème du logarithme discret dans le groupe  $\mathbb{F}_q^\times$  formé par les éléments inversibles d'un corps fini à  $q$  éléments.

**Problème du logarithme discret (DLP).** *Soit un groupe cyclique  $(\mathbb{G}, \oplus)$  d'ordre  $m$  fini et de générateur  $g$ . Étant donné un élément  $h$  dans  $\mathbb{G}$ , le problème du logarithme discret est de trouver un entier  $x$  dans  $\mathbb{Z}/m\mathbb{Z}$  tel que*

$$h = [x]g = \underbrace{g \oplus \cdots \oplus g}_{x \text{ fois}}.$$

Il existe plusieurs autres problèmes mathématiques permettant de définir des fonctions supposées à sens unique. Nous pouvons mentionner le calcul d'une racine  $n$ -ième modulo  $\mathcal{N}$  sur lequel repose le célèbre cryptosystème RSA [RSA78]. Sachant que ce problème peut être résolu très efficacement lorsque la factorisation de  $\mathcal{N}$  est connue, la sécurité des cryptosystèmes correspondant est étroitement liée à la complexité de factoriser des entiers.

Plus tard, ont également été introduits des cryptosystèmes basés sur la difficulté de résoudre des systèmes polynomiaux (*e.g.* HFE [Pat96]) ou sur la difficulté de résoudre le problème du plus court vecteur (*e.g.* NTRU [HPS98]).

Un groupe dans lequel il est intéressant d'instancier le DLP est le groupe formé par les points rationnels d'une courbe elliptique définie sur un corps fini. Dans ce cas particulier, le DLP est appelé le problème du logarithme discret sur les courbes elliptiques et est noté ECDLP. De nos jours, la cryptologie sur les courbes est devenue une des thématiques principales de la cryptologie à clé publique. En effet, contrairement au DLP dans les corps finis ou à la factorisation il existe des instances du ECDLP pour lesquelles les meilleurs algorithmes connus résolvant ce problème ont une complexité exponentielle en la taille du groupe. Dans certains cas, les attaques algébriques sur le ECDLP sont plus efficaces que les attaques génériques.

En 1995, Patarin [Pat95] initie les attaques par résolution de systèmes polynomiaux en proposant une attaque sur le cryptosystème de Matsumoto et Imai [MI88]. Ce type d'attaques sera par la suite appelé la *cryptanalyse algébrique*. Depuis les années 2000, les attaques basées sur la résolution de système polynomiaux ont connu de nombreux succès. Nous pouvons mentionner par exemple [FJ03, BFP12] qui proposent des attaques contre HFE et certaines de ces variantes.

La cryptanalyse algébrique se déroule en deux étapes. Dans un premier temps, nous devons mettre en place une modélisation sous forme de systèmes polynomiaux du cryptosystème à attaquer. Ensuite, la sécurité du cryptosystème est évaluée par la difficulté de résoudre les systèmes polynomiaux obtenus.

Ce type d'attaques arrive naturellement pour la cryptanalyse des cryptosystèmes sur les courbes. En effet, les courbes (hyper)-elliptiques étant des objets géométriques, elles admettent donc une représentation algébrique. Notons que l'utilisation effective des courbes (hyper)-elliptiques est rendue possible grâce à leur représentation algébrique qui induit une arithmétique efficace dans le groupe correspondant. Ainsi la résolution des problèmes liés aux courbes (hyper)-elliptiques est reliée à la résolution de systèmes polynomiaux. Par exemple, nous pouvons mentionner [GS12, FLR11] pour le comptage de points ou [Gau09, Die11b, Nag10, JV12] pour la résolution du DLP.

Bien que l'existence d'une mise en équations des problèmes sur les courbes est naturelle, trouver une *bonne* modélisation n'est pas toujours évident. L'efficacité de telles attaques repose donc d'une part sur le choix de la modélisation, pour s'assurer que les systèmes peuvent être efficacement construits et résolus. D'autre part sur l'efficacité des outils pour la résolution des systèmes polynomiaux.

Cette thèse se situe à l'intersection de la résolution de systèmes polynomiaux et la cryptologie sur les courbes elliptiques. Les enjeux principaux de cette thèse sont doubles. Dans un premier temps, notre but est de fournir des outils efficaces pour la cryptanalyse algébrique ou pour tout autre application de la résolution de systèmes polynomiaux. Puis, en tirant parti des propriétés intrinsèques des courbes notre second objectif est d'établir des modélisations des cryptosystèmes considérés les plus adaptées possible aux outils dont on dispose.

## Cryptographie sur les courbes elliptiques

La cryptographie sur les courbes elliptiques a été introduite indépendamment par Miller [Mil86] et Koblitz [Kob87]. L'avantage de ces cryptosystèmes comparés à ceux basés sur le DLP dans les corps finis ou sur la factorisation est qu'ils fournissent de meilleurs niveaux de sécurité pour des tailles de clés similaires. En effet, il existe des algorithmes de complexité sous-exponentielle pour la factorisation d'entiers ou la résolution du DLP dans  $\mathbb{F}_q^\times$ . Pour ces deux problèmes les algorithmes permettant d'obtenir une telle complexité sont basés sur les méthodes de calcul d'indice. On peut mentionner [BLP93, Cop93, CP05] pour la factorisation et [AD94, Jou13b] pour le DLP. Dans le cas de la factorisation, il existe également des méthodes utilisant les courbes elliptiques (ECM) [Len87, CP05].

Récemment, Barbulescu *et al* [BGJT13] ont amélioré la complexité du DLP dans les corps finis de petite caractéristique en proposant un algorithme de complexité quasi-polynomiale.

Il existe de nombreuses instances du ECDLP telles que les meilleurs algorithmes de résolution soient les algorithmes génériques. Ces algorithmes ne tirent parti d'aucune structure du groupe dans lequel est instancié le DLP. Leur complexité est exponentielle et un résultat de Shoup [Sho97] montre qu'en général, la meilleure complexité pour ces algorithmes est en  $O(\sqrt{m})$  opérations dans un groupe  $\mathbb{G}$  d'ordre  $m$ . Parmi les algorithmes génériques, la méthode  $\rho$  de Pollard [Pol78] est optimale.

En plus de la sécurité, une seconde problématique de la cryptologie est de fournir des cryptosystèmes les plus efficaces possibles. Dans ce contexte, un des buts des cryptologues est de fournir des représentations de courbes elliptiques procurant une arithmétique perfor-

mante. À titre d'exemple de représentations de courbes, nous pouvons mentionner les courbes d'Edwards tordues [BL07, BBJ<sup>+</sup>08, Edw07] ou les courbes en intersections de Jacobi tordues [CC86, FNW10]. L'impact des symétries particulières de ces courbes sur la résolution du DLP sera étudié plus tard. Une liste détaillée des représentations de courbes elliptiques existantes et de leur arithmétique respective est disponible dans [BL].

Quelques années après l'apparition de la cryptologie sur les courbes elliptiques, Koblitz [Kob89] suggère l'utilisation des courbes hyper-elliptiques. La sécurité des cryptosystèmes correspondants dépend donc de la difficulté de résoudre le problème du logarithme discret dans le groupe des classes de diviseurs d'une courbe hyper-elliptique définie sur un corps fini. Ce cas particulier du DLP est noté HCDLP pour « *hyperelliptic curve discrete logarithm problem* ».

Pour estimer la sécurité des cryptosystèmes basés sur le HCDLP, la résolution de ce problème a été largement étudiée ces dernières années. En particulier, pour différentes familles de courbes de genre grand, des méthodes par calcul d'indice ont été développées [ADH94, Cou01, EG02, EG07, Hes04]. En utilisant la méthode des « *double large prime* » de Gaudry *et al* [GTTD07], si la taille du corps fini est suffisamment grande et pour des courbes de genre fixé supérieur à trois les méthodes par calcul d'indice sont alors plus rapides que la méthode  $\rho$  de Pollard.

Dans le cas particulier de courbes non hyper-elliptiques de genre trois, Diem et Thomé proposent une amélioration des algorithmes par calcul d'indice [Die06, DT08]. Cependant les algorithmes par calcul d'indice pour la résolution du HCDLP ne s'appliquent pas aux genres un et deux.

Depuis les dix dernières années, afin d'obtenir de meilleures complexités pour la résolution du ECDLP, divers algorithmes par calcul d'indice ont été développés. Une des premières tentatives d'algorithme par calcul d'indice pour la résolution du ECDLP a été proposée par Semaev dans [Sem04]. Cependant son attaque ne s'applique réellement ni en pratique ni en théorie.

En 2009, Gaudry [Gau09] introduit une méthode de résolution par calcul d'indice du logarithme discret dans une variété abélienne de dimension  $n$  finie. Soit  $E$  une courbe elliptique définie sur un corps fini  $\mathbb{F}_{q^n}$  avec  $n > 1$ . L'application d'une restriction de Weil permet de transférer le DLP dans  $E(\mathbb{F}_{q^n})$  au DLP dans une variété abélienne de dimension  $n$  sur  $\mathbb{F}_q$ . Ainsi en utilisant les travaux de Semaev et son algorithme de résolution du DLP dans les variétés abéliennes, Gaudry propose [Gau09] un nouvel algorithme de résolution du ECDLP par calcul d'indice. Plus tard, Diem [Die11b, Die11a] obtient des preuves rigoureuses que pour certaines familles de courbes, le ECDLP peut être résolu en temps sous-exponentiel. Cependant son attaque n'a pas d'impact en pratique sur la résolution du ECDLP.

Notons que Nagao [Nag10] a introduit une variante des algorithmes par calcul d'indice adaptée aux courbes hyper-elliptiques. Cependant, dans le cas de courbes elliptiques, son algorithme semble moins efficace que ceux mentionnés précédemment.

Depuis l'introduction des algorithmes par calcul d'indice pour la résolution du ECDLP, la communauté leur porte un intérêt croissant. Par exemple, Joux et Vitse proposent une nouvelle version [JV13] de l'algorithme de Gaudry. Dans le cas où  $q$  est de taille moyenne, leur algorithme permet d'améliorer la complexité de l'attaque de Gaudry. Dans le cas de corps de caractéristique deux, Faugère *et al* [FPPR12] présentent une version améliorée de l'algorithme par calcul d'indice de Diem. Leur algorithme ne donne pas lieu à une attaque en pratique mais diminue la complexité de l'algorithme de Diem pour les courbes binaires. Suite à ces travaux, des hypothèses sur la complexité du ECDLP ont été proposées dans [PQ12].

Ces hypothèses sont à l’heure actuelle difficilement vérifiables en théorie et en pratique et font l’objet d’une étude intensive [YJSPT13, ST13].

Si le degré  $n$  de l’extension du corps est un nombre composé, Joux et Vitse [JV12] introduisent un nouvel algorithme de résolution du ECDLP. Leur algorithme combine l’attaque GHS [GHS02] et une variante de l’attaque par décomposition de Nagao. Ils donnent de plus, des applications pratiques de leur attaque. En particulier, ils résolvent le problème du logarithme discret sur une courbe définie sur  $\mathbb{F}_{p^6}$  avec  $p$  un nombre premier de 26 bits en environ 110 000 heures de calcul sur un coeur CPU.

Le point commun entre tous ces algorithmes de résolution du ECDLP par calcul d’indice est qu’ils requièrent tous la résolution de systèmes polynomiaux.

## Résolution de systèmes d’équations polynomiales

Résoudre des systèmes polynomiaux est un problème central en mathématiques. Ce n’est pas seulement un problème important en lui-même mais il a aussi un large champ d’applications. Ainsi, ce problème apparaît dans de nombreuses disciplines telles que la théorie des codes [LY97, DBP11], la théorie des jeux [Dat03, Stu02], l’optimisation [GS11], *etc* ou évidemment comme mentionné précédemment la cryptologie [BPW06, Jou13b, Nag10, Die11b, Gau09, JV12].

La nature omniprésente de ce problème fait de l’étude de sa complexité un problème central de l’informatique théorique. Par exemple, dans le contexte de la géométrie algébrique, Safey El Din et Schost [SS11, BRSS12] ont proposé le premier algorithme pour résoudre le problème des cartes routières améliorant la complexité de l’algorithme de Canny [Can93]. La complexité de leur algorithme dépend de la complexité de résoudre efficacement des systèmes polynomiaux. En cryptographie, la récente avancée majeure de Joux [Jou13b] pour la résolution du DLP dans les corps finis repose fortement sur la même capacité.

## Représentation des solutions

Selon le contexte, *résoudre un système polynomial* a différents sens. Si l’on considère des systèmes à coefficients dans les corps finis, alors généralement *résoudre* signifie lister toutes les solutions dans ce corps.

Afin, de répondre aux besoins des différentes applications, un algorithme de résolution de systèmes polynomiaux doit fournir une sortie *correcte* ou utilisable dans tous les contextes. Nous avons donc besoin d’une représentation des solutions permettant de retrouver ces dernières très efficacement.

Les bases de Gröbner sont aux systèmes polynomiaux ce que la forme échelonnée en ligne est aux systèmes linéaires. Pour un ordre monomial fixé, étant donné un système d’équations polynomiales, sa base de Gröbner associée par rapport à l’ordre monomial fixé est unique après normalisation. Une bonne représentation, permettant en particulier de lister les solutions dans le cas des corps finis, est donnée par la base de Gröbner pour l’ordre lexicographique (dénoté l’ordre LEX). En effet, sous des hypothèses de généricité, le système à résoudre engendre un idéal dit en *Shape Position*.

**Idéaux en *Shape Position*.** Soit  $\mathcal{I}$  un idéal de  $\mathbb{K}[x_1, \dots, x_n]$  avec un nombre fini de solutions  $D$  dans une clôture algébrique de  $\mathbb{K}$  et comptées avec multiplicité. L’idéal  $\mathcal{I}$  est dit en *Shape*

Position si sa base de Gröbner lexicographique est de la forme

$$\left\{ \begin{array}{c} x_1 - h_1(x_n) \\ \vdots \\ x_{n-1} - h_{n-1}(x_n) \\ h_n(x_n) \end{array} \right\}$$

où  $h_1, \dots, h_n \in \mathbb{K}[x_n]$ ,  $\deg(h_n) = D$  et  $\deg(h_i) < D$  pour  $i = 1, \dots, n-1$ .

À partir de la base de Gröbner LEX d'un idéal en *Shape Position*, résoudre un système polynomial se résume à la résolution du polynôme univarié  $h_n$ . Les algorithmes pour le calcul des racines de polynômes univariés ont leur complexité en fonction de  $D$  (le degré de  $h_n$ ) bien maîtrisée. En général la résolution de  $h_n$  est négligeable en comparaison du calcul de la base LEX.

Par exemple, si  $\mathbb{K} = \mathbb{F}_q$  est un corps fini, lister les solutions de  $h_n$  dans  $\mathbb{F}_q$  peut se faire en  $\tilde{O}(D)$  (voir [VZGG03]) opérations arithmétiques dans  $\mathbb{F}_q$  où la notation  $\tilde{O}$  signifie que l'on omet les facteurs logarithmiques en  $q$  et  $D$ .

Dans tous les cas, même si  $\mathcal{S}$  n'est pas en *Shape Position*, la base de Gröbner lexicographique donne une bonne représentation des solutions. En effet, à partir de cette base de Gröbner, trouver les solutions de  $\mathcal{S}$  est toujours réduit à la résolution d'un ou plusieurs polynômes univariés. Par conséquence, tout au long de cette thèse nous définirons le problème PoSSo comme suit.

**Résolution de systèmes polynomiaux (PoSSo).** *Étant donné un système d'équations polynomiales  $\mathcal{S}$  de  $\mathbb{K}[x_1, \dots, x_n]$ , le problème PoSSo consiste à calculer la base de Gröbner lexicographique de l'idéal engendré par  $\mathcal{S}$ .*

## Complexité du problème PoSSo

Une contribution clé pour la résolution de PoSSo est le résultant multivarié introduit par Macaulay [Mac94] au début de 20ième siècle. L'avancée majeure suivante apparut dans les années 60 lorsque Buchberger introduit, dans sa thèse [Buc06, Buc65], le concept de base de Gröbner et le premier algorithme pour les calculer. Depuis, les bases de Gröbner ont été intensivement étudiées (voir par exemple [BS87a, CLO07, Stu02, LL91, Laz83, Fau02]) et sont devenues un outil puissant pour la résolution de systèmes polynomiaux.

Un résultat de complexité majeur sur la résolution de PoSSo fut montré par Lakshman et Lazard dans [LL91]. Ce résultat établit que le problème PoSSo, pour des systèmes ayant un nombre fini de solutions, peut être résolu en un temps simplement exponentiel en le degré maximum  $d$  des équations du système en entrée. C'est à dire, le problème PoSSo peut être résolu en  $d^{O(n)}$  opérations arithmétiques où  $n$  est le nombre de variables. Grâce à la borne de Bézout, le nombre de solutions peut être borné par une quantité exponentielle en ce degré. Ainsi ce résultat donne une première étape vers une complexité polynomiale en le nombre de solutions pour la résolution de PoSSo. Dans notre contexte la borne de Bézout peut être énoncée de la manière suivante.

**Borne de Bézout.** *Soient  $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$  et  $d_1, \dots, d_n$  leur degré respectif. Si  $\mathcal{S} = \{f_1, \dots, f_n\}$  à un nombre de solutions fini  $D$  (dans la clôture algébrique de  $\mathbb{K}$  et comptées avec multiplicité) alors  $D \leq \prod_{i=1}^n d_i$ .*

Il existe certaines instances particulières du problème PoSSo pour lesquelles il peut être résolu en une complexité sous-cubique en  $D$ . Par exemple, lorsque  $\mathbb{K} = \mathbb{C}$  si les racines réelles sont en nombre  $O(\log_2(D))$  alors on peut approcher toutes ces racines réelles en  $\tilde{O}(12^n D^2)$  opérations arithmétiques dans  $\mathbb{K}$ , voir [MP98]. Toujours pour la caractéristique zéro, si la structure multiplicative de l'algèbre quotient est connue alors Bostan, Salvy et Schost [BSS03] ont montré que l'on pouvait calculer une RUR en  $O\left(n2^n D^{\frac{5}{2}}\right)$  opérations arithmétiques dans  $\mathbb{K}$ .

Tandis que pour ces cas particuliers il existe des algorithmes de complexité sous-cubique en  $D$ , à notre connaissance lorsqu'aucune structure n'est supposée sur le système, la meilleure complexité pour calculer une base de Gröbner lexicographique est en  $O(nD^3)$  opérations arithmétiques dans  $\mathbb{K}$ .

D'un point de vu algorithmique les ordres monomiaux peuvent différer. Certains sont intéressants pour leur efficacité (au sens calcul efficace de la base de Gröbner associée) tandis que d'autres (*e.g.* LEX) permettent d'obtenir une bonne représentation des solutions. Par exemple, les ordres du degré (pondéré) lexicographique inverse (DRL ou WDR dans le cas pondéré) sont usuellement plus efficaces pour le calcul de base de Gröbner. D'après ces observations, la stratégie usuelle de résolution du problème PoSSo par calculs de base de Gröbner est : dans un premier temps calculer une base de Gröbner pour l'ordre DRL ; ensuite de cette base de Gröbner retrouver la base de Gröbner LEX. Cet algorithme est décrit en Figure 1.1.

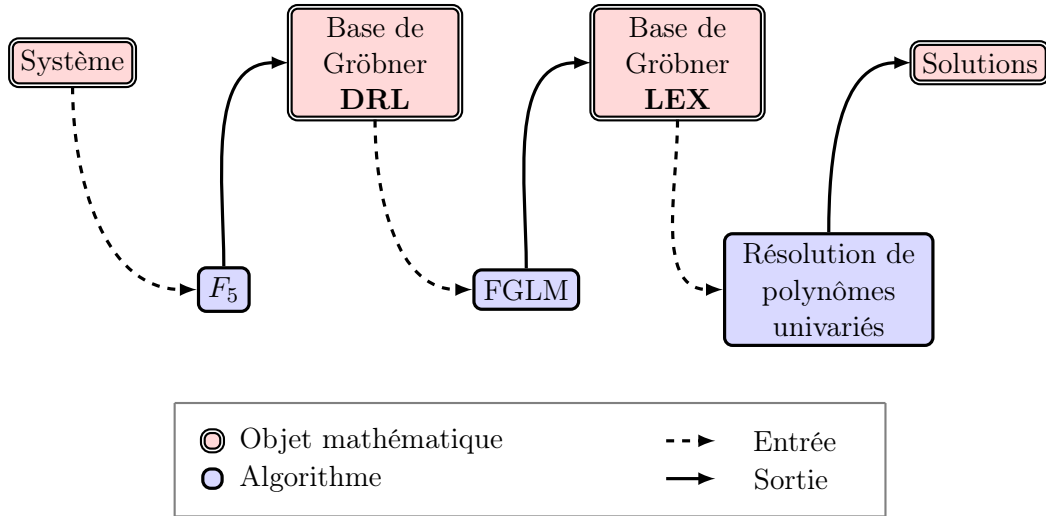


Figure 1.1: Résolution de systèmes polynomiaux par bases de Gröbner.

Pour calculer la base de Gröbner DRL, on peut utiliser les algorithmes efficaces  $F_4$  ou  $F_5$  proposés par Faugère [Fau99, Fau02]. Étant donné cette base de Gröbner, le calcul de la base LEX peut se faire en utilisant un algorithme de changement d'ordre tel que FGLM [FGLM93]. Plus récemment, des nouveaux algorithmes de changement d'ordre tirant parti du caractère creux des matrices impliquées ont été introduits par Mou dans sa thèse [Mou13], voir aussi les articles correspondant [FM11, FM13].

## Résolution de systèmes structurés

Des publications récentes ont montré que la résolution de systèmes structurés pouvait se faire de manière bien plus efficace que celle des systèmes génériques (sans structure particulière). Par exemple Spaenlehauer a étudié dans sa thèse [Spa12] les systèmes bihomogènes [FSS11], déterminantiels [FSS13] ou encore les systèmes booléens [BFSS13]. Il a en particulier montré que des algorithmes dédiés pour de telles structures permettent de diminuer de manière significative la complexité du problème PoSSo. Prenons l'exemple des systèmes bilinéaires dont l'un des deux blocs de variables ne contient que deux variables. Les algorithmes dédiés à la résolution des systèmes bilinéaires ont dans ce cas une complexité polynomiale en le nombre total de variables du système. Cette complexité est à comparer avec celle obtenue en utilisant les algorithmes usuels qui est elle exponentielle en le nombre de variables.

Dans le cas de systèmes admettant des symétries, la théorie des invariants [Kan01, CLO07, Stu08] permet d'accélérer la résolution des systèmes. En effet, admettons que le système soit invariant sous l'action d'un groupe linéaire  $\mathbb{G}$ . Une réécriture de ce système permet alors de diviser le nombre de solutions du système par le cardinal de  $\mathbb{G}$ . Comme mentionné précédemment, la complexité de résoudre un système ayant un lien étroit avec le nombre de solutions, l'utilisation des symétries va donc permettre d'accélérer la résolution. Notons tout de même que dans certains cas (selon le groupe  $\mathbb{G}$ ) les méthodes de la théorie des invariants ne seront pas toujours efficaces. Dans ce cas, une autre solution est l'utilisation des bases SAGBI [FR09] ou encore dans le cas de groupes abéliens l'utilisation d'un algorithme de calcul de base de Gröbner dédié [FS13].

Nous verrons que les systèmes avec symétries sont étroitement liés aux systèmes quasi-homogènes. Les systèmes quasi-homogènes ont été étudiés en particulier dans [FSV13]. Soit  $\mathcal{S}$  un système de  $\mathbb{K}[x_1, \dots, x_n]$  quasi-homogène selon le système de poids  $(w_1, \dots, w_n)$ . En comparaison avec un système homogène dont les équations sont de même degré que les équations de  $\mathcal{S}$ , les auteurs de [FSV13] montrent que tirer parti d'une telle structure permet de gagner un facteur polynomial en  $\prod_{i=1}^n w_i$  sur la complexité totale de la résolution de systèmes polynomiaux par base de Gröbner.

Récemment de telles structures ont été mises en évidence en cryptanalyse algébrique. En particulier pour la résolution du DLP dans les corps finis ou sur les courbes. Dans les récentes avancées algorithmiques sur la résolution du DLP sur  $\mathbb{F}_q^\times$  [BGJT13, Jou13b, GGMZ13] la première amélioration de la complexité fut obtenue grâce à la mise en évidence d'une structure bilinéaire sur les systèmes polynomiaux à résoudre [Jou13b]. Dans le cas de la résolution du DLP sur les courbes, la mise en oeuvre de l'attaque pratique dans [JV12] a été possible encore grâce à l'exploitation d'une telle structure.

## Énoncé des problématiques

Un des objectifs de cette thèse est l'étude des attaques par calcul d'indice pour la résolution du ECDLP. Plus particulièrement, nous nous intéressons à l'attaque de Gaudry [Gau09] que nous rappelons brièvement. Soit  $E$  une courbe elliptique définie sur un corps fini  $\mathbb{F}_{q^n}$  non premier *i.e.*  $n > 1$ . Étant donné  $P$  (d'ordre  $m$ ) et  $Q$  dans  $E(\mathbb{F}_{q^n})$  tel qu'il existe un entier  $x$  vérifiant  $Q = [x]P$ , l'algorithme de Gaudry, pour le calcul de  $x$ , se divise en trois étapes :

1. Calculer la base de facteurs  $\mathcal{F} = \{(x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}$  ;

2. Trouver au moins  $\#\mathcal{F} + 1$  relations de la forme  $[a_i]P \oplus [b_i]Q = P_1 \oplus \cdots \oplus P_n$  avec  $P_1, \dots, P_n \in \mathcal{F}$  et  $a_i, b_i$  sont choisis aléatoirement dans  $\mathbb{Z}/m\mathbb{Z}$  ;
3. Finalement, le calcul de  $x$  se fait par algèbre linéaire.

Si  $n$  est considéré fixé, en utilisant la « double large prime variation » [GTDD07] cet algorithme a une complexité en  $\tilde{O}\left(q^{2-\frac{2}{n}}\right)$  où la notation  $\tilde{O}$  signifie que l'on omet les facteurs logarithmiques en  $q$ . Cependant cette complexité cache un facteur exponentiel en  $n$ . En effet l'étape (2) de l'algorithme de Gaudry nécessite la résolution du problème suivant.

**Problème de décomposition de points (PDP).** Soit  $E$  une courbe elliptique définie sur  $\mathbb{F}_{q^n}$  avec  $n > 1$ . Étant donné  $R$  dans  $E(\mathbb{F}_{q^n})$  et  $\mathcal{F} = \{(x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}$  trouver  $P_1, \dots, P_n \in \mathcal{F}$  tels que  $R = P_1 \oplus \cdots \oplus P_n$ .

Grâce à l'utilisation des polynômes de sommation de Semaev [Sem04] la résolution du PDP se résume à la résolution d'un système d'équations polynomiales. Ces systèmes polynomiaux ayant un nombre de solutions exponentiel en  $n$  plus précisément  $D = 2^{n(n-1)}$ , l'étape de changement d'ordre sera l'étape bloquante, en théorie et également en pratique, de la résolution du PDP. La première problématique que l'on a rencontrée fut donc d'améliorer l'étape de changement d'ordre dans la résolution du PDP. Pour ce faire, deux solutions s'offrent à nous. Nous pouvons soit proposer de nouveaux algorithmes de changement d'ordre pour base de Gröbner ayant une complexité plus faible. Sinon, nous pouvons trouver une nouvelle modélisation du PDP permettant de diminuer le nombre de solutions des systèmes à résoudre.

Les courbes elliptiques sont des objets fortement structurés. En effet, les courbes elliptiques possèdent des symétries particulières. Notons que selon la représentation choisie, les symétries peuvent être plus ou moins présentes. Voir la Figure 1.2 pour différentes représentations graphiques de courbes elliptiques définies sur les réels. Nous pouvons remarquer que les trois représentations de courbes présentées en Figure 1.2 possèdent une symétrie axiale par rapport à l'axe des abscisses. Par contre les courbes d'Edwards ou les intersection de Jacobi possèdent des symétries supplémentaires. Par exemple, on observe une symétrie centrale par rapport à l'origine.

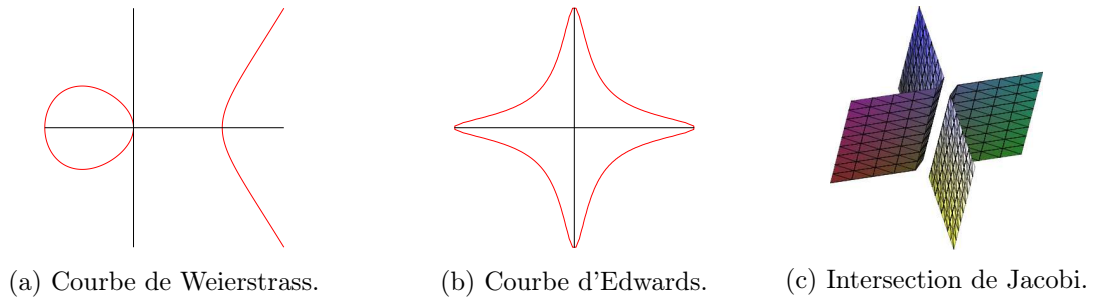


Figure 1.2: Exemples de courbes elliptiques définies sur les réels.

Une problématique naturelle est donc de trouver une modélisation du PDP permettant d'utiliser les symétries des courbes. C'est à dire de mettre en avant des groupes linéaires agissant sur les systèmes modélisant le PDP et ainsi de diminuer leur nombre de solutions.

Cependant, cette stratégie n'est efficace que dans les cas où la théorie des invariants peut être appliquée. C'est à dire lorsque la caractéristique du corps ne divise pas l'ordre du groupe

considéré (cas non modulaire). En particulier lorsque le corps est de caractéristique deux nous ne pourrions pas utiliser la théorie des invariants. De plus, même dans le cas non modulaire, la théorie des invariants nous permet de conclure seulement sur la complexité du changement d'ordre qui dépend directement du nombre de solutions. En particulier, nous n'avons aucune information sur le comportement des algorithmes de calcul de base de Gröbner tel que  $F_5$ .

L'étude de la complexité de l'algorithme  $F_5$  dépend fortement d'une propriété de *régularité*. Afin d'obtenir une analyse complète de l'impact de l'utilisation des symétries sur la résolution du PDP, il est indispensable de disposer d'un algorithme de résolution de systèmes avec symétries permettant de conserver cette propriété de régularité.

Outre la résolution du PDP, une deuxième étape bloquante dans l'algorithme de Gaudry est la construction des systèmes à résoudre. En effet, nous avons vu que ces systèmes sont obtenus à partir des polynômes de sommation. Or ces derniers sont de degré exponentiel en  $n$  et la méthode de Semaev habituellement utilisée pour calculer ces polynômes requiert l'utilisation de résultants multivariés. En pratique, les méthodes actuelles ne permettent de calculer les polynômes de sommation que pour  $n \leq 5$ .

Pour résumer certaines des problématiques liées aux attaques algébriques du DLP sur les courbes elliptiques dans le contexte de l'algorithme de Gaudry sont les suivantes :

- Calculer efficacement une modélisation du PDP sous forme de systèmes polynomiaux. De manière plus restrictive, comment calculer efficacement les polynômes de sommation ?
- Accélérer la phase de changement d'ordre dans la résolution du PDP.
  - Existe-t-il des algorithmes de changement d'ordre avec une complexité sous-cubique en le nombre de solutions ?
  - Mettre en évidence des symétries sur les systèmes à résoudre pour en diminuer le nombre de solutions.
- Étudier l'impact d'éventuelles symétries sur les algorithmes de calcul de base de Gröbner. Existe-t-il un algorithme de résolution de systèmes polynomiaux tirant parti des symétries et dont la complexité totale est maîtrisée ?
- Utilisation, pour la résolution du PDP, des symétries des courbes elliptiques définies sur des corps de caractéristique deux.

Dans la section suivante, nous présentons les contributions apportées dans cette thèse répondant en partie aux problématiques ci-dessus.

## Contributions

Dans un premier temps, nous présenterons de nouveaux résultats de complexité pour la résolution du problème PoSSo. Nous nous intéresserons en particulier à la complexité de l'algorithme présenté en Figure 1.1. Puis nous étudierons le cas des systèmes admettant des symétries.

### Complexité du problème PoSSo

Étant donné un système d'équations polynomiales  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  la complexité du problème PoSSo dépend du coût du calcul de la base de Gröbner pour l'ordre

DRL et de la complexité des algorithmes de changement d'ordre (voir Figure 1.1 pour rappel de l'algorithme de résolution de PoSSo). Les résultats de complexité de l'algorithme  $F_5$  [Bar04, BFS04, BFSY05] impliquent que le calcul de la base de Gröbner DRL peut se faire en  $O\left(n \binom{nd+1}{n}^\omega\right)$  opérations arithmétique dans  $\mathbb{K}$  où  $d$  est le degré maximal des équations en entrée du système  $\mathcal{S}$  et  $\omega$  est l'exposant dans la complexité de la multiplication de deux matrices denses. Ainsi, d'après [VW12] on a  $2 \leq \omega < 2.3727$ . Cette complexité peut se réécrire sous la forme  $O(ne^{\omega n} d^{\omega n})$  lorsque  $n \rightarrow \infty$  (que  $d$  tende vers l'infini ou non) et  $O(d^{\omega n})$  lorsque  $d \rightarrow \infty$  et  $n$  est fixé.

La complexité de l'étape de changement d'ordre pour le calcul de la base de Gröbner LEX peut s'exprimer selon le nombre de solutions du système  $D$  et est donnée par  $O(nD^3)$  opérations arithmétiques dans  $\mathbb{K}$ .

D'après la borne de Bézout on a  $D \leq d^n$ . La complexité de l'étape de changement d'ordre domine donc la complexité du problème PoSSo qui devient  $O(nd^{3n})$  opérations arithmétiques dans  $\mathbb{K}$  quelque soit le paramètre qui tend vers l'infini. Lorsque toutes les équations en entrée ont le même degré *i.e.*  $\deg(f_i) = d$  pour tout  $i \in \{1, \dots, n\}$  alors la borne de Bézout implique que génériquement  $D = d^n$  et la complexité du problème PoSSo peut s'exprimer en fonction de  $D$  par  $O(nD^3)$  opérations arithmétiques dans  $\mathbb{K}$ .

### Complexité sous-cubique pour la résolution de PoSSo.

Pour le cas particulier du changement d'ordre de l'ordre DRL vers l'ordre LEX, nous montrons que la complexité de cette étape peut se ramener à la complexité de la multiplication de matrices. Pour ce faire, nous proposons de nouveaux algorithmes de changement d'ordre pour base de Gröbner.

Étant donné une base de Gröbner pour un ordre monomial  $>_1$  d'un idéal  $\mathcal{I}$  de  $\mathbb{K}[x_1, \dots, x_n]$  et un second ordre monomial  $>_2$ , les algorithmes de changement d'ordre retournent la base de Gröbner de  $\mathcal{I}$  par rapport à l'ordre  $>_2$ .

Ces algorithmes se décomposent généralement en deux étapes. La première est le calcul de la structure multiplicative de l'algèbre quotient  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . Pour tout ordre monomial, l'algèbre quotient a une structure d'espace vectoriel de dimension  $D$  dont la base dépend de l'ordre monomial choisi. Notons  $B_{>_1} = \{\epsilon_D >_1 \dots >_1 \epsilon_1 = 1\}$  la base de l'algèbre quotient vu comme un  $\mathbb{K}$ -espace vectoriel par rapport à l'ordre monomial  $>_1$ . Notons que  $\epsilon_1, \dots, \epsilon_D$  sont des monômes de  $\mathbb{K}[x_1, \dots, x_n]$ . Le calcul de la structure multiplicative de l'algèbre quotient requiert de trouver un représentant de tous les monômes de la forme  $x_i \epsilon_j$  dans  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  par rapport à la base  $B_{>_1}$ . L'algorithme proposé dans [FGLM93] suggère de parcourir les monômes  $x_i \epsilon_j$  en nombre au plus  $nD$  dans l'ordre croissant pour  $>_1$ . Ainsi, chaque représentant peut être calculé grâce à un produit matrice-vecteur de taille  $(D, D) \times (D, 1)$ .

**Calcul de la structure multiplicative de l'algèbre quotient par multiplication de matrices.** Nous proposons un premier algorithme pour le calcul de la structure multiplicative de l'algèbre quotient permettant de calculer les représentants de tous les monômes de même degré simultanément. Plus précisément, nous montrerons que ces représentants peuvent être obtenus par mise sous forme échelon d'une matrice de Macaulay de taille  $(nD, (n+1)D)$ . On itère donc non plus sur les monômes mais sur les degrés en nombre  $nd$  soit  $\log(D)$  lorsque le degré  $d$  des équations en entrée est fixé ( $d$  ne tend pas vers l'infini) et lorsque  $>_1$  est l'ordre DRL (éventuellement pondéré).

Une fois la structure multiplicative de l'algèbre quotient connue, la deuxième étape des algorithmes de changement d'ordre consiste à calculer la base de  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  vu comme un  $\mathbb{K}$ -espace vectoriel mais cette fois-ci par rapport au second ordre monomial  $>_2$ . Simultanément à la construction de cette base, la base de Gröbner par rapport à l'ordre monomial  $>_2$  est reconstruite. Comme démontré dans [FGLM93], la construction de la base de l'algèbre quotient et de la base de Gröbner se ramène à des tests d'indépendance linéaire de vecteurs représentants des monômes dans l'algèbre quotient par rapport à la base  $B_{>_1}$ . Encore une fois l'algorithme usuel construit monôme par monôme en nombre  $D$  la nouvelle base.

Dans [FM11, FM13], les auteurs proposent des algorithmes de changement d'ordre dédiés au cas où  $>_2$  est l'ordre LEX. En supposant la structure de la base LEX connue (l'idéal est en *Shape Position*) Faugère et Mou supposent la connaissance au préalable de la base  $B_{>_{\text{lex}}}$ . Ainsi seule la base de Gröbner LEX contenant  $n$  polynômes est à calculer. En utilisant la structure de cette base de Gröbner, ils montrent que le calcul de chacun des polynômes se réduit à la résolution d'un système linéaire de type Hankel de taille  $D$ . Ils obtiennent ainsi un algorithme de changement d'ordre vers l'ordre LEX très efficace. Cependant la construction de la matrice de Hankel nécessite le calcul des vecteurs  $T^j \mathbf{r}$  pour  $j = 0, \dots, 2D-1$  où  $T$  est une matrice carrée de taille  $D$  et  $\mathbf{r}$  est un vecteur colonne de taille  $D$ . Les auteurs de [FM11, FM13] considérant la matrice  $T$  creuse construisent ces vecteurs de manière itérative. Seulement, dans le cas dense la complexité d'une telle construction est donc en  $O(D^3)$  opérations arithmétiques dans  $\mathbb{K}$ .

**Calcul de la base de Gröbner LEX par multiplication de matrices.** Nous proposons deux nouveaux algorithmes (un déterministe et un probabiliste) de changement d'ordre pour le calcul d'une base de Gröbner LEX en *Shape Position*. Ces algorithmes calculent les  $n$  polynômes de la base LEX de la même façon que dans [FM11, FM13]. C'est-à-dire par la résolution de systèmes de Hankel. La différence principale réside dans la construction des matrices de Hankel. Afin de calculer les vecteurs  $T^j \mathbf{r}$  pour  $j = 0, \dots, 2D-1$  nous utilisons un algorithme de Keller-Gehrig [KG85]. Cet algorithme ramène le calcul de ces vecteurs aux produits de  $O(\log(D))$  matrices de taille au plus  $(D, D)$ .

De plus, nous proposons une généralisation de l'algorithme de Keller-Gehrig. Cet algorithme nous permet de développer un algorithme de changement d'ordre dédié aux idéaux admettant un ensemble triangulaire (forme plus générale que la *Shape Position*) pour base de Gröbner LEX. Pour ces idéaux, nous adaptons directement l'algorithme FGLM. De manière similaire aux idéaux en *Shape Position* nous tirons avantage de la forme connue de la base LEX. En effet, ceci nous permet de prédire à l'avance les tests d'indépendance linéaire requis et de les effectuer simultanément.

Une particularité des algorithmes de changement d'ordre dédiés aux idéaux en *Shape Position* est qu'ils ne nécessitent pas de connaître toute la structure multiplicative de l'algèbre quotient. En effet, pour ces algorithmes seule la représentation matricielle de la multiplication par  $x_n$  dans l'algèbre quotient vue comme un  $\mathbb{K}$ -espace vectoriel est requise. Notons  $T_n$  cette matrice.

**Nouvel algorithme de type Las Vegas pour la résolution de PoSSo.** Nous proposons un nouvel algorithme de résolution de systèmes polynomiaux par base de Gröbner. Cet algorithme de type Las Vegas, permet de s'assurer que les idéaux considérés soient en *Shape*

*Position* dès lors qu'ils n'ont que des racines simples. De plus, le calcul de la matrice  $T_n$  est complètement optimisé dans le cas où  $>_1$  est l'ordre DRL et ne nécessite pas le calcul de la structure multiplicative complète de l'algèbre quotient. Soit  $\mathcal{I}$  un idéal de  $\mathbb{K}[x_1, \dots, x_n]$ . Notons  $g \cdot \mathcal{I}$  l'idéal  $\mathcal{I}$  auquel on applique le changement linéaire de variables  $g \in \mathbf{GL}(\mathbb{K}, n)$ . Nous montrons qu'il existe un ouvert de Zariski  $U \subset \mathbf{GL}(\mathbb{K}, n)$  tel que pour tout  $g \in U$  la matrice  $T_n$  peut être lue (*i.e.* sans opération arithmétique) à partir de la base de Gröbner DRL de l'idéal  $g \cdot \mathcal{I}$ . Ainsi, le calcul de la matrice  $T_n$  peut se faire gratuitement et le degré des équations n'intervient plus dans cette complexité.

L'analyse de complexité de ces différents algorithmes permet de montrer qu'étant donné une base de Gröbner pour l'ordre DRL, calculer la base de Gröbner LEX peut se faire en  $\tilde{O}(D^\omega)$  opérations arithmétiques dans  $\mathbb{K}$ . La notation  $\tilde{O}$  signifie que l'on omet les facteurs logarithmiques en  $D$  et polynomiaux en  $n$ . Nous obtenons donc le résultat suivant sur la complexité de PoSSo.

**Théorème 1.1.** *Soit  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  tel que  $d \geq \deg(f_i)$ . Sous des hypothèses de régularité résoudre le problème PoSSo peut se faire en*

- *temps déterministe si  $d$  est un entier fixé et  $\mathcal{S}$  admet une base LEX en Shape Position ;*
- *temps probabiliste si  $\mathcal{S}$  n'a que des racines simples.*

Dans les deux cas le nombre d'opérations arithmétiques nécessaires à la résolution de PoSSo est en  $\tilde{O}(e^{\omega n} d^{\omega n} + D^\omega)$  si  $n \rightarrow \infty$ .

Lorsque  $n$  est fixé et  $d \rightarrow \infty$  dans le deuxième cas, résoudre le problème PoSSo peut se faire en  $\tilde{O}(d^{\omega n} + D^\omega)$  opérations arithmétiques dans  $\mathbb{K}$ .

Sous des hypothèses de genericité, ces complexités s'expriment respectivement sous la forme  $\tilde{O}(e^{\omega n} D^\omega)$  et  $\tilde{O}(D^\omega)$ .

En Figure 1.3, nous résumons les différentes complexités liées à la résolution du problème PoSSo. « Fast FGLM » désigne les algorithmes rapides de changement d'ordre mentionnés dans le Théorème 1.1.

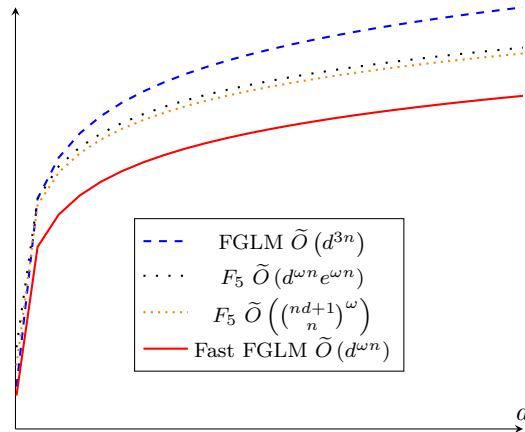


Figure 1.3: Comparaison des complexités des deux étapes de la résolution du problème PoSSo par base de Gröbner. Le nombre de variables est fixé à  $n = 20$  et le degré des équations  $d$  tend vers l'infini.

La complexité du problème PoSSo étant étroitement liée au nombre de solutions des systèmes à résoudre, nous nous sommes naturellement intéressé à la complexité de résoudre des systèmes admettant des symétries.

### Impact des symétries sur la complexité de PoSSo

Supposons que le système  $\mathcal{S} = \{f_1, \dots, f_n\}$  soit invariant sous l'action d'un groupe linéaire  $\mathbb{G}$ . La théorie des invariants nous permet de conclure que l'utilisation de l'action de  $\mathbb{G}$  divise par  $(\#\mathbb{G})^3$  (ou  $(\#\mathbb{G})^\omega$  en utilisant les algorithmes du Théorème 1.1) la complexité de l'étape de changement d'ordre. Par contre elle ne fournit pas d'information sur la complexité du calcul de la base DRL.

Nous nous intéressons au cas plus général où le système admet un changement de variables polynomial. C'est à dire qu'il existe  $\vartheta_1, \dots, \vartheta_n \in \mathbb{K}[x_1, \dots, x_n]$  et  $g_1, \dots, g_n \in \mathbb{K}[x_1, \dots, x_n]$  tels que  $g_i(\vartheta_1, \dots, \vartheta_n) = f_i$  pour tout  $i \in \{1, \dots, n\}$ . Les systèmes invariants sous l'action de groupes linéaires pseudo-réfectifs apparaîtront comme un cas particulier de ces systèmes. Notons  $w_i = \deg(\vartheta_i)$  et  $\vartheta_i^{(h)}$  la partie homogène de plus haut degré de  $\vartheta_i \in \mathbb{K}[x_1, \dots, x_n]$ .

La stratégie habituelle pour résoudre un tel système est dans un premier temps résoudre le système  $\mathcal{S}' = \{g_1, \dots, g_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  à l'aide des outils habituels comme par exemple l'algorithme en Figure 1.1. Puis de résoudre plusieurs systèmes de la forme  $\{\vartheta_1 - \alpha_1, \dots, \vartheta_n - \alpha_n\}$  où  $(\alpha_1, \dots, \alpha_n)$  est une solution de  $\mathcal{S}'$ .

**Utilisation de la structure quasi-homogène pour la résolution de systèmes avec symétries.** L'algorithme que nous proposons pour la résolution de ces systèmes est très proche de celui présenté ci-dessus. La différence principale réside dans la résolution du système  $\mathcal{S}'$ . En effet, pour sa résolution nous proposons d'utiliser l'algorithme en Figure 1.1 à la différence près que nous ne considérons pas l'ordre DRL pour la première base de Gröbner. Nous chercherons donc à calculer en premier lieu une base de Gröbner pour l'ordre du degré pondéré lexicographique inverse défini par le système de poids  $(\deg(\vartheta_1), \dots, \deg(\vartheta_n))$ . En effet, nous montrons que l'existence d'un tel changement de variables entraîne une structure quasi-homogène. Plus précisément, nous obtenons le résultat suivant.

**Proposition 1.1.** *Soit  $\{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  un système régulier. Soit  $\vartheta_1, \dots, \vartheta_n \in \mathbb{K}[x_1, \dots, x_n]$  tels que  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  sont algébriquement indépendants. Si pour tout  $i$  dans  $\{1, \dots, n\}$  il existe  $g_i \in \mathbb{K}[y_1, \dots, y_n]$  tel que  $f_i = g_i(\vartheta_1, \dots, \vartheta_n)$ , alors le système  $\{g_1, \dots, g_n\}$  de  $\mathbb{K}[y_1, \dots, y_n]$  équipé du degré pondéré donné par le système de poids  $(\deg(\vartheta_1), \dots, \deg(\vartheta_n))$  est régulier. De plus si  $\text{wdeg}$  dénote le degré pondéré mentionné précédemment, on a  $\deg(f_i) = \text{wdeg}(g_i)$ .*

Ainsi considérer ce système de poids permet de conserver la propriété de régularité et la complexité totale de l'algorithme pourra être estimée. Plus précisément, nous montrons qu'une telle structure permet de diviser la complexité totale de la résolution de PoSSo par un facteur  $(\prod_{i=1}^n w_i)^\alpha$  où selon les hypothèses  $\alpha = \omega$  ou 3.

**Théorème 1.2.** *Soit un système  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  admettant le changement de variables donné par  $\vartheta_1, \dots, \vartheta_n \in \mathbb{K}[x_1, \dots, x_n]$  avec  $\deg(\vartheta_i) = w_i$ . Supposons que  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  soient algébriquement indépendants et  $\deg(f_i) \leq d$  pour tout  $i \in \{1, \dots, n\}$ . Alors sous des hypothèses de régularité la complexité de résoudre  $\mathcal{S}$  est en*

- $\tilde{O}\left(\frac{e^{\omega n} d^{\omega n}}{(\prod_{i=1}^n w_i)^\omega}\right)$  opérations arithmétiques dans  $\mathbb{K}$  si  $d$  est un entier fixé;
- $\tilde{O}\left(\frac{d^{3n}}{(\prod_{i=1}^n w_i)^3}\right)$  opérations arithmétiques dans  $\mathbb{K}$  sinon.

La notation  $\tilde{O}$  signifie que l'on omet les facteurs polynomiaux en  $n$ .

Dans le cas des systèmes invariants sous l'action d'un groupe linéaire pseudo-réfectif, nous verrons que cette action de groupe permet de mettre en évidence un changement de variables tel que  $\prod_{i=1}^n w_i = \#\mathbb{G}$ .

Muni de ces nouveaux outils, nous nous intéressons ensuite à la résolution du problème de décomposition de points.

## Modélisation et résolution du PDP

Les systèmes polynomiaux modélisant le PDP sont de la forme  $\{f_1, \dots, f_n\} \subset \mathbb{F}_q[x_1, \dots, x_n]$  avec  $\deg(f_1) = \dots = \deg(f_n) = 2^{n-1}$ . La borne de Bézout implique donc que  $D$ , le nombre de solutions de ces systèmes, est borné par  $2^{n(n-1)}$ . En pratique, on observe que cette borne est atteinte. Ainsi dans le contexte où Gaudry a présenté son algorithme de résolution du ECDLP par calcul d'indice, la complexité de la résolution du PDP était en  $O(n2^{3n(n-1)})$  opérations arithmétiques dans  $\mathbb{F}_q$ .

**Utilisation des symétries pour la résolution du PDP.** La mise en évidence de représentations de courbes elliptiques possédant des symétries particulières ainsi que l'utilisation des résultats précédents nous a permis d'établir les résultats suivants.

**Théorème 1.3.** *Soit  $E$  une courbe elliptique définie sur  $\mathbb{F}_{q^n}$  avec  $n > 1$  et  $q = p^k$  tel que  $p > n$ . Si  $E$  peut être mise en représentation d'Edwards tordue ou sous forme d'intersection de Jacobi tordue ou encore respectant le modèle d'Edwards universel alors, sous des hypothèses de régularité, la résolution du PDP dans  $E$  peut se faire en*

- (complexité prouvée)  $O(n2^{3(n-1)^2})$  ;
- (complexité heuristique)  $O(n^2 e^{\omega n} 2^{\omega(n-1)^2})$

opérations arithmétiques dans  $\mathbb{F}_q$ .

Ainsi, si  $p > n$  pour certaines familles de courbes elliptiques, la complexité de la résolution du PDP est divisée par  $2^{3(n-1)}$ . Dans le cas de la caractéristique deux, nous obtenons le résultat suivant.

**Théorème 1.4.** *Soit  $E$  une courbe elliptique définie sur  $\mathbb{F}_{2^{nk}}$  avec  $n > 1$  par l'équation suivante :*

$$E : y^2 + xy = x^3 + \alpha$$

où  $\alpha \in \mathbb{F}_{2^{nk}}$ . Sous des hypothèses de régularité, la résolution du PDP dans  $E$  peut se faire en

- (complexité prouvée)  $O(n2^{3(n-1)(n-2)})$  ;
- (complexité heuristique)  $O(n^2 e^{\omega n} 2^{\omega(n-1)(n-2)})$

opérations arithmétiques dans  $\mathbb{F}_{2^k}$ .

La complexité du PDP en caractéristique deux est donc divisée par  $2^{6(n-1)}$ . La première complexité des deux théorèmes précédents est obtenue en utilisant la complexité de l'algorithme FGLM pour le changement d'ordre. La deuxième complexité est obtenue en utilisant les résultats du Théorème 1.1. Cette dernière reste heuristique puisque nous avons observé que les systèmes à résoudre vérifient bien les hypothèses nécessaires à l'application de ce résultat.

Hormis la résolution des systèmes modélisant le PDP, une étape bloquante dans la résolution du PDP et le calcul des systèmes à résoudre. En effet, la modélisation du PDP sous forme de systèmes polynomiaux requiert le calcul des polynômes de sommation. L'algorithme proposé par Semaev dans [Sem04] pour le calcul des polynômes de sommation requiert l'utilisation de résultants multivariés. En effet, il montre que le  $n$ -ième polynôme de sommation est construit récursivement par

$$S_n(x_1, \dots, x_n) = \text{Res}_X (S_{n-k+1}(x_1, \dots, x_{n-k}, X), S_{k+1}(x_{n-k+1}, \dots, x_n, X)) \quad (1.1)$$

pour tout  $k$  dans  $\{2, \dots, n-2\}$ .

Pour avoir une représentation plus compacte des polynômes de sommation nous pouvons utiliser les symétries des courbes. Cependant comme montré dans [JV13] le calcul de cette représentation nécessite des calculs de base de Gröbner. Une des étapes bloquantes de cet algorithme est que les calculs intermédiaires produisent des objets significativement plus gros que l'entrée ou la sortie.

**Polynômes de sommation par évaluation-interpolation implicite.** Afin de pallier ce problème, nous proposons un algorithme de calcul des polynômes de sommation ne faisant intervenir que des objets dont la taille est bornée par celle de la sortie. De plus, nous montrons que les polynômes de sommation en caractéristique deux admettent une représentation très compacte. En effet, l'étude des symétries des courbes binaires nous permet d'obtenir le résultat suivant.

**Proposition 1.2.** *Soit  $E$  une courbe elliptique définie sur  $\mathbb{F}_{2^k}$  par l'équation*

$$E : y^2 + xy = x^3 + \alpha$$

*où  $\alpha \in \mathbb{F}_{2^k}$ . Le  $n$ -ième polynôme de sommation  $S_n \in \mathbb{K}[x_1, \dots, x_n]$  de  $E$  admet le changement de variables  $\phi$  défini par*

$$\begin{aligned} \phi^{-1} : \mathbb{K}[y_1, \dots, y_n] &\rightarrow \mathbb{K}[x_1, \dots, x_n] \\ f &\mapsto f(e_1(\mathbf{y}), e_2(\mathbf{Z}), \dots, e_{n-1}(\mathbf{Z}), e_n(\mathbf{z})) \end{aligned}$$

*où  $\mathbf{y} = (x_1^2, \dots, x_n^2)$ ,  $\mathbf{z} = (x_1^2 + x_1, \dots, x_n^2 + x_n)$  et  $\mathbf{Z} = (x_1^2 + x_1^4, \dots, x_n^2 + x_n^4)$ .*

Soit  $S'_n \in \mathbb{K}[y_1, \dots, y_n]$  le  $n$ -ième polynôme de sommation exprimé selon le changement de variables  $\phi$  i.e.  $S'_n = \phi(S_n)$ . La représentation considérée entraîne que les polynômes de sommation sont très creux. Ainsi, pour tirer parti de cette représentation, nous proposons un algorithme de calcul des polynômes de sommation par interpolation de polynômes multivariés creux. La difficulté dans l'élaboration d'un tel algorithme est l'évaluation du polynôme  $S'_n$ .

En effet, étant donné un point  $\tilde{\mathbf{y}} = (\tilde{y}_1, \dots, \tilde{y}_n)$  de  $\mathbb{K}^n$  évaluer  $S'_n$  en ce point requiert de trouver le point d'évaluation correspondant du résultant en équation (1.1). Trouver le point d'évaluation du résultant, revient donc à inverser le changement de variables  $\phi$  *i.e.* à résoudre le système suivant

$$\mathcal{S} = \left\{ \begin{array}{c} e_1(f(x_1), \dots, f(x_n)) - \tilde{y}_1 \\ e_2(g(x_1), \dots, g(x_n)) - \tilde{y}_2 \\ \vdots \\ e_{n-1}(g(x_1), \dots, g(x_n)) - \tilde{y}_{n-1} \\ e_n(h(x_1), \dots, h(x_n)) - \tilde{y}_n \end{array} \right\}$$

où  $f(x) = x^2$ ,  $g(x) = x^4 + x^2$  et  $h(x) = x^2 + x$ . Notons que le système  $\mathcal{S}$  est donné par des compositions de polynômes symétriques élémentaires. En utilisant la structure de  $\mathcal{S}$  et les propriétés des corps finis de caractéristique deux, nous montrons que la résolution de  $\mathcal{S}$  peut se faire à l'aide uniquement de factorisations de polynômes univariés.

De plus, nous ne chercherons pas à calculer le résultant en équation (1.1). Puisque nous cherchons uniquement son évaluation, nous évaluerons directement les polynômes  $S_{n-k+1}$  et  $S_{k+1}$ . L'évaluation de  $S'_n$  sera alors donnée par le calcul d'un résultant univarié.

Finalement, notre algorithme de calcul des polynômes de sommation ne requiert que des opérations sur les polynômes univariés (factorisation et résultant). Ainsi nous évitons les calculs coûteux de bases de Gröbner et de résultants multivariés.

## Impacts en pratique

Les algorithmes proposés ont permis de résoudre en pratique des instances encore jamais résolu du PDP, de PoSSo et pour la première fois de calculer au delà du cinquième polynôme de sommation.

- Avant ces travaux, la résolution en pratique du PDP n'était possible que jusqu'à  $n = 4$ .

En caractéristique  $p > n$ , pour  $n = 4$  nous notons une accélération significative de la résolution. En effet nous obtenons un facteur 400 sur la résolution totale du PDP. Nous sommes de plus maintenant capables de résoudre les instances pour  $n = 5$ . À titre d'exemple lorsque  $q = 65521$ , la résolution du PDP (en utilisant FGb [Fau10]) peut s'achever en environ 45 minutes. Une interpolation du nombre total d'opérations requises pour résoudre le ECDLP permet de nous comparer aux attaques génériques. Pour  $n = 5$  si  $q$  est de l'ordre de 64 bits les attaques génériques requièrent de l'ordre de  $2^{160}$  opérations dans  $E(\mathbb{F}_{q^5})$ . Notre attaque améliorée ne nécessite que  $2^{130}$  multiplications de mots de 32 bits.

En caractéristique  $p = 2$ , pour  $n = 4$  et  $q$  de l'ordre de 32 bits, nous notons une accélération de la résolution du PDP d'un facteur environ 5500. Pour ces instances, la méthode dans [Gau09] requiert plus de 10 minutes pour résoudre le PDP. Nos travaux permettent de réduire cette résolution à moins d'une seconde. Pour  $n = 5$ , le PDP peut maintenant être résolu en utilisant MAGMA [BCP97]. Pour  $q$  de l'ordre de 16 bits cette résolution peut se faire en environ 5 minutes.

- Avant ces travaux, nous pouvions calculer au plus le cinquième (respectivement sixième) polynôme de sommation si  $p \neq 2$  (resp.  $p = 2$ ).

En caractéristique  $p = 2$ , la Proposition 1.2 nous donne une représentation très compacte des polynômes de sommation. Cette représentation permet de calculer avec les méthodes usuelles le septième polynôme de sommation en environ 6 minutes. De plus l'algorithme par évaluation-interpolation que nous avons proposé permet le calcul du huitième polynôme de sommation contenant 470369 termes. Par exemple, si  $q$  est de l'ordre de 32 bits, il peut se calculer avec MAGMA en environ 6 heures en utilisant huit coeurs CPU.

- Nous avons également observé que l'algorithme que l'on propose pour la résolution de PoSSo permet des gains notables sur des instances « pire cas » pour cet algorithme. Ces instances sont de la forme  $n$  équations de degré deux en  $n$  variables. Supposons que les systèmes sont à coefficients dans  $\mathbb{F}_{65521}$ . Pour  $n = 11$ , notre algorithme est 1500 fois plus rapide que l'algorithme classique en Figure 1.1. De plus  $n = 11$  sont les dernières instances pour lesquels l'algorithme classique permet la résolution. Notre algorithme permet quant à lui de résoudre les instances jusqu'à  $n = 16$ . Pour  $n = 16$  la résolution de ces systèmes peut maintenant se faire en environ 15 heures en utilisant FGb.

## Perspectives

À l'issue de cette thèse, certaines questions restent à traiter et certains résultats pourraient être améliorés. Nous donnons ci-dessous quelques suggestions.

**Changement d'ordre pour bases de Gröbner.** Les algorithmes de changement d'ordre pour bases de Gröbner ont une complexité polynomiale en le nombre de solutions. Nous avons montré que l'exposant dans cette complexité peut être réduit de trois à  $\omega$ . Les  $n$  matrices de multiplication impliquées dans ces algorithmes ne sont pas dénuées de structure. En particulier elles commutent deux à deux. Ainsi pouvons nous tirer parti de la structure de ces matrices pour obtenir des algorithmes plus efficace ?

**Changement d'ordre rapide et systèmes avec symétries.** Soit un système dont les équations ont un degré fixé (c'est à dire ne dépend pas d'un paramètre). La version déterministe du changement d'ordre rapide s'applique également au cas où la première base de Gröbner est donnée pour l'ordre du degré pondéré lexicographique inverse. Ainsi, cet algorithme de changement d'ordre peut être utilisé dans l'algorithme de résolution de systèmes admettant des symétries.

Lorsque les équations du système en entrée ont un degré non fixé (c'est à dire le degré peut dépendre d'un paramètre tel que le nombre de variables) alors l'algorithme déterministe peut s'appliquer mais sa complexité n'est plus polynomiale en  $D$  avec exposant  $\omega$ . La version probabiliste du changement d'ordre rapide, permet de traiter le cas de tels systèmes. Cependant en général, elle nécessite l'application d'un changement de variables linéaire. Ce changement de variables a pour effet de casser la structure quasi-homogène induite par les symétries. Ainsi dans le cas d'équations ayant un degré non fixé, les algorithmes de changement d'ordre rapides ne peuvent pas toujours être utilisés dans l'algorithme de résolution des systèmes admettant des symétries. Peut on mettre en place des algorithmes de changement d'ordre rapide applicable dans ce contexte ?

**Attaque algébrique du logarithme discret sur les courbes.** Nous avons mis en évidence des familles de courbes elliptiques ayant une structure particulière. Ces structures induisent des symétries dans la résolution des systèmes polynomiaux sous-jacents à l'attaque par calcul d'indice du ECDLP de Gaudry [Gau09]. Elles ont donc permis d'en accélérer la résolution en théorie et en pratique.

Comme mentionné en début d'introduction, la version de l'attaque par calcul d'indice de Diem [Die11a, Die11b] a permis de mettre en évidence des familles de courbes dans lesquelles le ECDLP peut être résolu en temps sous-exponentiel. Une question naturelle est donc de déterminer si l'utilisation des symétries peut s'appliquer dans cette attaque ? De plus il serait intéressant d'étendre ce type de résultats aux cas des courbes hyper-elliptiques de genre 2. Par exemple, l'utilisation des symétries peut-elle être mise en oeuvre dans l'attaque de Nagao [Nag10] ?

**Comptage de points.** Un autre problème fondamentale de la cryptologie sur les courbes est le comptage de points dans la jacobienne des courbes hyper-elliptiques. En 1985 Schoof [Sch85] a introduit le premier algorithme en temps polynomial de comptage de points d'une courbe elliptique définie sur un corps fini. Plus tard, cet algorithme fut étendu pour diverses variétés abéliennes définies sur des corps finis [AH96, GS12, HI98, Pil90]. Une des étapes principales de ces algorithmes consiste à trouver une représentation la plus compacte possible de la  $\ell$ -torsion de la courbe. La représentation considérée est en réalité un système d'équations polynomiales dont les solutions sont l'ensemble des éléments de la  $\ell$ -torsion. L'algorithme de comptage de points se ramène alors à des opérations dans l'algèbre quotient. Les symétries des courbes peuvent-elles être utilisées pour optimiser ces algorithmes ? En particulier permettent-elles de trouver des représentations plus compactes de la  $\ell$ -torsion et d'accélérer l'arithmétique dans l'algèbre quotient correspondante ?

## Organisation du manuscrit

Ce manuscrit se divise en deux parties. La première portant sur la résolution de systèmes polynomiaux par base de Gröbner est constituée de trois chapitres.

**Chapitre 2 :** Ce chapitre introduit la notion de base de Gröbner et certaines de leurs propriétés. Nous présentons également dans ce chapitre les algorithmes existants et leur complexité pour le calcul de base de Gröbner, le changement d'ordre et la résolution de PoSSo par base de Gröbner.

**Chapitre 3 :** Dans ce chapitre nous présentons l'algorithme pour la résolution de systèmes polynomiaux admettant un changement de variables polynomial. En particulier, nous obtiendrons la Proposition 1.1 et le Théorème 1.2. Nous rappelons quelques définitions et résultats de la théorie des invariants nécessaires à l'appréhension de la résolution des systèmes avec symétries. Nous présenterons également comment appliquer les résultats précédemment obtenus dans le chapitre aux systèmes possédant des symétries.

Les résultats présentés dans ce chapitre sont un travail en commun avec Jean-Charles Faugère, Pierrick Gaudry et Guénaél Renault. Un cas particulier de ces résultats a été publié dans [FGHR13b].

**Chapitre 4 :** Ce chapitre présente de nouveaux algorithmes de changement d'ordre pour base de Gröbner ainsi qu'un nouvel algorithme de résolution de PoSSo par base de Gröbner. En particulier, nous obtiendrons le Théorème 1.1.

Les résultats présentés dans ce chapitre sont un travail en commun avec Jean-Charles Faugère, Pierrick Gaudry et Guénaël Renault. Une version préliminaire de ces travaux a été publiée dans [FGHR12b] et présentée sous forme de poster à la conférence ISSAC 2012 [FGHR12a] et a reçu le prix du meilleur poster. L'ensemble de ces résultats a fait l'objet d'une pré-publication [FGHR13a].

La deuxième partie du manuscrit porte sur la cryptographie sur les courbes elliptiques. Cette partie est également divisée en trois chapitres.

**Chapitre 5 :** Dans ce chapitre nous donnons une définition des courbes elliptiques ainsi que différentes représentations de courbes. Nous rappelons quelques attaques génériques pour la résolution du DLP. Nous introduisons ensuite les polynômes de sommation de Semaev. Puis, nous présentons également l'attaque par calcul d'indice de Gaudry pour la résolution du DLP sur les courbes.

**Chapitre 6 :** Dans ce chapitre nous mettons en évidence des familles de courbes elliptiques possédant des symétries particulières. Nous montrerons comment ces symétries impactent la résolution du PDP. En particulier, nous obtiendrons le Théorème 1.3.

Les résultats présentés dans ce chapitre sont un travail en commun avec Jean-Charles Faugère, Pierrick Gaudry et Guénaël Renault. Une version préliminaire de ces travaux a été publiée dans [FGHR12b]. L'ensemble des résultats a fait l'objet d'une deuxième publication [FGHR13b].

**Chapitre 7 :** Dans ce dernier chapitre, nous présentons comment les symétries des courbes en caractéristique deux permettent d'obtenir une représentation compacte des polynômes de sommation. Nous présentons également dans ce chapitre un algorithme pour le calcul de ces polynômes par évaluation-interpolation. À titre d'application nous donnons une description détaillée du calcul du huitième polynôme de sommation. Nous présenterons également comment les symétries des courbes en caractéristique deux permettent d'améliorer la résolution du PDP. En particulier, nous obtiendrons la Proposition 1.2 et le Théorème 1.4.

Les résultats présentés dans ce chapitre sont un travail (toujours en cours) en commun avec Jean-Charles Faugère, Antoine Joux, Guénaël Renault et Vanessa Vitse.



## Part I

# Gröbner Bases and Polynomial Systems Solving



## CHAPTER 2

# Gröbner bases

---

### Contents

|                                                                                        |           |
|----------------------------------------------------------------------------------------|-----------|
| <b>2.1 Preliminaries</b>                                                               | <b>24</b> |
| 2.1.1 Ideals and varieties                                                             | 24        |
| 2.1.2 Gröbner bases: definition and general properties                                 | 26        |
| 2.1.3 Properties of degree reverse lexicographical Gröbner bases                       | 31        |
| 2.1.4 Properties of lexicographical Gröbner bases                                      | 37        |
| 2.1.5 What means solving?                                                              | 41        |
| <b>2.2 Gröbner bases algorithms</b>                                                    | <b>41</b> |
| 2.2.1 Lazard's algorithm                                                               | 41        |
| 2.2.2 Efficient algorithms for Gröbner bases: $F_4$ and $F_5$                          | 43        |
| <b>2.3 Change of ordering algorithms</b>                                               | <b>45</b> |
| 2.3.1 The FGLM algorithm                                                               | 45        |
| 2.3.2 Sparse change of ordering for Shape Position ideals: the probabilistic algorithm | 48        |
| 2.3.3 Sparse change of ordering for Shape Position ideals: the deterministic algorithm | 50        |
| 2.3.4 Computation of $T_n$                                                             | 53        |
| <b>2.4 Complexity</b>                                                                  | <b>53</b> |
| 2.4.1 Gröbner bases algorithms                                                         | 53        |
| 2.4.2 Change of ordering                                                               | 58        |
| 2.4.3 Polynomial systems solving                                                       | 61        |

---

In this chapter we present all the general theoretical and algorithmic backgrounds about Gröbner bases required in this thesis. First, we give general definitions and results about Gröbner bases. Then, two sections are devoted to algorithms to compute Gröbner bases or to change their monomial ordering. Finally, the complexity of these algorithms and polynomial systems solving using Gröbner bases are studied in the last section of this chapter.

In the whole thesis,  $\omega$  denotes the exponent in the complexity of multiplying two dense matrices. In particular from [VW12] we have  $2 \leq \omega < 2.3727$ . Moreover, unless specified the notation  $\tilde{O}(f(n))$  always means that we neglect logarithm factors in  $n$  *i.e.* factors of the form  $\log(n)^k$ . Hence, in the case where  $n = d^m$  this notation means that we neglect logarithm factors in  $n$  and  $d$  and polynomial factors in  $m$ .

## 2.1 Preliminaries

In this section, we recall general definitions and results required in this thesis about ideals and Gröbner bases. For a more thorough reading on the subject see [CLO07] for an introduction on computational commutative algebra. From now on,  $\mathbb{K}$  denotes a field.

### 2.1.1 Ideals and varieties

To avoid ambiguities we first recall what we call monomials and terms. A monomial in the indeterminates  $x_1, \dots, x_n$  is a product of the form  $x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$ , denoted  $x^\alpha$ , where  $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$ . A polynomial  $f$  in the indeterminates  $x_1, \dots, x_n$  with coefficients in  $\mathbb{K}$  is a finite linear combination of monomials in  $x_1, \dots, x_n$  with coefficients in  $\mathbb{K}$  i.e.  $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha$  where  $c_\alpha \in \mathbb{K}$  and the cardinality of  $\{\alpha \mid c_\alpha \neq 0\}$  is finite. The set of all polynomials in  $x_1, \dots, x_n$  with coefficients in  $\mathbb{K}$  is a polynomial ring denoted  $\mathbb{K}[x_1, \dots, x_n]$ . A term of  $\mathbb{K}[x_1, \dots, x_n]$  is a product of a coefficients in  $\mathbb{K}$  and a monomial in  $\mathbb{K}[x_1, \dots, x_n]$ .

**Definition 2.1** (Degree). *Let  $m = x^\alpha$  be a monomial in  $\mathbb{K}[x_1, \dots, x_n]$ . The degree of  $m$  is defined by*

$$\deg(m) = |\alpha| = \sum_{i=1}^n \alpha_i.$$

*The degree of  $m$  in the variable  $x_i$  is defined by  $\deg_{x_i}(m) = \alpha_i$ . Let  $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha$  be a non-zero polynomial in  $\mathbb{K}[x_1, \dots, x_n]$ , the degree of  $f$  is defined by*

$$\deg(f) = \max \{|\alpha| \mid c_\alpha \neq 0\}$$

*and its degree in the variable  $x_i$  is defined by*

$$\deg_{x_i}(f) = \max \{\alpha_i \mid c_\alpha \neq 0\}.$$

*By convention,  $\deg(0) = -1$ .*

In this thesis, we consider only polynomial ideals of which the definition is given below.

**Definition 2.2** (Polynomial ideal). *A subset  $\mathcal{I}$  of the polynomial ring  $\mathbb{K}[x_1, \dots, x_n]$  is an ideal if it satisfies the following conditions:*

1.  $0 \in \mathcal{I}$ ;
2. if  $f, g \in \mathcal{I}$  then  $f + g \in \mathcal{I}$ ;
3. if  $f \in \mathcal{I}$  then for all  $h \in \mathbb{K}[x_1, \dots, x_n]$ ,  $hf \in \mathcal{I}$ .

Actually, polynomial ideals are defined thanks to a set of polynomials. More precisely, a set of polynomials  $\{f_1, \dots, f_s\}$  of  $\mathbb{K}[x_1, \dots, x_n]$  defines the ideal

$$\langle f_1, \dots, f_s \rangle = \left\{ \sum_{i=1}^s h_i f_i \mid \text{for all } h_1, \dots, h_s \in \mathbb{K}[x_1, \dots, x_n] \right\}$$

of  $\mathbb{K}[x_1, \dots, x_n]$ . Thus, assuming we want to solve a polynomial system we are then interested in the set of the zeroes of the ideal it generates that we call variety.

**Definition 2.3** (Affine variety). Let  $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ , we define the  $\mathbb{L}$ -affine variety associated to  $f_1, \dots, f_s$  to be the set

$$\mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s) = \{(a_1, \dots, a_n) \in \mathbb{L}^n \mid f_i(a_1, \dots, a_n) = 0 \text{ for } i = 1, \dots, s\} \subset \mathbb{L}^n$$

where  $\mathbb{L}$  is an extension of  $\mathbb{K}$ . When  $\mathbb{L} = \overline{\mathbb{K}}$  the set  $\mathbf{V}_{\overline{\mathbb{K}}}(f_1, \dots, f_s)$  is called the affine variety. Similarly, the  $\mathbb{L}$ -affine variety of an ideal  $\mathcal{I}$  of  $\mathbb{K}[x_1, \dots, x_n]$  is the set

$$\mathbf{V}_{\mathbb{L}}(\mathcal{I}) = \{(a_1, \dots, a_n) \in \mathbb{L}^n \mid f(a_1, \dots, a_n) = 0 \text{ for all } f \in \mathcal{I}\} \subset \mathbb{L}^n.$$

When  $\mathbb{L} = \mathbb{K}$  we denote  $\mathbf{V}_{\mathbb{K}}(f_1, \dots, f_s)$  (respectively  $\mathbf{V}_{\mathbb{K}}(\mathcal{I})$ ) by  $\mathbf{V}(f_1, \dots, f_s)$  (respectively  $\mathbf{V}(\mathcal{I})$ ).

The property that  $\mathbf{V}_{\mathbb{L}}(\mathcal{I})$  is an affine variety is due to the Hilbert basis Theorem.

**Theorem 2.4** (Hilbert basis Theorem). Any ideal  $\mathcal{I}$  of  $\mathbb{K}[x_1, \dots, x_n]$  has a finite generating set  $\{f_1, \dots, f_s\} \subset \mathbb{K}[x_1, \dots, x_n]$ . That is to say  $\mathcal{I} = \langle f_1, \dots, f_s \rangle$ .

By consequence, if  $f_1, \dots, f_s$  is a basis of  $\mathcal{I}$  we have  $\mathbf{V}_{\mathbb{L}}(\mathcal{I}) = \mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s)$  which defines  $\mathbf{V}_{\mathbb{L}}(\mathcal{I})$  as an affine variety. For more details see [CLO07, p. 75-80]. Conversely, we can define the ideal associated to an affine variety.

**Definition 2.5.** Let  $\mathbf{V} \subset \mathbb{K}^n$  be an affine variety. The ideal of  $\mathbf{V}$  is defined as

$$I(\mathbf{V}) = \{f \in \mathbb{K}[x_1, \dots, x_n] \mid f(a_1, \dots, a_n) = 0 \text{ for all } (a_1, \dots, a_n) \in \mathbf{V}\}.$$

Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ , it is straightforward to see that  $\mathcal{I} \subset I(\mathbf{V}(\mathcal{I}))$ . However, the equality may not occur.

**Example 2.6.** Let  $\mathcal{I} = \langle x_1^2, x_2^2 \rangle$  be an ideal of  $\mathbb{K}[x_1, x_2]$ . The affine variety of  $\mathcal{I}$  is given by  $\mathbf{V}_{\overline{\mathbb{K}}}(\mathcal{I}) = \{(0, 0)\}$ . Hence, the ideal  $I(\mathbf{V}_{\overline{\mathbb{K}}}(\mathcal{I})) = \langle x_1, x_2 \rangle \supset \mathcal{I}$ .

Nevertheless, thanks to the Hilbert's Nullstellensatz we can characterize the ideals such that the equality holds.

**Theorem 2.7** (Hilbert's Nullstellensatz). Let  $\mathbb{K}$  be an algebraically closed field and let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . For all  $f \in I(\mathbf{V}(\mathcal{I}))$ , there exists  $m \in \mathbb{N}$  such that  $f^m \in \mathcal{I}$ .

Consequently, the ideals satisfying  $\mathcal{I} = I(\mathbf{V}(\mathcal{I}))$  are radical ideals whose the definition is given below.

**Definition 2.8** (Radical ideal). Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ ;  $\mathcal{I}$  is said to be radical if for all  $f \in \mathbb{K}[x_1, \dots, x_n]$ , if there exists an integer  $m \geq 1$  such that  $f^m \in \mathcal{I}$  then  $f \in \mathcal{I}$ .

Since  $\mathcal{I} \subset I(\mathbf{V}(\mathcal{I}))$  the Hilbert's Nullstellensatz implies that if  $\mathcal{I}$  is a radical ideal then  $\mathcal{I} = I(\mathbf{V}(\mathcal{I}))$ . The Strong Hilbert's Nullstellensatz even shows that  $I(\mathbf{V}(\mathcal{I}))$  is the radical of  $\mathcal{I}$ .

**Definition 2.9** (Radical of an ideal). Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal. The radical of  $\mathcal{I}$  denoted  $\sqrt{\mathcal{I}}$  is the set  $\sqrt{\mathcal{I}} = \{f \mid f^m \in \mathcal{I} \text{ for some } m \geq 1\}$ .

The following proposition characterizes the sets of polynomials that have the same affine variety.

**Proposition 2.10.** *Let  $f_1, \dots, f_s$  and  $g_1, \dots, g_r$  be polynomials in  $\mathbb{K}[x_1, \dots, x_n]$ . If the two ideals  $\langle f_1, \dots, f_s \rangle$  and  $\langle g_1, \dots, g_r \rangle$  are equal then for any extension  $\mathbb{L}$  of  $\mathbb{K}$  we have  $\mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s) = \mathbf{V}_{\mathbb{L}}(g_1, \dots, g_r)$ .*

*Proof.* Let  $a \in \mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s)$  by definition we have  $f_i(a) = 0$  for  $i = 1, \dots, s$ . By hypothesis  $g_j \in \langle f_1, \dots, f_s \rangle$  for  $j = 1, \dots, r$ . Hence, for any  $j \in \{1, \dots, r\}$  we have  $g_j = \sum_{i=1}^s h_i f_i$  for some  $h_1, \dots, h_s \in \mathbb{K}[x_1, \dots, x_n]$ . By consequence,  $g_j(a) = \sum_{i=1}^s h_i(a) f_i(a) = 0$  for  $j = 1, \dots, r$ . Then,  $a \in \mathbf{V}_{\mathbb{L}}(g_1, \dots, g_r)$  and  $\mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s) \subset \mathbf{V}_{\mathbb{L}}(g_1, \dots, g_r)$ . In the same way, we can show that  $\mathbf{V}_{\mathbb{L}}(g_1, \dots, g_r) \subset \mathbf{V}_{\mathbb{L}}(f_1, \dots, f_s)$  which finishes the proof.  $\square$

This result will be useful for polynomial systems solving. Indeed, to solve a polynomial system of equations the usual strategy is to find a new set of polynomials having same zeroes or equivalently generating the same ideal from which the solutions are much easier to find.

In this thesis, we focus on ideals of dimension zero of which the definition is given below.

**Definition 2.11** (Zero-dimensional ideal). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . The ideal  $\mathcal{I}$  is of dimension zero if the affine variety  $\mathbf{V}_{\mathbb{K}}(\mathcal{I})$  is of dimension zero, that is to say  $\#\mathbf{V}_{\mathbb{K}}(\mathcal{I}) < \infty$ .*

### 2.1.2 Gröbner bases: definition and general properties

A Gröbner basis is defined with respect to a monomial ordering. We will see that depending on the monomial ordering Gröbner bases can have different properties. We first recall the definition of monomial ordering.

**Definition 2.12** (Total order). *A total order, denoted  $\prec$ , on some set  $X$  is a binary relation on the elements of  $X$  which is*

- *transitive: for all  $a, b, c \in X$  if  $a \prec b$  and  $b \prec c$  then  $a \prec c$ ;*
- *antisymmetric: for all  $a, b \in X$  if  $a \prec b$  and  $b \prec a$  then  $a = b$ ;*
- *total: for all  $a, b \in X$   $a \prec b$  or  $b \prec a$ .*

**Definition 2.13** (Monomial ordering). *A monomial ordering  $>$  on  $\mathbb{K}[x_1, \dots, x_n]$  is a total order on the set of monomials of  $\mathbb{K}[x_1, \dots, x_n]$  (or equivalently on  $\mathbb{N}^n$ ) which satisfies the following conditions:*

1. *If  $x^\alpha > x^\beta$  then for any  $\gamma \in \mathbb{N}^n$  we have  $x^{\alpha+\gamma} > x^{\beta+\gamma}$ ;*
2.  *$>$  is a well-ordering on  $\mathbb{N}^n$  i.e. every nonempty subset of  $\mathbb{N}^n$  has a smallest element w.r.t.  $>$ .*

We now define the two most commonly used monomial orderings for Gröbner bases computations which are also the only ones used in this thesis.

**Definition 2.14** (Lexicographical ordering). *Let  $x^\alpha, x^\beta \in \mathbb{K}[x_1, \dots, x_n]$  be two monomials. The lexicographical ordering, denoted  $>_{\text{lex}}$ , is defined by  $x_1 >_{\text{lex}} \dots >_{\text{lex}} x_n$  and  $x^\alpha >_{\text{lex}} x^\beta$  if and only if there exists  $i \in \{1, \dots, n\}$  such that  $\alpha_j = \beta_j$  for  $j = 1, \dots, i-1$  and  $\alpha_i > \beta_i$ .*

The next monomial ordering is a graded ordering. By consequence, we need to fix a grading for  $\mathbb{K}[x_1, \dots, x_n]$ .

**Definition 2.15** (Graded ring). *The ring  $R$  is graded if there exists a grading  $\Gamma : R \rightarrow \mathbb{N}$  such that*

$$R = \bigoplus_{n \in \mathbb{N}} R_n = R_0 \oplus R_1 \oplus R_2 \oplus \dots$$

where  $R_n = \{e \in R \mid \Gamma(e) = n\}$  is an additive subgroup of  $R$  and  $R_i R_j \subset R_{i+j}$ .

Polynomial rings are graded rings and the two commonly used gradings and the only ones in this thesis are the usual degree, see Definition 2.1, or the weighted degree. That is to say if  $R = \mathbb{K}[x_1, \dots, x_n]$ ,  $R_d$  is the  $\mathbb{K}$ -vector space generated by all monomials of  $R$  of (weighted) degree  $d$ .

**Definition 2.16** (Homogeneous/affine ideal). *Once a grading is fixed, we say that a polynomial is homogeneous if all its monomials are of same graduation. Otherwise, the polynomial is called an affine polynomial. A homogeneous ideal is an ideal such that there exists a basis of it consisting of homogeneous polynomials. Otherwise, it is called an affine ideal.*

In the literature, a polynomial which is homogeneous for a weighted degree is usually said quasi-homogeneous but we do not use this terminology here. It is important to note that the homogeneity of a polynomial depends on the grading.

**Definition 2.17** (Weighted degree). *Let  $m = x^\alpha$  be a monomial in  $\mathbb{K}[x_1, \dots, x_n]$ . Given a weights system  $(w_1, \dots, w_n)$  the weighted degree of  $m$  is defined by*

$$\text{wdeg}(m) = |\alpha|_w = \sum_{i=1}^n w_i \alpha_i.$$

*Let  $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha$  be a non-zero polynomial in  $\mathbb{K}[x_1, \dots, x_n]$ , the weighted degree of  $f$  is defined by*

$$\text{wdeg}(f) = \max \{|\alpha|_w \mid c_\alpha \neq 0\}.$$

**Definition 2.18** (Graded reverse lexicographical ordering). *Let  $x^\alpha$  and  $x^\beta$  be two monomials of  $\mathbb{K}[x_1, \dots, x_n]$ . Given a grading  $\Gamma$  on  $\mathbb{K}[x_1, \dots, x_n]$ , the graded reverse lexicographical ordering, denoted  $>_{\text{grl}}$ , is defined by  $x_1 >_{\text{grl}} \dots >_{\text{grl}} x_n$  and  $x^\alpha >_{\text{grl}} x^\beta$  if and only if  $\Gamma(x^\alpha) > \Gamma(x^\beta)$  or  $\Gamma(x^\alpha) = \Gamma(x^\beta)$  and there exists  $i \in \llbracket 1; n \rrbracket$  such that  $\alpha_j = \beta_j$  for  $j = i+1, \dots, n$  and  $\alpha_i < \beta_i$ .*

When the grading  $\Gamma$  is the usual degree we denote the corresponding monomial ordering  $>_{\text{drl}}$  for degree reverse lexicographical ordering, DRL for short. Whereas, when the grading  $\Gamma$  is the weighted degree we denote the corresponding monomial ordering  $>_{\text{wdrl}}$  for weighted degree reverse lexicographical ordering, WDRL for short.

**Definition 2.19** (Leading term). *Let  $f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha \in \mathbb{K}[x_1, \dots, x_n]$  be a non-zero polynomial. The leading term of  $f$  w.r.t. the monomial ordering  $>$  denoted  $\text{LT}_>(f)$  is defined by  $\text{LT}_>(f) = c_\alpha x^\alpha$  such that  $c_\alpha \neq 0$  and for all  $\beta \in \mathbb{N}^n$  such that  $c_\beta \neq 0$  we have  $x^\alpha > x^\beta$ .*

From the leading term of polynomials one can construct a monomial ideal as in the following definition.

**Definition 2.20** (Initial ideal). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . Given a monomial ordering  $>$ , we denote by  $\text{in}_>(\mathcal{I})$  the initial ideal of  $\mathcal{I}$  defined by*

$$\text{in}_>(\mathcal{I}) = \{\text{LT}_>(f) \mid f \in \mathcal{I}\}.$$

**Proposition 2.21** ([CLO07] page 76). *Let  $\mathcal{I}$  be an ideal in  $\mathbb{K}[x_1, \dots, x_n]$  and  $>$  a monomial ordering. There exist  $g_1, \dots, g_s \in \mathcal{I}$  such that  $\text{in}_>(\mathcal{I}) = \langle \text{LT}_>(g_1), \dots, \text{LT}_>(g_s) \rangle$ .*

Gröbner bases are to polynomials what row echelon form is to linear algebra. Once a monomial ordering is fixed, the corresponding reduced Gröbner basis is unique and allows to obtain the canonical basis of an ideal

**Definition 2.22** (Gröbner basis). *Given an ideal  $\mathcal{I}$  of  $\mathbb{K}[x_1, \dots, x_n]$  and a monomial ordering  $>$ , a finite subset  $\mathcal{G}_> = \{g_1, \dots, g_s\}$  of  $\mathcal{I}$  is a Gröbner basis w.r.t.  $>$  of  $\mathcal{I}$  if  $\text{in}_>(\mathcal{I}) = \langle \text{LT}_>(g_1), \dots, \text{LT}_>(g_s) \rangle$ . The Gröbner basis  $\mathcal{G}_>$  is the unique reduced Gröbner basis of  $\mathcal{I}$  if  $g_1, \dots, g_s$  are monic polynomials and for any  $g_i \in \mathcal{G}_>$  all the terms in  $g_i$  are not divisible by a leading term of  $g_j$  for all  $g_j \in \mathcal{G}_>$  such that  $j \neq i$ .*

From now on, unless indicated otherwise we consider only reduced Gröbner bases so we omit the term reduced. Following Definition 2.22, Proposition 2.21 implies the following result.

**Corollary 2.23** ([CLO07] page 77). *Let  $>$  be a monomial ordering. Every nonzero ideal  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  has a Gröbner basis w.r.t.  $>$ . Moreover, any Gröbner basis of  $\mathcal{I}$  is a basis of  $\mathcal{I}$ .*

The property that once the monomial ordering is fixed, any ideal of  $\mathbb{K}[x_1, \dots, x_n]$  has a unique reduced Gröbner basis is shown in [CLO07, p. 92].

From the previous corollary, any Gröbner basis of an ideal  $\mathcal{I}$  is a basis of  $\mathcal{I}$ . Hence, from Proposition 2.10 the affine variety of  $\mathcal{I}$  and the affine variety of its Gröbner basis are the same. Moreover, we will see that there exist efficient algorithms to compute Gröbner bases and finding the affine variety of a Gröbner basis can be much easier than finding the variety of an arbitrary basis of  $\mathcal{I}$ . This is why Gröbner bases are a fundamental tool for polynomial systems solving.

We now introduced some general properties of Gröbner bases. We mean by general that these properties are true for any monomial ordering. Specific properties of Gröbner bases for particular orderings are given in the two next sections.

**Proposition 2.24** ([CLO07] page 82). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . Let  $\mathcal{G}_>$  be a Gröbner basis of  $\mathcal{I}$  w.r.t. the monomial ordering  $>$ . Let  $f$  be a polynomial of  $\mathbb{K}[x_1, \dots, x_n]$  there exists a unique polynomial  $r$  such that*

- *there exists  $h \in \mathcal{I}$  such that  $f = h + r$ ;*
- *no term of  $r$  is divisible by  $\text{LT}_>(g)$  for any  $g \in \mathcal{G}_>$ .*

*The polynomial  $r$  is called the normal form of  $f$  and is denoted  $\text{NF}_>(f)$ .*

Note that the normal form map is a linear map. Indeed, let  $f_1$  and  $f_2$  in  $\mathbb{K}[x_1, \dots, x_n]$  we have  $f_i = h_i + \text{NF}_>(f_i)$  with  $h_i \in \mathcal{I}$  for  $i = 1, 2$ . It is clear that  $\text{NF}_>(f_1) + \text{NF}_>(f_2)$  satisfies the two conditions of Proposition 2.24 for the polynomial  $f_1 + f_2$ . Hence,  $\text{NF}_>(f_1 + f_2) = \text{NF}_>(f_1) + \text{NF}_>(f_2)$ . Moreover, for any  $c \in \mathbb{K}$  and any  $f \in \mathbb{K}[x_1, \dots, x_n]$  we have  $\text{NF}_>(cf) = c \cdot \text{NF}_>(f)$ .

Given a Gröbner basis of an ideal  $\mathcal{I}$ , the normal form of a polynomial allows to decide if this polynomial is in  $\mathcal{I}$  or not. Indeed, we have the following result.

**Corollary 2.25** ([CLO07] page 82). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$  and let  $f$  be a polynomial in  $\mathbb{K}[x_1, \dots, x_n]$ . Then,  $f \in \mathcal{I}$  if and only if  $\text{NF}_{>}(\mathcal{I}) = 0$  for any monomial ordering  $>$ .*

The division algorithm in [CLO07, p.61-67] gives a way to compute the normal form of a polynomial given the corresponding Gröbner basis. In the case of ideals of dimension zero computing normal forms can be done by using linear algebra techniques.

First, we need to define the quotient of the polynomial ring  $\mathbb{K}[x_1, \dots, x_n]$  by one of its ideal  $\mathcal{I}$ . For this purpose, we need an equivalence relation on the ring  $\mathbb{K}[x_1, \dots, x_n]$ .

**Definition 2.26** (Congruence modulo an ideal). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$  and let  $f, g$  be two polynomials in  $\mathbb{K}[x_1, \dots, x_n]$ . We say that  $f$  and  $g$  are congruent modulo  $\mathcal{I}$  denoted  $f \equiv g \pmod{\mathcal{I}}$  if  $f - g \in \mathcal{I}$ .*

The congruence modulo  $\mathcal{I}$  is an equivalence relation on  $\mathbb{K}[x_1, \dots, x_n]$  (see for instance [CLO07, page 221]). Hence, it allows to construct the quotient of a polynomial ring w.r.t. one of its ideal.

**Definition 2.27** (Quotient ring). *The quotient of  $\mathbb{K}[x_1, \dots, x_n]$  modulo one of its ideal  $\mathcal{I}$ , denoted  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ , is the set of equivalence classes of the congruence modulo  $\mathcal{I}$ . That is to say*

$$\mathbb{K}[x_1, \dots, x_n]/\mathcal{I} = \{[f] : f \in \mathbb{K}[x_1, \dots, x_n]\}$$

where  $[f]$  denotes the class of  $f$  defined by the set of polynomials  $g \in \mathbb{K}[x_1, \dots, x_n]$  such that  $f \equiv g \pmod{\mathcal{I}}$ .

A fundamental result states that the quotient ring associated to an ideal  $\mathcal{I}$  is a  $\mathbb{K}$ -vector space of known basis. More precisely, we have the following result.

**Proposition 2.28** ([CLO07] page 232). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal and let  $>$  be a monomial ordering. The quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  is isomorphic as a  $\mathbb{K}$ -vector space to  $\text{Span}(x^\alpha \mid x^\alpha \notin \text{in}_{>}(\mathcal{I}))$ .*

**Notation 2.29.** *We denote by  $B = \{x^\alpha \mid x^\alpha \notin \text{in}_{>}(\mathcal{I})\}$  the canonical basis w.r.t.  $>$  of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  seen as a  $\mathbb{K}$ -vector space.*

When the ideal  $\mathcal{I}$  is of dimension zero the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  seen as a  $\mathbb{K}$ -vector space is of finite dimension  $\mathcal{D}_{\mathcal{I}}$ . In that case, we denote by  $\mathbb{V}_{>}(\mathcal{I})$  the representation of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  as a subset of  $\mathbb{K}^{\mathcal{D}_{\mathcal{I}}}$ .

**Definition 2.30** (Degree of an ideal). *Let  $\mathcal{I}$  be an ideal of dimension zero of  $\mathbb{K}[x_1, \dots, x_n]$ . We call the dimension of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  the degree of  $\mathcal{I}$  and we denote it  $\mathcal{D}_{\mathcal{I}}$ .*

The degree of an ideal is related to its number of zeroes by the following result.

**Proposition 2.31** ([CLO07] p. 234-236). *Let  $\mathcal{I}$  be an ideal of dimension zero. We have  $\#\mathbb{V}_{\overline{\mathbb{K}}}(\mathcal{I}) \leq \mathcal{D}_{\mathcal{I}}$ . The equality holds if the ideal is radical on  $\overline{\mathbb{K}}$ . More generally, the degree of  $\mathcal{I}$  is the number of solutions of  $\mathcal{I}$  in an algebraic closure of  $\mathbb{K}$  counted with multiplicities.*

By consequence, the canonical basis  $B$  is of finite cardinality  $\mathcal{D}_{\mathcal{I}}$  and we denote its element in increasing order i.e.  $B = \{\epsilon_{\mathcal{D}_{\mathcal{I}}} > \dots > \epsilon_1 = 1\}$ . From now on, we consider only ideals of dimension zero.

We recall that  $x^\alpha$  denotes a monomial of  $\mathbb{K}[x_1, \dots, x_n]$  with  $\alpha \in \mathbb{N}^n$ . The isomorphism between  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  and  $\mathbb{V}_>(\mathcal{I})$  is constructed from the normal form map. Indeed, it is easy to see that for any monomial ordering  $>$ ,  $g \in [f]$  if and only if  $\text{NF}_>(g) = \text{NF}_>(f)$ ; in particular  $\text{NF}_>(f) \in [f]$ . Moreover, from its definition, the normal form of a polynomial  $f$  w.r.t. the monomial ordering contains only monomials in  $B$  i.e. for any  $f \in \mathbb{K}[x_1, \dots, x_n]$ ,  $\text{NF}_>(f) = \sum_{i=1}^{\mathcal{D}_{\mathcal{I}}} c_i \epsilon_i$  with  $c_i \in \mathbb{K}$ . Hence, the isomorphism  $\Phi$  is then defined by

$$\begin{aligned} \Phi : \mathbb{K}[x_1, \dots, x_n]/\mathcal{I} &\rightarrow \mathbb{V}_>(\mathcal{I}) \\ [f] &\mapsto (c_1, \dots, c_{\mathcal{D}_{\mathcal{I}}}) \text{ with } \text{NF}_>(f) = \sum_{i=1}^{\mathcal{D}_{\mathcal{I}}} c_i \epsilon_i \end{aligned} \quad (2.1)$$

By abusing the notation, in the following it may be that we apply  $\Phi$  directly on  $\mathbb{K}[x_1, \dots, x_n]$  instead on  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . In that case, for any  $m \in \mathbb{K}[x_1, \dots, x_n]$ ,  $\Phi(m)$  denotes  $\Phi([m])$  and  $\Phi^{-1}(\mathbf{v})$  denotes the unique normal form of the polynomials in the class  $\Phi^{-1}(\mathbf{v})$  i.e.  $\Phi^{-1}(\mathbf{v})$  may denote  $\sum_{i=1}^{\mathcal{D}_{\mathcal{I}}} v_i \epsilon_i$  instead of  $\left[ \sum_{i=1}^{\mathcal{D}_{\mathcal{I}}} v_i \epsilon_i \right]$  with  $\mathbf{v} = (v_1, \dots, v_{\mathcal{D}_{\mathcal{I}}})$ .

Let  $\lambda_i$  be the linear map corresponding to the multiplication by  $x_i$  in  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  i.e.

$$\begin{aligned} \lambda_i : \mathbb{K}[x_1, \dots, x_n]/\mathcal{I} &\rightarrow \mathbb{K}[x_1, \dots, x_n]/\mathcal{I} \\ \left[ \sum_{j=1}^{\mathcal{D}_{\mathcal{I}}} c_j \epsilon_j \right] &\mapsto \left[ \text{NF}_>\left(x_i \sum_{j=1}^{\mathcal{D}_{\mathcal{I}}} c_j \epsilon_j\right) \right] \end{aligned}$$

Hence, if  $\Lambda_i$  is the linear map corresponding to the multiplication by  $x_i$  in  $\mathbb{V}_>(\mathcal{I})$  then  $\Lambda_i$  is given by

$$\begin{aligned} \Lambda_i : \mathbb{V}_>(\mathcal{I}) &\rightarrow \mathbb{V}_>(\mathcal{I}) \\ (c_1, \dots, c_{\mathcal{D}_{\mathcal{I}}}) &\mapsto \Phi\left(x_i \sum_{j=1}^{\mathcal{D}_{\mathcal{I}}} c_j \epsilon_j\right) \end{aligned}$$

If the ideal is of dimension zero, we can represent the linear map  $\Lambda_i$  as a  $(\mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}})$  matrix.

**Definition 2.32** (Multiplication matrices). *The matrix representation of the linear map  $\Lambda_i$  is called the multiplication matrix by  $x_i$  and we denote it  $T_i$ . Thus, the  $j$ th column of the matrix  $T_i$  contains  $\Lambda_i(\epsilon_j)$  that is to say a vector representation of  $\text{NF}_>(x_i \epsilon_j)$ .*

**Example 2.33.** Let  $\mathcal{I} = \langle f_1, f_2 \rangle \subset \mathbb{F}_{53}[x_1, x_2]$  with

$$\begin{aligned} f_1 &= 5x_1^2 + 46x_1x_2 + 3x_2^2 + 30x_1 + 5x_2 + 27 \\ f_2 &= 2x_1^2 + 52x_1x_2 + 47x_2^2 + 9x_1 + 4x_2 + 5 \end{aligned}$$

The DRL Gröbner basis of  $\mathcal{I}$  with  $x_1 >_{\text{drl}} x_2$  is given by

$$\mathcal{G}_{>_{\text{drl}}} = \left\{ \begin{array}{l} x_2^3 + 20x_2^2 + 42x_1 + 43x_2 + 31 \\ x_1^2 + 48x_2^2 + 39x_1 + 32x_2 + 48 \\ x_1x_2 + 49x_2^2 + 16x_1 + 7x_2 + 38 \end{array} \right\}$$

thus  $\text{in}_{>_{\text{drl}}}(\mathcal{I}) = \langle x_1^2, x_1x_2, x_2^3 \rangle$ . Consequently, the canonical basis w.r.t. the DRL ordering of  $\mathbb{F}_{53}[x_1, x_2]/\mathcal{I}$  seen as a  $\mathbb{F}_{53}$ -vector space is given by  $B = \{x^\alpha \mid x^\alpha \notin \text{in}_{>_{\text{drl}}}(\mathcal{I})\} = \{x_2^2 >_{\text{drl}} x_1 >_{\text{drl}} x_2 >_{\text{drl}} 1\}$ . Moreover, the normal form w.r.t. the DRL ordering of  $x_1 \epsilon_i$  are given by

$$\begin{aligned} \text{NF}_{>_{\text{drl}}}(x_1 1) &= x_1 \\ \text{NF}_{>_{\text{drl}}}(x_1 x_2) &= -49x_2^2 - 16x_1 - 7x_2 - 38 = 4x_2^2 - 16x_1 - 7x_2 + 15 \\ \text{NF}_{>_{\text{drl}}}(x_1 x_1) &= -48x_2^2 - 39x_1 - 32x_2 - 48 = 5x_2^2 + 14x_1 + 21x_2 + 5 \\ \text{NF}_{>_{\text{drl}}}(x_1 x_2^2) &= 8x_2^2 + 35x_1 + 8x_2 + 7 \end{aligned}$$

By consequence the multiplication matrix  $T_1$  is given by

$$T_1 = \begin{pmatrix} 0 & 15 & 5 & 7 \\ 0 & -7 & 21 & 8 \\ 1 & -16 & 14 & 35 \\ 0 & 4 & 5 & 8 \end{pmatrix}.$$

Once the multiplication matrices by all the variables are known, computing the normal form of a polynomial  $f = \sum_{\alpha} c_{\alpha} x^{\alpha}$  can be done by using linear algebra computations, see Algorithm 1.

---

**Algorithm 1:** Computing normal forms by linear algebra.

---

**Input** : An ideal  $\mathcal{I}$ , its multiplication matrices  $T_1, \dots, T_n$  w.r.t. the monomial ordering  $>$  and a polynomial  $f = \sum_{\alpha \in \mathbb{N}^n} c_{\alpha} x^{\alpha} \in \mathbb{K}[x_1, \dots, x_n]$ .

**Output:** The normal form of  $f$  w.r.t.  $\mathcal{I}$  and the monomial ordering  $>$ .

- 1 Let  $\mathbf{1} = (1, 0, \dots, 0)^t = \Phi(1)^t$ ;
  - 2 Return  $\sum_{\alpha \in \mathbb{N}^n} c_{\alpha} T_1^{\alpha_1} \dots T_n^{\alpha_n} \mathbf{1}$ ;
- 

This vector representation of the quotient ring and operations in this vector space are the basic tools of change of ordering algorithms in Section 2.3 and Chapter 4.

### 2.1.3 Properties of degree reverse lexicographical Gröbner bases

In this section we present some properties of DRL Gröbner bases with  $x_1 >_{\text{drl}} \dots >_{\text{drl}} x_n$ , more precisely of  $\text{in}_{>_{\text{drl}}}(\mathcal{I})$ . These properties will be used in Chapter 4 to show that when using DRL ordering the multiplication matrix by the smallest variable *i.e.*  $T_n$  can be computed very efficiently. First, we investigate *generic* ideals.

**Definition 2.34** (Generic ideals). *A generic ideal is an ideal generated by a generic sequence of polynomials. A generic sequence of polynomials  $(f_1, \dots, f_s)$  is a sequence of polynomials whose coefficients are indeterminates *i.e.*  $f_i = \sum_{\alpha} c_{i,\alpha} x^{\alpha} \in \mathbb{K}[x_1, \dots, x_n]$  with  $\mathbb{K} = k(\{c_{i,\alpha}\})$  and  $k$  is a field.*

During his PdD, Moreno-Socías [MS91] precisely studied the shape of the stair of generic ideals for the particular case of DRL ordering.

**Definition 2.35** (Stair). *Given a monomial ordering  $>$ , the stair of an ideal  $\mathcal{I}$  is a minimal set of generators of  $\text{in}_{>}(\mathcal{I})$ . Note that if the reduced Gröbner basis w.r.t.  $>$ , denoted  $\mathcal{G}_{>}$ , of  $\mathcal{I}$  is known then the stair of  $\mathcal{I}$  is given by the set of leading terms of polynomials in  $\mathcal{G}_{>}$  and is denoted  $E_{>}(\mathcal{I})$ .*

A common tool of commutative algebra to study the stair of an ideal  $\mathcal{I}$  is the Hilbert series of the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ .

Indeed, for homogeneous (respectively affine) ideals, the coefficient of the terms of degree  $d$  in the Hilbert series counts the number of monomials of degree exactly  $d$  (respectively less than or equal to  $d$ ) that are not in  $\text{in}_{>}(\mathcal{I})$ . Note that this number and thus the Hilbert series does not depend on the monomial ordering. For more details about Hilbert series see Section 2.4.1. Using known results about Hilbert series associated to generic ideals and properties of the DRL ordering Moreno-Socías gives a complete description of the stair of generic ideals. More

precisely, using the compatibility with sections of the DRL ordering and regularity of generic algebra, he studies the Hilbert series of the sections w.r.t. the smallest variable of the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . Let denote  $\mathcal{R}$  the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . We call  $x_i^d \mathcal{R} / \langle x_i^{d+1} \rangle$  the section of  $\mathcal{R}$  by  $x_i^d$  that is to say the section of the basis of  $\mathcal{R}$  seen as a subspace of  $\mathbb{N}^n$  by the hyperplane  $x_i = d$ .

**Definition 2.36.** Let  $\mathcal{I}$  and  $\mathcal{J}$  be two ideals of  $\mathbb{K}[x_1, \dots, x_n]$ . The product of  $\mathcal{I}$  and  $\mathcal{J}$  is the ideal defined as

$$\mathcal{I}\mathcal{J} = \{fg \mid f \in \mathcal{I}, g \in \mathcal{J}\}.$$

Similarly, the ideal  $\mathcal{I}^k$  is defined by

$$\mathcal{I}^k = \left\{ \prod_{i=1}^k f_i \mid f_i \in \mathcal{I} \right\}.$$

**Lemma 2.37** (Compatibility with sections [MS91, MS03]). Let denote by  $o$  the valuation order of the  $(x_n)$ -adic filtration i.e.  $o(f) = \max \{i \mid f \in \langle x_n \rangle^i\}$  where  $f$  is a polynomial of  $\mathbb{K}[x_1, \dots, x_n]$ . That is to say  $o(f)$  is the maximal power of  $x_n$  that divides  $f$ . If  $\mathcal{G}_{>\text{drl}} = \{g_1, \dots, g_s\}$  is a Gröbner basis w.r.t. the DRL ordering of an ideal  $\mathcal{I}$  then  $\{g_j \mid o(g_j) < i\} \cup \{x_n^i\}$  is a Gröbner basis w.r.t. the DRL ordering of  $\mathcal{I} + \langle x_n \rangle^i$ .

For instance, in case of generic ideals this allow him for any  $d$  to count exactly the number of monomials in the canonical basis w.r.t. DRL ordering of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  that are of degree  $d$  in the smallest variable. More precisely, he shows that the intersection of the section of  $\mathcal{R}$  by  $x_{i_1}^{d_1}, \dots, x_{i_{n-2}}^{d_{n-2}}$  has steps of depth two and height one for any  $d_1, \dots, d_{n-2} \geq 0$  and  $i_1, \dots, i_{n-2} \leq n-1$  all pairwise distinct. We illustrate this result on Figure 2.1.

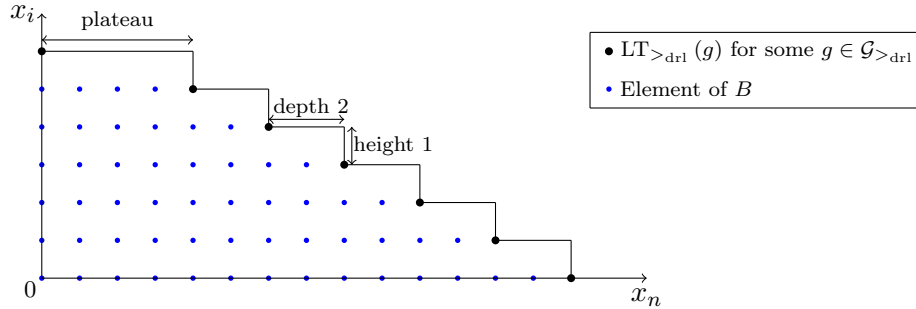


Figure 2.1: Intersection of sections of the quotient ring  $\mathcal{R} = \mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  by  $x_1^{d_1}, \dots, x_{i-1}^{d_{i-1}}, x_{i+1}^{d_{i+1}}, \dots, x_{n-1}^{d_{n-1}}$  with  $\mathcal{I}$  a generic ideal.

The shape of the stair in Figure 2.1 is formally stated in the following theorem.

**Theorem 2.38** (Moreno-Socías [MS91, MS03]). Let  $\mathcal{I} = \langle h_1, \dots, h_n \rangle \subset \mathbb{K}[x_1, \dots, x_n]$  be a generic ideal with  $\mathbb{K}$  a field of characteristic zero or  $n = 2$ . Let  $B$  be the canonical basis of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  and  $\tilde{B}_i = \{m = x_1^{\alpha_1} \dots x_{n-1}^{\alpha_{n-1}} \mid mx_n^i \in B\}$ . Let  $\delta = \sum_{i=1}^n (\deg(h_i) - 1)$ ,  $\delta^* = \sum_{i=1}^{n-1} (\deg(h_i) - 1)$  and  $\sigma = \min(\delta^*, \lfloor \frac{\delta}{2} \rfloor)$ . Let  $\mu = \delta - 2\sigma$ , then

a.  $\tilde{B}_0 = \dots = \tilde{B}_\mu$  (plateau) and  $\tilde{B}_i = \tilde{B}_{i+1}$  for  $\mu < i < \delta$  and  $i \not\equiv \delta \pmod{2}$  (depth two);

- b. The leading term of polynomials in  $\mathcal{G}_{>\text{drl}}$  of degree 0 in  $x_n$  have degree at most  $\sigma + 1 = \bar{\sigma}$ ;
- c. The leading term of polynomials in  $\mathcal{G}_{>\text{drl}}$  of degree  $\alpha$  in  $x_n$  with  $\mu < \alpha \leq \delta + 1$  with  $\alpha \not\equiv \delta \pmod{2}$  are all of total degree  $d + \alpha$  where  $d = \max(\deg(m) \mid m \in \tilde{B}_{\alpha-1})$ . Moreover, all these leading terms are exactly given by  $t = mx_n^\alpha$  for all  $m \in \tilde{B}_{\alpha-1}$  of degree  $d$  (height one);
- d. There is no leading term of polynomials in  $\mathcal{G}_{>\text{drl}}$  of degree  $1, \dots, \mu$  in  $x_n$  (plateau) or of degree  $\alpha$  in  $x_n$  with  $\alpha > \delta + 1$  or  $\mu \leq \alpha \leq \delta$  and  $\alpha \equiv \delta \pmod{2}$  (depth two).

This precise description of the stair of generic ideals w.r.t. the DRL ordering will allow us to show in Chapter 4 that  $T_n$  the multiplication matrix by  $x_n$  can be computed very efficiently. Moreover, Moreno-Socias extends his result by proposing the following conjecture.

**Definition 2.39** (Weakly reverse lexicographical ideal). Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be a monomial ideal i.e.  $\mathcal{I}$  is generated by monomials. Let  $E = \{m_1, \dots, m_s\}$  be a minimal basis of  $\mathcal{I}$ . We say that  $\mathcal{I}$  is a weakly reverse lexicographical ideal if for any  $t \in E$  and for any monomial  $m >_{\text{drl}} t$  such that  $\deg(m) = \deg(t)$  we have  $m \in \mathcal{I}$ .

**Conjecture 2.40** (Moreno-Socias [MS91, MS03]). Let  $\mathbb{K}$  be an infinite field and  $\mathcal{I}$  be a generic ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . The initial ideal of  $\mathcal{I}$  w.r.t. the DRL ordering is a weakly reverse lexicographical ideal.

To extend this result to non generic ideals and any fields we will use results about the generic initial ideal of Galligo [Gal73, Gal], Bayer and Stillman [BS87b] and Pardue [Par94]. These results are summarized in [Eis95, p. 351-358]. Indeed, they show that applying a generic linear change of variables allows to obtain a new ideal whose “generic” initial ideal has a known structure.

**Definition 2.41** (Linear change of variables). Let  $g \in \mathbf{GL}(\mathbb{K}, n)$  the ideal  $g \cdot \mathcal{I}$  is defined as follows  $g \cdot \mathcal{I} = \{f(g \cdot X) \mid f \in \mathcal{I}\}$  where  $X$  is the vector  $[x_1, \dots, x_n]$ .

**Theorem 2.42** ([Eis95] pages 351-358). Let  $\mathbb{K}$  be an infinite field and  $\mathcal{I}$  be a homogeneous ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . There exists a Zariski open set  $U \subset \mathbf{GL}(\mathbb{K}, n)$  and a monomial ideal  $\mathcal{J}$  such that  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \mathcal{J}$  for all  $g \in U$ .

**Definition 2.43** (Generic initial ideal). With  $\mathcal{I}$  and  $\mathcal{J}$  as in Theorem 2.42, the generic initial ideal of  $\mathcal{I}$  is denoted  $\mathbf{Gin}(\mathcal{I})$  and is defined by  $\mathcal{J}$ .

In order to state the proof of Theorem 2.42 we need to introduce some definitions and notations about multilinear algebra. For more details about multilinear algebra see [Eis95, Appendix 2].

The polynomial ring  $\mathbb{K}[x_1, \dots, x_n]$  is denoted  $R$ . It is a graded ring for the usual degree and we consider DRL ordering. Hence,  $R_d$  denotes the  $\mathbb{K}$ -vector space generated by monomials of  $R$  of degree  $d$ . If  $V \subset R_d$  is a  $t$  dimensional space of polynomials of degree  $d$  then multilinear algebra allows to represent it as a one dimensional subspace  $L = \wedge^t V \subset \wedge^t R_d$  where  $L$  is spanned by  $f = g_1 \wedge \dots \wedge g_t$  with  $g_1, \dots, g_t$  a basis of  $V$ .

A monomial of  $\wedge^t R_d$  is an element of the form  $m = m_1 \wedge \dots \wedge m_t$  with  $m_1, \dots, m_t$  are monomials in  $R_d$ . A term  $c \cdot m$  of  $\wedge^t R_d$  is the product of a monomial  $m$  in  $\wedge^t R_d$  and an element  $c$  of  $\mathbb{K}$ . An element  $f \in \wedge^t R_d$  is a finite linear combination of monomials in  $\wedge^t R_d$  or equivalently a finite sum of terms in  $\wedge^t R_d$ . If the  $m_i$ 's are not pairwise distinct then

$m = 0$ . Moreover, for any permutation  $\sigma$ ,  $m_1 \wedge \cdots \wedge m_t = \text{sign}(\sigma) \cdot m_{\sigma(1)} \wedge \cdots \wedge m_{\sigma(t)}$  where  $\text{sign}(\sigma)$  denotes the signature of  $\sigma$ . Thus, the normal expression of  $m$  is  $m_1 \wedge \cdots \wedge m_t$  with  $m_1 >_{\text{drl}} \cdots >_{\text{drl}} m_t$  and the set  $\wedge^t R_d$  is the  $\mathbb{K}$ -vector space of dimension  $\delta = \binom{r}{t}$  with basis

$$\{\kappa_{i_1} \wedge \cdots \wedge \kappa_{i_t} \mid 1 \leq i_1 < \cdots < i_t \leq r\} = \{\varepsilon_1, \dots, \varepsilon_\delta\}$$

where  $r = \dim_{\mathbb{K}}(R_d)$  and  $\{\kappa_1 >_{\text{drl}} \cdots >_{\text{drl}} \kappa_r\}$  is the basis of  $R_d$ . In the following, we always consider normal expressions of monomials in  $\wedge^t R_d$ .

The monomials of  $\wedge^t R_d$  are ordered lexicographically as described in the following definition.

**Definition 2.44.** Let  $m = m_1 \wedge \cdots \wedge m_t$  and  $v = v_1 \wedge \cdots \wedge v_t$  be two monomials of  $\wedge^t R_d$  then  $m \succ v$  if and only if there exists  $i \in \{1, \dots, t\}$  such that  $m_i >_{\text{drl}} v_i$  and  $m_j = v_j$  for any  $j < i$ . The leading term of  $f \in \wedge^t R_d$  is the greatest term in  $f$  w.r.t.  $\succ$ .

Let  $f_1, \dots, f_t$  be polynomials in  $R_d$  and let  $M$  be a matrix representation of these polynomials i.e.

$$M = (M_{i,j}) = \begin{array}{ccc|c} \kappa_1 & \cdots & \kappa_\delta & \\ \hline \star & \cdots & \star & f_1 \\ \vdots & \ddots & \vdots & \vdots \\ \star & \cdots & \star & f_t \end{array} \quad (2.2)$$

where  $M_{i,j}$  is the coefficient of  $\kappa_j$  in  $f_i$ . The element  $f$  of  $\wedge^t R_d$  associated to  $f_1, \dots, f_t$  is given by

$$f = f_1 \wedge \cdots \wedge f_t = \sum_{i=1}^{\delta} c_i \varepsilon_i$$

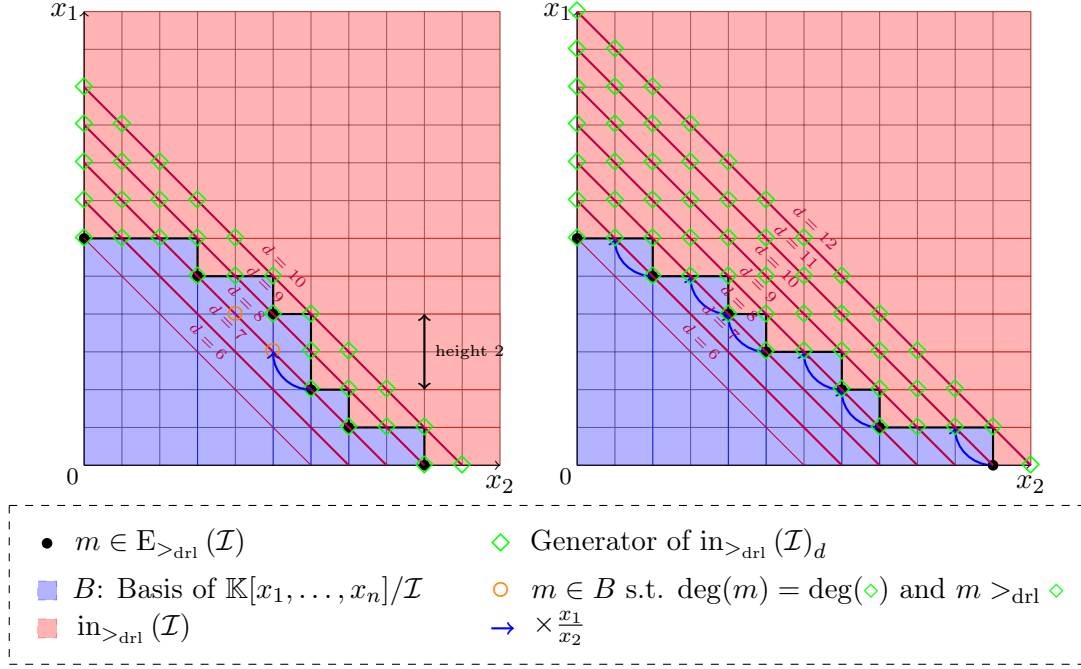
where  $c_i$  is the determinant of the  $t \times t$  sub-matrix of  $M$  constructed by keeping only the columns corresponding to the monomials defining  $\varepsilon_i$ .

Let  $\mathcal{I}$  be a homogeneous ideal of  $R$ . The degree- $d$  part of  $\mathcal{I}$  is defined by  $\mathcal{I}_d = \mathcal{I} \cap R_d$ . It is a  $\mathbb{K}$ -vector space of dimension  $t_d$ .

*Proof of Theorem 2.42.* Let  $f_1, \dots, f_{t_d}$  be a basis of  $\mathcal{I}_d$ . Let  $\mathbf{g} = (\mathbf{g}_{i,j})$  be a matrix of indeterminates of size  $n \times n$ . We have  $\mathbf{g} \cdot (f_1 \wedge \cdots \wedge f_{t_d}) = \mathbf{g} \cdot f_1 \wedge \cdots \wedge \mathbf{g} \cdot f_{t_d}$  is a linear combination of monomials in  $\wedge^{t_d} R_d$  whose coefficients are polynomials in the  $\mathbf{g}_{i,j}$ 's. Let  $p_d(\mathbf{g}_{1,1}, \dots, \mathbf{g}_{n,n}) \cdot m$  be the leading term of  $\mathbf{g} \cdot f_1 \wedge \cdots \wedge \mathbf{g} \cdot f_{t_d}$  with  $m = m_1 \wedge \cdots \wedge m_{t_d}$ . We define  $U_d$  the subset of  $\mathbf{GL}(\mathbb{K}, n)$  as  $U_d = \{g = (g_{i,j}) \mid p_d(g_{1,1}, \dots, g_{n,n}) \neq 0\}$ . Hence, the degree- $d$  part of  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})$  is generated by  $(m_1, \dots, m_{t_d})$  if and only if  $g \in U_d$ . We define  $J_d$  as the subset of  $R_d$  spanned by  $m_1, \dots, m_{t_d}$ .

We now show that  $J = \bigoplus_{d \in \mathbb{N}} J_d$  is an ideal. To this aim we show that for all  $d \in \mathbb{N}$  one has  $R_1 J_d \subset J_{d+1}$ . Since  $U_d$  and  $U_{d+1}$  are open and dense then  $U_d \cap U_{d+1} \neq \emptyset$ . Thus, there exists  $g \in U_d \cap U_{d+1} \subset \mathbf{GL}(\mathbb{K}, n)$  such that  $J_d$  (resp.  $J_{d+1}$ ) is the degree- $d$  (resp. degree- $(d+1)$ ) part of  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})$ . By consequence, for any  $d \in \mathbb{N}$ ,  $R_1 J_d \subset J_{d+1}$  and  $J$  is an ideal.

It remains to prove that  $U = \bigcap_{d \in \mathbb{N}} U_d$  is a Zariski dense open subset of  $\mathbf{GL}(\mathbb{K}, n)$ . To this aim, it suffices to show that  $U$  is actually equal to a finite intersections of the  $U_d$ . Assume that  $J$  is generated by monomials of degree less than or equal to  $e$ . Let  $g \in \bigcap_{d=0}^e U_d$  for  $d = 0, \dots, e$  we have  $J_d$  is the degree- $d$  part of  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})$  denoted  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})_d$ . Since  $J$  is generated by monomials of degree less than or equal to  $e$  we have  $\bigoplus_{d=0}^e R J_d = J$  and since  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})$  is an ideal we have  $J = \bigoplus_{d=0}^e R J_d = \bigoplus_{d=0}^e R \text{in}_{>\text{drl}}(g \cdot \mathcal{I})_d \subset \text{in}_{>\text{drl}}(g \cdot \mathcal{I})$ .

Figure 2.2: Steps of height one and generators of  $\text{in}_{>\text{drl}}(\mathcal{I})_d$ .

Moreover, for any  $d$  we have  $\dim_{\mathbb{K}}(J_d) = \dim_{\mathbb{K}}(\mathcal{I}_d) = \dim_{\mathbb{K}}(\text{in}_{>\text{drl}}(g \cdot \mathcal{I})_d)$ . By consequence,  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = J$ .  $\square$

The structure of generic initial ideals was first studied by Galligo in [Gal, Gal73] in the case of fields of characteristic zero. In particular, he shows that if  $\mathbb{K}$  is a field of characteristic zero, then the generic initial ideal is Borel fixed.

**Definition 2.45.** *The Borel subgroup of  $\mathbf{GL}(\mathbb{K}, n)$  is the set of invertible upper triangular matrices and is denoted  $\mathcal{B}$ .*

Later, Bayer and Stillman extend this result in [BS87b] to infinite field of any characteristic. These results are summarized in the following theorem.

**Theorem 2.46** ([Gal73, Gal, BS87b]). *If  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  is a homogeneous ideal and  $\mathbb{K}$  an infinite field, then  $\mathbf{Gin}(\mathcal{I})$  is Borel fixed. That is to say for all  $g \in \mathcal{B}$ ,  $g \cdot \mathbf{Gin}(\mathcal{I}) = \mathbf{Gin}(\mathcal{I})$ .*

In [Gal73, Gal] it is also shown that generic initial ideals satisfy the following property.

**Property 2.47.** *Let  $\mathcal{I}$  be a homogeneous ideal of  $\mathbb{K}[x_1, \dots, x_n]$  with  $\mathbb{K}$  a field of characteristic zero. Let  $m \in \mathbf{Gin}(\mathcal{I})$  then for all  $n \geq j > i \geq 1$  such that  $x_j$  divides  $m$  we have  $\frac{x_i}{x_j}m \in \mathbf{Gin}(\mathcal{I})$ .*

The idea in [Gal73, Gal] was to follow step by step the computation of the DRL Gröbner basis of  $g \cdot \mathcal{I}$  by assuming that  $g$  is generic enough to ensure that the leading terms of the polynomials in the DRL Gröbner basis (i.e. the stair of  $g \cdot \mathcal{I}$ ) are the greatest as possible. It is important to note that, for fields of characteristic zero, it is exactly how the generic initial ideal is constructed in proof of Theorem 2.42.

Indeed, since the field is of characteristic zero, for each integer  $d$  the matrix  $M$  as in equation (2.2) associated to  $\mathbf{g} \cdot f_1, \dots, \mathbf{g} \cdot f_{t_d}$  has only non zero entries. Hence, the Zariski open subset  $U \subset \mathbf{GL}(\mathbb{K}, n)$  such that for any  $g \in U$ ,  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \mathbf{Gin}(\mathcal{I})$  is constructed to ensure that the degree- $d$  part of  $\mathbf{Gin}(\mathcal{I})$  is generated by the  $t_d$  largest possible (w.r.t. DRL ordering) monomials of degree  $d$ . This implies that all the intersections of sections of  $\mathcal{R}_G = \mathbb{K}[x_1, \dots, x_n] / \mathbf{Gin}(\mathcal{I})$  by  $x_{i_1}^{d_1}, \dots, x_{i_{n-2}}^{d_{n-2}}$  have steps of height one (with  $x_j$  on the  $x$ -axis); where  $\{i_1, \dots, i_{n-2}\} = \{1, \dots, n\} \setminus \{i, j\}$ ,  $d_1, \dots, d_{n-2} \geq 0$  and  $1 \leq j < i \leq n$ . This is exactly what means Property 2.47. In Figure 2.2 in the case of two variables, we describe the link between steps of height one and generators of degree- $d$  part of monomials ideals. Keeping in mind that all the monomials of same degree are on a same diagonal (red line in Figure 2.2). Moreover, for the DRL ordering, if  $m_1$  and  $m_2$  are two monomials of same degree then  $m_1 >_{\text{drl}} m_2$  if and only if  $m_1$  is closer than  $m_2$  to the axis corresponding to the greatest variable (here  $x_1$ ).

If the field  $\mathbb{K}$  is of positive characteristic then some entries of the matrix  $M$  as in equation (2.2) associated to  $\mathbf{g} \cdot f_1, \dots, \mathbf{g} \cdot f_{t_d}$  can be zero. Thus, some expected (for the characteristic zero case) non-zero minors of size  $t_d \times t_d$  may become identically null. In that case, the generic initial ideal might not satisfy Property 2.47.

**Example 2.48.** Let us consider the ideal  $\mathcal{I} = \langle x_1^2, x_2^2 \rangle \subset \mathbb{K}[x_1, x_2]$  and the matrix

$$\mathbf{g} = \begin{pmatrix} \mathbf{g}_{1,1} & \mathbf{g}_{1,2} \\ \mathbf{g}_{2,1} & \mathbf{g}_{2,2} \end{pmatrix}.$$

Whatever the field  $\mathbb{K}$ , the degree-2 part of  $\mathcal{I}$  is of dimension 2 and is generated by  $f_1 = x_1^2$  and  $f_2 = x_2^2$ . To construct the generic initial ideal of  $\mathcal{I}$ , one looks for the leading term of  $\mathbf{g} \cdot f_1 \wedge \mathbf{g} \cdot f_2$ . Hence, we study the minors of size  $2 \times 2$  of the matrix representation of  $\mathbf{g} \cdot f_1$  and  $\mathbf{g} \cdot f_2$ . Let  $M$  be such a matrix and  $p$  be the characteristic of  $\mathbb{K}$  one has

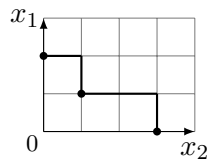
$$M = \begin{pmatrix} \boxed{\mathbf{g}_{1,1}^2} & \boxed{2\mathbf{g}_{1,1}\mathbf{g}_{1,2}} & \boxed{\mathbf{g}_{1,2}^2} \\ \boxed{\mathbf{g}_{2,1}^2} & \boxed{2\mathbf{g}_{2,1}\mathbf{g}_{2,2}} & \boxed{\mathbf{g}_{2,2}^2} \end{pmatrix} \begin{matrix} \mathbf{g} \cdot f_1 \\ \mathbf{g} \cdot f_2 \end{matrix}$$

if  $p = 0$  or  $p > 2$ .

Hence, the leading term of  $\mathbf{g} \cdot f_1 \wedge \mathbf{g} \cdot f_2$  is  $2\mathbf{g}_{1,1}\mathbf{g}_{2,1}(\mathbf{g}_{1,1}\mathbf{g}_{2,2} - \mathbf{g}_{2,1}\mathbf{g}_{1,2}) \cdot x_1^2 \wedge x_1x_2$ .

The degree-2 part of the generic initial ideal of  $\mathcal{I}$  is then generated by  $x_1^2$  and  $x_1x_2$  the two greatest monomials w.r.t. the DRL ordering of degree 2.

The generic initial ideal of  $\mathcal{I}$  **satisfies** Property 2.47, see its stair below.



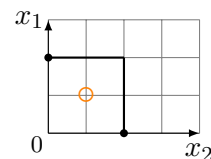
$$M = \begin{pmatrix} \boxed{\mathbf{g}_{1,1}^2} & 0 & \boxed{\mathbf{g}_{1,2}^2} \\ \boxed{\mathbf{g}_{2,1}^2} & 0 & \boxed{\mathbf{g}_{2,2}^2} \end{pmatrix} \begin{matrix} \mathbf{g} \cdot f_1 \\ \mathbf{g} \cdot f_2 \end{matrix}$$

if  $p = 2$ .

Hence, the leading term of  $\mathbf{g} \cdot f_1 \wedge \mathbf{g} \cdot f_2$  is  $(\mathbf{g}_{1,1}\mathbf{g}_{2,2} + \mathbf{g}_{2,1}\mathbf{g}_{1,2}) \cdot x_1^2 \wedge x_2^2$ .

The degree-2 part of the generic initial ideal of  $\mathcal{I}$  is then generated by  $x_1^2$  and  $x_2^2$ .

The generic initial ideal of  $\mathcal{I}$  **does not satisfy** Property 2.47, see its stair below.



The study of the structure of generic initial ideals in any characteristic has been done by Pardue in [Par94]. This is summarized in Theorem 2.49. In order to state his result, we need to introduce a partial order on the integer. The partial order  $\prec_p$  on the natural numbers is defined as follows: for any  $a, b \in \mathbb{N}$ ,  $a \prec_p b$  if  $\binom{b}{a} \not\equiv 0 \pmod{p}$ . When  $p = 0$  then  $\prec_0$  is the usual total order  $\leq$ .

**Theorem 2.49** ([Par94]). *Let  $\mathcal{I}$  be a monomial ideal of  $\mathbb{K}[x_1, \dots, x_n]$  with  $\mathbb{K}$  an infinite field of characteristic  $p \geq 0$ . The monomial ideal  $\mathcal{I}$  is Borel fixed if and only if for all generators  $m$  of  $\mathcal{I}$  and for all  $1 \leq i < j \leq n$  such that  $x_j^t$  is the largest power of  $x_j$  dividing  $m$  then  $\left(\frac{x_i}{x_j}\right)^s \cdot m \in \mathcal{I}$  for all  $s \prec_p t$ .*

Since  $1 \prec_p t$  if  $t \not\equiv 0 \pmod{p}$ , from Theorem 2.46 and 2.49 we get the following corollary.

**Corollary 2.50.** *Let  $\mathcal{I}$  be a homogeneous ideal of  $\mathbb{K}[x_1, \dots, x_n]$  with  $\mathbb{K}$  an infinite field of characteristic  $p \geq 0$ . Let  $m \in \mathbf{Gin}(\mathcal{I})$  then for all  $n \geq j > i \geq 1$  such that  $x_j$  divides  $m$  we have  $\frac{x_i}{x_j}m \in \mathbf{Gin}(\mathcal{I})$  if  $p = 0$  or  $p > 0$  and  $t \not\equiv 0 \pmod{p}$  with  $x_j^t$  is the maximal power of  $x_j$  dividing  $m$ .*

In Chapter 4, we will show that this particular structure of *Generic initial ideals* allows to significantly speed up one step in the polynomial systems solving process using Gröbner bases.

#### 2.1.4 Properties of lexicographical Gröbner bases

Among the many properties of Gröbner bases, one of the most useful property for polynomial systems solving is the particular shape of lexicographical Gröbner bases induces by *The Elimination Theorem*.

**Definition 2.51** (Elimination order). *A monomial ordering on  $\mathbb{K}[x_1, \dots, x_n]$ , is called an elimination order w.r.t. the variables  $\{x_k, \dots, x_n\}$  and denoted  $>_k$ , if for all  $f \in \mathbb{K}[x_1, \dots, x_n]$ ,  $\text{LT}_{>_k}(f) \in \mathbb{K}[x_k, \dots, x_n]$  implies that  $f \in \mathbb{K}[x_k, \dots, x_n]$ .*

**Example 2.52.** • *The lexicographical ordering is an elimination order w.r.t. any set of variables  $\{x_k, \dots, x_n\}$  with  $k = 1, \dots, n$ .*

- *A block ordering,  $>_{\text{drl}, \text{drl}}$  w.r.t. the two sets of variables  $\{x_1, \dots, x_{k-1}\}$  and  $\{x_k, \dots, x_n\}$  defined as follows:  $x^\alpha >_{\text{drl}, \text{drl}} x^\beta$  if  $x^{(\alpha_1, \dots, \alpha_{k-1}, 0, \dots, 0)} >_{\text{drl}} x^{(\beta_1, \dots, \beta_{k-1}, 0, \dots, 0)}$  or  $\alpha_i = \beta_i$  for  $i = 1, \dots, k-1$  and  $x^{(0, \dots, 0, \alpha_k, \dots, \alpha_n)} >_{\text{drl}} x^{(0, \dots, 0, \beta_k, \dots, \beta_n)}$ ; is an elimination order w.r.t. the variables  $\{x_k, \dots, x_n\}$ .*

**Theorem 2.53** (The Elimination Theorem [CLO07] page 116). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal and let  $>_k$  be an elimination order w.r.t. the set of variables  $\{x_k, \dots, x_n\}$ . Let  $\mathcal{G}_{>_k}$  be the Gröbner basis of  $\mathcal{I}$  w.r.t.  $>_k$ . Then,  $\mathcal{G}_{>_k}^{(k)} = \mathcal{G}_{>_k} \cap \mathbb{K}[x_k, \dots, x_n]$  is a Gröbner basis of the  $k$ -th elimination ideal  $\mathcal{I}^{(k)} = \mathcal{I} \cap \mathbb{K}[x_k, \dots, x_n]$ .*

In [CLO07], *The Elimination Theorem* is stated for the particular case of the LEX ordering. However, the proof works *mutatis mutandis* with any elimination order.

A first consequence of *The Elimination Theorem* is that computing a Gröbner basis w.r.t. an elimination order allows to perform a polynomial change of variables. Indeed, let  $f \in$

$\mathbb{K}[x_1, \dots, x_n]$ . Assume that there exist  $h \in \mathbb{K}[y_1, \dots, y_n]$  and  $g_1, \dots, g_n \in \mathbb{K}[x_1, \dots, x_n]$  such that  $f = h(g_1, \dots, g_n)$ . Let  $>_{n+1}$  be an elimination order w.r.t.  $\{y_1, \dots, y_n\}$  of the polynomial ring  $\mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_n]$ . Let  $\mathcal{G}_{>_{n+1}}$  be the Gröbner basis w.r.t.  $>_{n+1}$  of the ideal

$$\mathcal{I} = \langle f, y_1 - g_1, \dots, y_n - g_n \rangle \subset \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_n].$$

Since,  $\{h\}$  is a basis of  $\mathcal{I}^{(n+1)}$  we have  $\{h\} = \mathcal{G}_{>_{n+1}} \cap \mathbb{K}[y_1, \dots, y_n]$ . This procedure is summarized in Algorithm 2.

---

**Algorithm 2:** Applying a polynomial change of variables (1).

---

**Input** :  $g_1, \dots, g_n \in \mathbb{K}[x_1, \dots, x_n]$  and  $f \in \mathbb{K}[x_1, \dots, x_n]$  such that there exists  $h \in \mathbb{K}[y_1, \dots, y_n]$  satisfying  $f = h(g_1, \dots, g_n)$ .  
**Output:**  $h \in \mathbb{K}[y_1, \dots, y_n]$  such that  $f = h(g_1, \dots, g_n)$ .  
1  $\mathcal{I} := \langle f, y_1 - g_1, \dots, y_n - g_n \rangle \subset \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_n]$ ;  
2 Compute  $\mathcal{G}_{>_{n+1}}$  be the Gröbner basis of  $\mathcal{I}$  w.r.t. an elimination order  $>_{n+1}$  w.r.t.  $\{y_1, \dots, y_n\}$ ;  
3  $\{h\} := \mathcal{G}_{>_{n+1}} \cap \mathbb{K}[y_1, \dots, y_n]$ ;  
4 **return**  $h$ ;

---

Another similar way to perform polynomial change of variables is to compute  $\mathcal{G}'_{>_{n+1}}$  the Gröbner basis w.r.t.  $>_{n+1}$  of the ideal  $\langle y_1 - g_1, \dots, y_n - g_n \rangle \subset \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_n]$ . Then,  $h$  is computed as the remainder of  $f$  w.r.t.  $\mathcal{G}'_{>_{n+1}}$  i.e.  $h = \text{NF}_{>_{n+1}}(f)$ . For more details see [CLO07, page 341]. This strategy is summarized in Algorithm 3.

---

**Algorithm 3:** Applying a polynomial change of variables (2).

---

**Input** :  $g_1, \dots, g_n \in \mathbb{K}[x_1, \dots, x_n]$  and  $f \in \mathbb{K}[x_1, \dots, x_n]$ .  
**Output:**  $h \in \mathbb{K}[y_1, \dots, y_n]$  – if it exists – such that  $f = h(g_1, \dots, g_n)$  or *fail* otherwise.  
1  $\mathcal{I} := \langle y_1 - g_1, \dots, y_n - g_n \rangle \subset \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_n]$ ;  
2 Compute  $\mathcal{G}_{>_{n+1}}$  be the Gröbner basis of  $\mathcal{I}$  w.r.t. an elimination order  $>_{n+1}$  w.r.t.  $\{y_1, \dots, y_n\}$ ;  
3  $h :=$  normal form of  $f$  w.r.t.  $\mathcal{I}$  and the monomial ordering  $>_{n+1}$ ;  
4 **if**  $h \in \mathbb{K}[y_1, \dots, y_n]$  **then** Return  $h$ ;  
5 **else** Return *fail*;

---

The lexicographical ordering is quite specific among the elimination orders. Indeed, the property to be an elimination order for any set of variables implies that the Gröbner basis w.r.t. this ordering is very particular.

**Proposition 2.54.** *Let  $\mathcal{I}$  be an ideal of dimension zero i.e.  $|\mathbf{V}_{\mathbb{K}}(\mathcal{I})| < \infty$ . The lexicographical Gröbner basis of  $\mathcal{I}$  has the following shape:*

$$\mathcal{G}_{lex} = \left\{ \begin{array}{c} g_{1,1}(x_1, \dots, x_n), \dots, g_{1,s_1}(x_1, \dots, x_n) \\ \vdots \\ g_{n-1,1}(x_{n-1}, x_n), \dots, g_{n-1,s_{n-1}}(x_{n-1}, x_n) \\ g_n(x_n) \end{array} \right\} \quad (2.3)$$

with  $s_i > 0$  for  $i = 1, \dots, n-1$ .

*Proof.* This result is obtained by applying The Elimination Theorem. Indeed,  $\mathcal{G}_{\text{lex}}^{(n)} = \mathcal{G}_{\text{lex}} \cap \mathbb{K}[x_n]$  is the lexicographical Gröbner basis of the  $n$ -th elimination ideal  $\mathcal{I}^{(n)} = \mathcal{I} \cap \mathbb{K}[x_n]$ . Since  $\mathcal{I}^{(n)}$  is a principal ideal it is generated by a unique polynomial  $g_n(x_n)$ . Moreover, since  $\mathcal{I}$  is zero-dimensional  $g_n \neq 0$  and  $\mathcal{G}_{\text{lex}}^{(n)} = \{g_n\}$ . Now, for  $i = k, \dots, n-1$  assume that

$$\mathcal{G}_{\text{lex}}^{(i)} = \{g_{i,1}, \dots, g_{i,s_i}, \dots, g_{n-1,1}, \dots, g_{n-1,s_{n-1}}, g_n\}$$

with  $s_j > 0$  and  $g_{j,1}, \dots, g_{j,s_j}$  in  $\mathbb{K}[x_j, \dots, x_n]$  but not in  $\mathbb{K}[x_{j+1}, \dots, x_n]$  for  $j = i, \dots, n-1$ . We have  $\mathcal{G}_{\text{lex}}^{(k-1)} = \mathcal{G}_{\text{lex}} \cap \mathbb{K}[x_{k-1}, \dots, x_n] = \mathcal{G}_{\text{lex}}^{(k)} \cup \{g_{k-1,1}, \dots, g_{k-1,s_{k-1}}\}$  with  $g_{k-1,j} \in \mathbb{K}[x_{k-1}, \dots, x_n]$  for  $j = 1, \dots, s_{k-1}$ . Moreover, since  $\mathcal{I}$  is zero-dimensional we have  $s_{k-1} > 0$ .  $\square$

Since from Corollary 2.23 the LEX Gröbner basis,  $\mathcal{G}_{>\text{lex}} = \{g_1, \dots, g_r\}$ , of an ideal  $\mathcal{I} = \langle f_1, \dots, f_s \rangle$  is a basis of  $\mathcal{I}$ , Proposition 2.10 implies that the solutions of the polynomial system  $\{f_1 = 0, \dots, f_s = 0\}$  are exactly the same as the solutions of the polynomial system  $\{g_1 = 0, \dots, g_r = 0\}$ . Hence, given the LEX Gröbner basis of  $\langle f_1, \dots, f_s \rangle$ , solving the system  $\{f_1 = 0, \dots, f_s = 0\}$  can be done by solving the system  $\{g_1 = 0, \dots, g_r = 0\}$ . This can be done by solving some sequence of univariate polynomials. However, since  $s_1, \dots, s_{n-1}$  in equation (2.3) can be greater than one, the choice of the sequence of univariate polynomials to solve may be not unique. Some lexicographical Gröbner bases can have a more particular structure avoiding this ambiguity.

**Definition 2.55** (Triangular set). *Consider polynomial systems in  $\mathbb{K}[x_1, \dots, x_n]$  with  $x_1 > \dots > x_n$ . The main variable of a polynomial  $f$  in  $\mathbb{K}[x_1, \dots, x_n]$  is the greatest variable appearing in  $f$ . A set  $S$  of  $n$  polynomials in  $\mathbb{K}[x_1, \dots, x_n]$  is a triangular set if for  $i \in \{1, \dots, n\}$  the main variable of the  $i$ th polynomial is  $x_i$  and if this polynomial seen as a polynomial in  $x_i$  is monic. That is to say  $S$  has the following shape*

$$\left\{ \begin{array}{c} x_1^{d_1} + h_1(x_1, \dots, x_n) \\ \vdots \\ x_{n-1}^{d_{n-1}} + h_{n-1}(x_{n-1}, x_n) \\ x_n^{d_n} + h_n(x_n) \end{array} \right\}$$

where  $\deg_{x_i}(h_i) < d_i$  for  $i = 1, \dots, n$  where  $\deg_{x_i}(f)$  denotes the degree of  $f$  seen as a univariate polynomial in  $x_i$ .

When the lexicographical Gröbner basis of an ideal  $\mathcal{I}$  is also a triangular set then there is a unique sequence of length at most  $(n-1)\mathcal{D}_{\mathcal{I}} + 1$  of univariate polynomials to solve to find the solutions of the system. We recall that  $\mathcal{D}_{\mathcal{I}}$  denotes the degree of  $\mathcal{I}$ . Otherwise, when the lexicographical Gröbner basis is not a triangular set then one can use LexTriangular algorithm of Lazard [Laz92] which given a LEX Gröbner basis computes a set of triangular sets of which the union of the solutions are the solutions of the input system.

Among the LEX Gröbner basis, some of them are particular triangular sets from which finding the solutions of the system can be done by solving a unique univariate polynomial. This particular shape of LEX Gröbner bases is called *Shape Position* and is defined below.

**Definition 2.56** (Shape Position). *A zero-dimensional ideal  $\mathcal{I}$  of  $\mathbb{K}[x_1, \dots, x_n]$  is said to be in Shape Position if its LEX Gröbner basis is of the form*

$$\left\{ \begin{array}{c} x_1 - h_1(x_n) \\ \vdots \\ x_{n-1} - h_{n-1}(x_n) \\ h_n(x_n) \end{array} \right\}$$

with  $\deg(h_n) = \mathcal{D}_{\mathcal{I}}$  and  $\deg(h_i) < \mathcal{D}_{\mathcal{I}}$  for  $i = 1, \dots, n-1$  where  $\mathcal{D}_{\mathcal{I}}$  is the number of solutions of  $\mathcal{I}$  counted with multiplicities in  $\overline{\mathbb{K}}$ , the algebraic closure of  $\mathbb{K}$  i.e. the degree of  $\mathcal{I}$ .

From such a LEX Gröbner basis, one can notice that solving a polynomial system can be done by solving one univariate polynomial and evaluating  $n - 1$  univariate polynomials in  $\mathcal{D}_{\mathcal{I}}$  points. By consequence, from such a LEX Gröbner basis recovering the solutions of the system does not require the LexTriangular algorithm and can be done very efficiently. Even if this shape of LEX Gröbner bases is very particular it is not less common. Indeed, in most applications the *Shape Position* is the expected shape of LEX Gröbner bases. Moreover, for radical ideals the *Shape Position* property is a generic one.

**Lemma 2.57** (Shape Lemma [GM89, Lak90]). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be a radical ideal of dimension zero where  $\mathbb{K}$  is a field of characteristic zero. There exists a Zariski open subset  $U \subset \mathbf{GL}(\mathbb{K}, n)$  such that for all  $g \in U$  the ideal  $g \cdot \mathcal{I}$  is in Shape Position.*

*Proof.* Here we only detail the construction of  $U$ . For a complete proof of the Shape Lemma see [GM89, Lak90]. Since  $\mathcal{I}$  is radical, all its solutions are of multiplicity one. Thus, the set of solutions of  $\mathcal{I}$ :  $\{a_i = (a_{i,1}, \dots, a_{i,n}) \in \overline{\mathbb{K}}^n \mid f_j(a_1, \dots, a_n) = 0, j = 1, \dots, n\}$  is of cardinality  $\mathcal{D}_{\mathcal{I}}$ . Let  $g$  be a given matrix in  $\mathbf{GL}(\mathbb{K}, n)$ . We denote by  $v_i = (v_{i,1}, \dots, v_{i,n})$  the point obtained after transformation of  $a_i$  by  $g$ , i.e  $v_i = g \cdot a_i^t$ . To ensure that  $g \cdot \mathcal{I}$  admits a LEX Gröbner basis in *Shape Position*,  $g$  should be such that  $v_{i,n} \neq v_{j,n}$  for all couples of integers  $(i, j)$  verifying  $1 \leq j < i \leq \mathcal{D}_{\mathcal{I}}$ . Hence, let  $\mathbf{g} = (\mathbf{g}_{i,j})$  be a  $(n \times n)$  matrix of unknowns, the polynomial  $P_U$  defining the Zariski open subset  $U$  is then given as the determinant of the Vandermonde matrix associated to  $\mathbf{v}_{i,n}$  for  $i = 1, \dots, \mathcal{D}_{\mathcal{I}}$  where  $\mathbf{v}_i = (\mathbf{v}_{i,1}, \dots, \mathbf{v}_{i,n}) = \mathbf{g} \cdot a_i^t$ . Therefore, we know exactly the degree of  $P_U$  which is  $\frac{\mathcal{D}_{\mathcal{I}}(\mathcal{D}_{\mathcal{I}}-1)}{2}$ .  $\square$

Note that a zero-dimensional ideal which is not radical can have a LEX Gröbner basis in *Shape Position* and also up to a linear change of coordinates, almost all zero-dimensional ideals have a LEX Gröbner basis in *Shape Position*.

**Remark 2.58.** *Any ideal can be represented thanks to a finite basis i.e. a finite sequence of polynomials. Moreover, the set of sequences of  $s$  polynomials of degree  $d_1, \dots, d_s$  can be viewed as an affine space whose coordinates are given by the coefficients of the polynomials. Consequently, any ideals can be seen as an element of an affine space. Hence, in case of fields of characteristic zero we mean by almost all that there exists a Zariski open subset of ideals satisfying this property.*

The characterization of zero-dimensional ideals having a LEX Gröbner basis in *Shape Position* after a linear change of coordinates have been done by Becker *et al* in [BMMT94]. In order to give this characterization we need to introduce some technical definitions not used elsewhere in this thesis. By consequence, we refer the interested reader to [BMMT94].

### 2.1.5 What means solving?

Depending on the context, solving a polynomial system has different meanings. We saw that from a LEX Gröbner basis solving a polynomial system is reduced to solve univariate polynomials. Hence, the LEX Gröbner basis gives a symbolic representation of the solutions of a polynomial system from which it is easy to deduce the solutions of the system. Indeed, extracting the solutions (or an approximation of them) of an univariate polynomial can be efficiently done. The algorithms to compute such roots have their complexities well handled and in general they are negligible in comparison to the cost of computing a LEX Gröbner basis. For more details about solving univariate polynomials in finite fields see [VZGG03]. In the characteristic zero case, see [Pan02] to find an approximation of all the real roots.

Consequently, in the whole of this thesis, solving a polynomial system means computing the lexicographical Gröbner basis of the ideal that the system generates. In the 1960s Buchberger introduced, in his PhD thesis [Buc06, Buc65], the concept of Gröbner bases and the first algorithm to compute them. Then, in the 1980s the link between linear algebra and Gröbner bases is highlighted by Lazard in [Laz83] where he proposed the first algorithm using linear algebra to compute Gröbner bases. Following the work of Lazard, new efficient algorithms to compute Gröbner bases based on linear algebra have been proposed around the 2000s by Faugère in [Fau99, Fau02]. The next section is devoted to present the outline of the algorithms based on linear algebra.

## 2.2 Gröbner bases algorithms

We first present the algorithm introduced by Lazard in [Laz83] reducing Gröbner bases to linear algebra. Then, we briefly introduced the improvements by Faugère. Throughout, this section we equip the ring  $\mathbb{K}[x_1, \dots, x_n]$  with the grading  $\Gamma$ .

### 2.2.1 Lazard's algorithm

Let  $\langle f_1, \dots, f_s \rangle = \mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be homogeneous polynomials with  $d_i = \Gamma(f_i)$  for  $i = 1, \dots, s$ . The idea of Lazard is to note that the polynomials  $mf_i$  for all monomials  $m$  of graduation  $d - d_i$  generates the  $\mathbb{K}$ -vector space  $\mathcal{I}_d = \mathcal{I} \cap \mathbb{K}[x_1, \dots, x_n]_d$ . Then, from a matrix representation of these polynomials one can compute a linear basis of  $\mathcal{I}_d$  by computing the reduced row echelon form of the matrix. Finally, from the linear bases of  $\mathcal{I}_j$  for  $j = 0, \dots, d$  one can construct a  $d$ -Gröbner basis of  $\mathcal{I}$  as defined below.

**Definition 2.59** ( $d$ -Gröbner bases). *A subset  $\{f_1, \dots, f_s\}$  of an ideal  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  is a  $d$ -Gröbner basis of  $\mathcal{I}$  w.r.t. the monomial ordering  $>$  if for any  $f \in \mathcal{I}$  with  $\Gamma(f) \leq d$  we have*

$$\text{LT}_>(f) \in \langle \text{LT}_>(f_1), \dots, \text{LT}_>(f_s) \rangle.$$

**Proposition 2.60.** *Let  $\mathcal{I}$  be a homogeneous ideal and  $>$  a monomial ordering. There exists an integer  $d_0$  such that for any  $d \geq d_0$  every  $d$ -Gröbner basis of  $\mathcal{I}$  w.r.t.  $>$  is actually a Gröbner basis of  $\mathcal{I}$  w.r.t.  $>$ .*

Hence, the algorithm of Lazard computing  $d$ -Gröbner bases allows to compute also Gröbner bases. In order to present this algorithm we introduce the notion of *Macaulay matrix* which is a matrix representation of the polynomials  $mf_i$  aforementioned.

**Definition 2.61** (Macaulay matrix). Let  $F = (f_1, \dots, f_s)$  be a sequence of homogeneous polynomials of  $\mathbb{K}[x_1, \dots, x_n]$  with  $d_i = \Gamma(f_i)$ . Let  $>$  be a monomial ordering on  $\mathbb{K}[x_1, \dots, x_n]$ . The Macaulay matrix in graduation  $d$  associated to  $F$ , denoted  $\text{Mac}_{>,d}(F)$ , is the matrix with columns indexed with monomials in  $\mathbb{K}[x_1, \dots, x_n]$  of graduation  $d$  and arranged in decreasing order w.r.t.  $>$ . A signature  $(m)$  is attached to each column with  $m$  being the corresponding monomial of graduation  $d$ . The rows of  $\text{Mac}_{>,d}(F)$  contains all the polynomials  $m f_i$  for  $i = 1, \dots, s$  and for all monomials  $m$  of  $\mathbb{K}[x_1, \dots, x_n]$  of graduation  $d - d_i$ . The signature  $(m, i)$  is attached to the row of  $\text{Mac}_{>,d}(F)$  containing the polynomial  $m f_i$ . More precisely, the coefficient of the row  $(m, i)$  and column  $(m')$  is the coefficient of the monomial  $m'$  in the polynomial  $m f_i$ . The rows are arranged in decreasing order as follows

$$(m, i) \succ (m', j) \Leftrightarrow i < j \text{ or } (i = j \text{ and } m > m').$$

The construction of the Macaulay matrix is depicted in Figure 2.3.

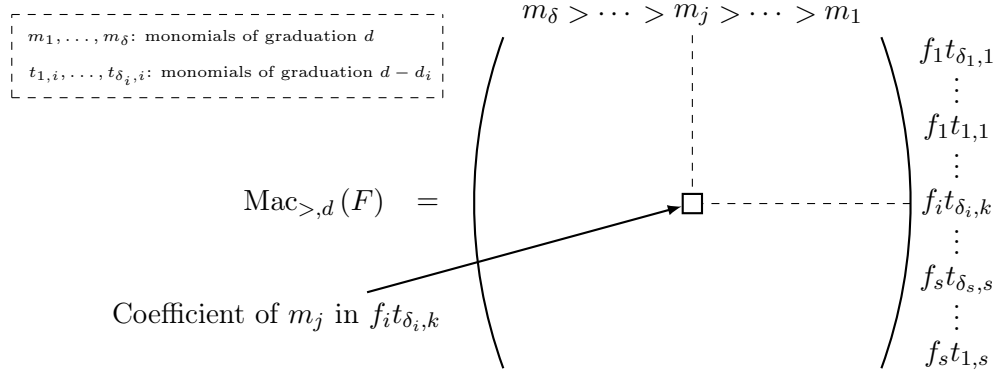


Figure 2.3: Macaulay matrix of  $(f_1, \dots, f_s)$  in graduation  $d$  w.r.t.  $>$ .

We can now describe the algorithm of Lazard (Algorithm 4). We denote by  $\widetilde{M}$  the reduced row echelon form of the matrix  $M$ . For a proof of completeness (termination is straightforward) see [Laz83].

---

**Algorithm 4:** Computing Gröbner bases by linear algebra: Lazard's algorithm.

---

**Input** : A sequence of homogeneous polynomial  $(f_1, \dots, f_s)$  of  $\mathbb{K}[x_1, \dots, x_n]$ , an integer  $d$  and a monomial ordering  $>$ .

**Output:** The reduced  $d$ -Gröbner basis of  $\langle f_1, \dots, f_s \rangle$  w.r.t.  $>$ .

```

1  $G := \{\}$ ;
2 for  $i := \min\{\Gamma(f_1), \dots, \Gamma(f_s)\}$  to  $d$  do
3    $M := \text{Mac}_{>,i}(f_1, \dots, f_s)$ ;
4    $\mathbf{m}_i :=$  column vector of size  $\binom{n+i-1}{i}$  containing all the monomials of graduation  $i$  in
    $\mathbb{K}[x_1, \dots, x_n]$  arranged in decreasing order w.r.t.  $>$ ;
5    $I_i := \widetilde{M} \cdot \mathbf{m}_i$ ;
6    $G := G \cup \{h \in I_i \mid \text{for all } g \in G \cup I_i \text{ s.t. } g \neq h, \text{LT}_{>}(g) \text{ does not divide } \text{LT}_{>}(h)\}$ ;
7 return  $G$ ;
```

---

When the ideal  $\mathcal{I}$  is of dimension zero then Algorithm 4 does not need the parameter  $d$  to ensure the termination. Indeed, for zero-dimensional ideals there exists an integer  $\delta$  such that

all the monomials of graduation  $d \geq \delta$  are in  $\text{in}_>(\mathcal{I})$ . Hence, at each step  $i$  one can check if all the monomials of degree  $i$  are in  $\langle \text{LT}_>(g_1), \dots, \text{LT}_>(g_r) \rangle$  with  $G = \{g_1, \dots, g_r\}$  to ensure that  $G$  is a Gröbner basis.

**Definition 2.62** (Homogenization). *Let  $(w_1, \dots, w_n)$  be the weights associated to the grading  $\Gamma$ . Let  $f$  be an affine polynomial of  $\mathbb{K}[x_1, \dots, x_n]$ . The homogenization  $\bar{f}$  of  $f$  is the polynomial of  $\mathbb{K}[x_1, \dots, x_n, x_0]$  equipped with the grading given by the weights system  $(w_1, \dots, w_n, 1)$  with  $x_1 > \dots > x_n > x_0$  defined by  $\bar{f} = x_0^\gamma f\left(\frac{x_1}{x_0}, \dots, \frac{x_n}{x_0}\right)$  where  $\gamma = \Gamma(f)$ .*

**Remark 2.63.** *Note that if  $f_1, \dots, f_s$  are affine polynomials one can compute a Gröbner basis of  $\langle f_1, \dots, f_s \rangle$  by using Algorithm 4. Indeed, it suffices to compute a Gröbner basis  $\bar{\mathcal{G}}_>$  of  $\langle \bar{f}_1, \dots, \bar{f}_s \rangle \subset \mathbb{K}[x_1, \dots, x_n, x_0]$  then a (possibly non reduced) Gröbner basis  $\mathcal{G}_>$  of  $\langle f_1, \dots, f_s \rangle$  is given by evaluating the variable  $x_0$  to one in the polynomials in  $\bar{\mathcal{G}}_>$ .*

In Lazard's algorithm or Buchberger's algorithm most of the time is spent to useless computations *i.e.* polynomials that are not added to  $G$  or rows identically null in the row echelon form of the matrix in Lazard's algorithm or equivalently polynomials which are reduced to zero in Buchberger's algorithm. The aim of the efficient algorithms to compute Gröbner bases is to avoid these useless computations.

### 2.2.2 Efficient algorithms for Gröbner bases: $F_4$ and $F_5$

The principle of  $F_4$  algorithm of Faugère is a clever mix between Buchberger's algorithm and Lazard's algorithm. The idea is to follow Buchberger's algorithm using critical pairs but reducing all the polynomials of same degree at the same time using linear algebra, as in Lazard's algorithm. The matrices involved in  $F_4$  algorithm are much smaller than in Lazard's algorithm. Indeed, they are constructed as sub-matrices of the Macaulay matrix by using the computation at the previous degree to remove useless rows. Moreover, even for ideals of positive dimension the algorithm  $F_4$  does not need in input the parameter  $d$  to terminate. Indeed, as Buchberger's algorithm it terminates when the set of critical pairs is empty. Although  $F_4$  algorithm is more efficient than Lazard's or Buchberger's algorithm it does not improve the complexity in the worst case of computing a Gröbner basis. Since  $F_4$  algorithm follows the principle of Buchberger's algorithm the Buchberger criterion [Buc06, Buc65, CLO07] allows to decrease the number of useless computation in comparison to Lazard's algorithm. However, useless computations are still the most time consuming step of  $F_4$  algorithm. There exists a powerful theoretical criterion ( $F_5$  criterion) to avoid useless computations but it is too costly to take it into account in  $F_4$  algorithm. This criterion allows to determine some useless rows in the Macaulay matrix. More precisely, it determines some rows that are linear combinations of the greater rows (w.r.t. the order on the signature of the rows of the Macaulay matrix). The aim of  $F_5$  algorithm is to provide an algorithm computing Gröbner bases with an efficient implementation of this criterion. More precisely, the aim of  $F_5$  algorithm is to construct only matrices with full rank.

**Theorem 2.64** ( $F_5$  criterion [Fau02]). *Let  $F = (f_1, \dots, f_s)$  be a sequence of homogeneous polynomials. Let  $(m, i)$  be the signature of a row of  $\text{Mac}_{>,d}(F)$ . If  $m \in \text{in}_>(\langle f_1, \dots, f_{i-1} \rangle)$  then the row  $(m, i)$  is a linear combination of the greater rows *i.e.* rows with greater signature.*

Here, we give only an outline of Matrix  $F_5$  algorithm [Fau02, Bar04] which is a variant of  $F_5$  algorithm more convenient for complexity analysis. However, the  $F_5$  algorithm is more efficient.

The idea in Matrix  $F_5$  algorithm is to add a level of iteration. Indeed, this algorithm still proceeds degree by degree but to compute the  $d$ -Gröbner basis it actually computes all the  $d$ -Gröbner bases of the ideals  $\langle f_1, f_2 \rangle, \langle f_1, f_2, f_3 \rangle, \dots, \langle f_1, \dots, f_s \rangle$ . The principle is that at step  $i$  we know the linear triangular bases of  $\langle f_1, \dots, f_{i-1} \rangle \cap \mathbb{K}[x_1, \dots, x_n]_j$  for all  $j = 1, \dots, d$ . Hence, using these bases and  $F_5$  criterion we can construct a sub-matrix of the Macaulay matrix in graduation  $d$  associated to  $f_1, \dots, f_i$  whose rows also generate the  $\mathbb{K}$ -vector space  $\langle f_1, \dots, f_i \rangle \cap \mathbb{K}[x_1, \dots, x_n]_d$ . The particularity of this matrix is that for almost all polynomial systems it will be of full rank and then we avoid useless computations. We make this statement more explicit in Theorem 2.67. We give in Algorithm 5 a description of Matrix  $F_5$  algorithm. Note that here the notation  $\widetilde{M}$  denotes the reduced row echelon form of the matrix  $M$  *without permutations of the rows*. Moreover, if  $(f, s)$  is a couple of polynomial and signature,  $M \text{ cat } (f, s)$  means the matrix  $M$  on which we add (by the bottom) the polynomial  $f$  with signature  $s$ . The notation  $\text{sign}(f)$  denotes the signature of the polynomial  $f$ .

---

**Algorithm 5:** Computing Gröbner bases by linear algebra: Matrix  $F_5$  algorithm.

---

**Input** : A sequence of homogeneous polynomials  $(f_1, \dots, f_s)$  of  $\mathbb{K}[x_1, \dots, x_n]$  with  $d_i = \Gamma(f_i)$  and  $d_1 \leq \dots \leq d_s$ , an integer  $d$  and a monomial ordering  $>$ .

**Output:** A  $d$ -Gröbner basis of  $\langle f_1, \dots, f_s \rangle$  w.r.t.  $>$ .

```

1  G := {f1, ..., fs};
2  for j := 1 to d do
3      M̃j,0 := empty matrix;
4      for i := 1 to n do
5          Mj,i := M̃j,i-1;
6          if di = j then Mj,i := Mj,i cat (fi, (1, i));
7          if j > di then
8              for f ∈ M̃j-1,i do
9                  (m, k) := signature of f;
10                 xλ := main variable of m (see Definition 2.55);
11                 for ℓ := λ to n do
12                     if xℓm is not a leading monomial of a row of M̃j-di,i-1 then
13                         Mj,i := Mj,i cat (xℓf, (xℓm, i));
14             G := G ∪ {f ∈ M̃j,i | ∄g ∈ Mj,i s.t. LT>(f) = LT>(g) and sign(f) = sign(g)};
15 return G;
```

---

For a proof of completeness or more details about (Matrix)  $F_5$  algorithm, see [Fau02, Bar04].

**Definition 2.65** (Regular sequence of polynomials). *Let  $F = (f_1, \dots, f_s)$  be a sequence of non-zero homogeneous polynomials of  $\mathbb{K}[x_1, \dots, x_n]$  and  $s \leq n$ . The sequence  $F$  is said to be regular if for all  $i \in \{1, \dots, s-1\}$ , the polynomial  $f_{i+1}$  does not divide 0 in the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\langle f_1, \dots, f_i \rangle$ .*

From Theorem 2.75 if a sequence of polynomials is regular then any permutation of this sequence forms a regular sequence of polynomials. Thus, we define a regular polynomial system as follows.

**Definition 2.66** (Regular systems). *A homogeneous polynomial system  $\{f_1, \dots, f_s\}$  is said to be regular if the sequence  $(f_1, \dots, f_s)$  is regular.*

**Theorem 2.67** ([Fau02]). *If the sequence  $(f_1, \dots, f_s)$  is a regular sequence of homogeneous polynomials then (Matrix)  $F_5$  algorithm generates only full rank matrices i.e. there is no reduction to zero.*

Let  $d_0$  be the integer such that every  $d$ -Gröbner basis of  $\mathcal{I}$  for  $d \geq d_0$  is a Gröbner basis of  $\mathcal{I}$ . The  $F_4$  and  $F_5$  algorithms have been design for monomial orderings implying that  $d_0$  is not too large. These algorithms are then particularly efficient for graded reverse lexicographical orderings but they are not efficient to compute lexicographical Gröbner bases since it may contain a polynomial whose degree is the degree of the ideal. However, we have seen in Section 2.1.4 that the Gröbner basis interesting for polynomial system solving is the LEX Gröbner basis. This issue motivates the usefulness of change of ordering algorithms. These algorithms take as input a Gröbner basis w.r.t. a first monomial ordering and compute a Gröbner basis of the same ideal w.r.t. a second monomial ordering. For instance, from the (W)DRL Gröbner basis (that we can compute with  $F_5$  algorithm) change of ordering algorithms allow to compute the LEX Gröbner basis which is more suitable for polynomial systems solving.

## 2.3 Change of ordering algorithms

In this section we give a precise description of different change of ordering algorithms. This section contains the algorithmic tools on which the results of Chapter 4 are based. In 1993, Faugère *et al.* showed in [FGLM93] that change of ordering for zero dimensional ideals is closely related to linear algebra. The next section is devoted to present their algorithm called FGLM in the literature.

All the ideals considered in this section are of dimension zero and  $D$  denotes the degree of the ideal i.e. the number of solutions counted with multiplicities in an algebraic closure of  $\mathbb{K}$ .

### 2.3.1 The FGLM algorithm

This algorithm proceeds in two stages. Let  $\mathcal{G}_{>_1}$  be the given reduced Gröbner basis w.r.t. the order  $>_1$  of an ideal  $\mathcal{I}$  in  $\mathbb{K}[x_1, \dots, x_n]$ . First, we need to compute the multiplicative structure of the quotient ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  seen as the  $\mathbb{K}$ -vector space  $\mathbb{V}_{>_1}(\mathcal{I})$ , see Proposition 2.28. That is to say the multiplication matrices  $T_1, \dots, T_n$  which are a matrix representation of the linear map  $\Lambda_i$  of  $\mathbb{V}_{>_1}(\mathcal{I})$  corresponding to the multiplication by  $x_i$  in  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . Once all the multiplication matrices are computed, the second Gröbner basis w.r.t. the new monomial ordering  $>_2$  is recovered by testing linear dependency of well chosen vectors.

#### Multiplication matrices

We denote by  $B_1 = \{\epsilon_D >_1 \dots >_1 \epsilon_1 = 1\}$  the canonical basis w.r.t.  $>_1$  of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . Hence,  $B_1$  is constructed as the monomials of  $\mathbb{K}[x_1, \dots, x_n]$  that are not divisible by a leading term of a polynomial in  $\mathcal{G}_{>_1}$ . To compute the multiplication matrices, we need to compute the normal forms of all monomials  $\epsilon_i x_j$  where  $1 \leq i \leq D$  and  $1 \leq j \leq n$ .

**Proposition 2.68** ([FGLM93]). *Let  $F = \{x_j \epsilon_i \mid 1 \leq i \leq D \text{ and } 1 \leq j \leq n\} \setminus B_1$  be the frontier. Let  $t = \epsilon_i x_j$  with  $i \in \{1, \dots, D\}$  and  $j \in \{1, \dots, n\}$ . We have the following three cases*

- I. *either  $t \in B_1$  and  $\text{NF}_{>_1}(t) = t$ ;*
- II. *or  $t = \text{LT}_{>_1}(g)$  for some  $g \in \mathcal{G}_{>_1}$  hence,  $\text{NF}_{>_1}(t) = t - g$ ;*
- III. *or  $t = x_k t'$  with  $t' \in F$  and  $\deg(t') < \deg(t)$ . Hence, if  $\text{NF}_{>_1}(t') = \sum_{l=1}^s \alpha_l \epsilon_l$  with  $t' >_1 \epsilon_s$ ,  $\text{NF}_{>_1}(t) = \text{NF}_{>_1}(x_k \text{NF}_{>_1}(t')) = \sum_{l=1}^s \alpha_l \text{NF}_{>_1}(\epsilon_l x_k)$ .*

From this proposition, it is not difficult to see that the normal form of all the monomials  $\epsilon_i x_j$  can be easily computed if we consider them in increasing order. Indeed, let  $t = \epsilon_i x_j$  for some  $i \in \{1, \dots, D\}$  and  $j \in \{1, \dots, n\}$ . Assume that we have already computed the normal form of all monomials less than  $t$  and of the form  $\epsilon_{i'} x_{j'}$ . If  $t$  is in  $B_1$  or is a leading term of a polynomial in  $\mathcal{G}_{>_1}$  then its normal form is trivially known. If  $t$  is of type (III) of Proposition 2.68 then  $t = x_k t'$  with  $t >_1 t'$  hence  $\text{NF}_{>_1}(t') = \sum_{l=1}^s \alpha_l \epsilon_l$  is known. Finally,  $\text{NF}_{>_1}(t) = \sum_{l=1}^s \alpha_l \text{NF}_{>_1}(x_k \epsilon_l)$  with  $x_k t' = t >_1 x_k \epsilon_l$  for all  $l = 1, \dots, s$ . Thus, the normal forms of  $x_k \epsilon_l$  are known for all  $l = 1, \dots, s$ . This yields the algorithm proposed in [FGLM93] that we summarize in Algorithm 6. We recall that  $\Phi$  denotes the isomorphism from  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  to  $\mathbb{V}_{>_1}(\mathcal{I})$ , see equation (2.1). Let  $M$  be a matrix,  $M[*, i]$  denotes the  $i$ th column of  $M$ .

---

**Algorithm 6:** Computing the multiplication matrices: the original algorithm.

---

**Input** : A reduced Gröbner basis  $\mathcal{G}_{>_1}$  w.r.t. the monomial ordering  $>_1$  of a zero dimensional ideal  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$ .

**Output:** The multiplication matrices  $T_1, \dots, T_n$  of  $\mathbb{V}_{>_1}(\mathcal{I})$ .

- 1 Compute  $B_1 = \{\epsilon_1, \dots, \epsilon_D\}$  and  $F = \{x_i \epsilon_j \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq D\} \setminus B_1$ ;
- 2  $F := \text{Sort}_{>_1}(F) = \{t_1, \dots, t_N\}$ ;
- 3  $T_1, \dots, T_n :=$  Null matrix of size  $D \times D$ ;
- 4  $\text{NF} := []$ ;
- 5 **for**  $i := 1$  to  $D$  **do**  $T_1, \dots, T_n := \text{UPDATE}(T_1, \dots, T_n, \epsilon_i, \epsilon_i)$ ;
- 6 **for**  $i := 1$  to  $N$  **do**
  - 7   **if** there exist  $g \in \mathcal{G}_{>_1}$  such that  $\text{LT}_{>_1}(g) = t_i$  **then**
  - 8      $T_1, \dots, T_n := \text{UPDATE}(T_1, \dots, T_n, t_i, t_i - g)$ ;
  - 9      $\text{NF}[t_i] := t_i - g$ ;
  - 10   **else**
  - 11     Find  $k$  and  $t_j$  with  $j < i$  such that  $t_i = x_k t_j$ ;
  - 12      $\mathbf{v} = \Phi(\text{NF}[t_j])$ ;  $\text{NF}[t_i] := \Phi^{-1}(T_k \cdot \mathbf{v})$ ;
  - 13      $T_1, \dots, T_n := \text{UPDATE}(T_1, \dots, T_n, t_i, \text{NF}[t_i])$ ;
- 14 **return**  $T_1, \dots, T_n$ ;

---

### Computing the new basis

The idea in [FGLM93], to compute the new basis is to note that if  $f = \sum_{\alpha} c_{\alpha} x^{\alpha}$  is a polynomial in  $\mathcal{G}_{>_2} \subset \mathcal{I}$  then its normal form w.r.t. the first ordering is zero. Hence,  $\text{NF}_{>_1}(f) = 0$  implies

**Algorithm 7:** UPDATE( $T_1, \dots, T_n, t, \text{nf}$ )

**Input** : The multiplication matrices  $T_1, \dots, T_n$  of  $\mathbb{V}_{>_1}(\mathcal{I})$  under construction, a monomial  $t = x_i \epsilon_j$  and its normal form  $\text{nf}$ .

**Output**: The multiplication matrices  $T_1, \dots, T_n$  of  $\mathbb{V}_{>_1}(\mathcal{I})$  updated with  $t$ .

```

1 for  $k = 1$  to  $n$  do
2   if there exists  $j$  such that  $t = x_k \epsilon_j$  then  $T_k[* , j] := \Phi(\text{nf})$ ;
3 return  $T_1, \dots, T_n$ ;

```

that  $\sum_{\alpha} c_{\alpha} T_1^{\alpha_1} \dots T_n^{\alpha_n} \mathbf{1} = \mathbf{0}$  where  $\mathbf{0}$  is the column vector  $(0, \dots, 0)$  and  $\mathbf{1}$  is the column vector  $(1, 0, \dots, 0)$ . That is to say there is a linear dependency between the vectors  $\mathbf{v}_{\alpha} = T_1^{\alpha_1} \dots T_n^{\alpha_n} \mathbf{1}$  such that  $c_{\alpha} \neq 0$ . By consequence, the principle of the algorithm is to enumerate the monomial in increasing order w.r.t.  $>_2$  and to compute the corresponding vector in  $\mathbb{V}_{>_1}(\mathcal{I})$ . If there is a linear combination between these vectors then we can deduce a polynomial in  $\mathcal{G}_{>_2}$  otherwise we found an element in the canonical basis w.r.t.  $>_2$  of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ .

We now describe more precisely the outline of the algorithm. We denote by  $B_2$  the canonical basis w.r.t.  $>_2$  of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ . Since  $\mathcal{I}$  is zero dimensional,  $D = \#B_2 \geq 1$  and then the monomial 1 is certainly the first element of  $B_2$ . Hence, initially  $B_2 = \{\epsilon_1 = 1\}$ . We denote by  $\mathbf{v}_m$  the vector representing the monomial  $m$  in  $\mathbb{V}_{>_1}(\mathcal{I})$  i.e. the coordinates vector of  $m$  w.r.t.  $B_1$ . The vector  $\mathbf{v}_1$  (corresponding to the monomial 1) is trivially known and is  $(1, 0, \dots, 0)$ . A subset of monomials that are not in  $B_2$  is denoted LT and is initially empty and the Gröbner basis  $\mathcal{G}_{>_2}$  too.

Then the current monomial to consider is constructed as the minimal monomial w.r.t.  $>_2$  which is on the frontier of the current basis  $B_2$  i.e.  $m = \min_{>_2} \{x_i \epsilon_j \mid 1 \leq i \leq n, \epsilon_j \in B_2 \text{ s.t. } x_i \epsilon_j \notin B_2 \cup \text{LT}\}$ . Following Proposition 2.68, the monomial  $m$  can be of three types

1.  $m$  has to be inserted in  $B_2$ ;
2.  $m$  is a leading monomial of a polynomial  $g$  which has to be inserted in  $\mathcal{G}_{>_2}$ ;
3.  $m$  is a strict multiple of a monomial in LT.

Checking the third case is easy since it suffices to check if a monomial in LT divides  $m$ . If it is not the case then to decide if  $m$  is of type 1 or 2 it suffices to check the linear dependency of the vectors  $\mathbf{v}_m$  and  $\mathbf{v}_{\epsilon}$  for all  $\epsilon \in B_2$ . If these vectors are linearly independent then we add  $m$  to  $B_2$  otherwise using the linear dependency we construct a polynomial  $g$  in  $\mathcal{I}$  that we add to  $\mathcal{G}_{>_2}$  and we add  $m$  to LT. FGLM algorithm is summarize in Algorithm 8.

Recently, Faugère and Mou have proposed in [FM11, FM13, Mou13] new change of ordering algorithms taking advantage of the sparsity of the multiplication matrices. They proposed two kinds of algorithms. The first is dedicated to change of ordering from any monomial ordering to LEX ordering and more precisely to ideals having a LEX Gröbner basis in *Shape Position*. As previously mentioned, most ideals (up to a linear change of variables) have a LEX Gröbner basis in *Shape Position*. Hence, these algorithms are very useful for polynomial systems solving. Indeed, since the shape of the expected Gröbner basis is known they design particularly efficient algorithms. The second kind of algorithm that they proposed is a general algorithm for change of ordering for Gröbner bases. Although, their algorithm can be more efficient in practice than FGLM the total complexity of their algorithm in terms of the degree of the ideal and the number of variables is not better than the complexity of FGLM.

**Algorithm 8:** A change of ordering algorithm for Gröbner bases: FGLM.

---

**Input** : The Gröbner basis  $\mathcal{G}_{>_1}$  w.r.t.  $>_1$  of  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  a zero dimensional ideal and a monomial ordering  $>_2$ .

**Output:** The Gröbner basis  $\mathcal{G}_{>_2}$  w.r.t.  $>_2$  of  $\mathcal{I}$ .

- 1 Compute the multiplication matrices  $T_1, \dots, T_n$  using Algorithm 6;
- 2  $B_2 := \{\varepsilon_1 = 1\}$ ;  $\mathbf{v}[1] := (1, 0, \dots, 0)$ ;  $\text{LT} := \emptyset$ ;  $\mathcal{G}_{>_2} := \emptyset$ ;
- 3  $L := \{x_i \varepsilon_j \mid 1 \leq i \leq n, \varepsilon_j \in B_2 \text{ s.t. } x_i \varepsilon_j \notin B_2 \cup \text{LT}\}$ ;
- 4 **while**  $\#L > 0$  **do**
- 5      $m :=$  minimum of  $L$  w.r.t. the monomial ordering  $>_2$ ;
- 6     **if** there exists  $m'$  in  $\text{LT}$  such that  $m'$  divides  $m$  **then**  $\text{LT} := \text{LT} \cup \{m\}$ ;
- 7     **else**
- 8         Find  $x_i$  and  $\varepsilon_j \in B_2$  such that  $m = x_i \varepsilon_j$ ;
- 9          $\mathbf{v}[m] := T_i \cdot \mathbf{v}[\varepsilon_j]$ ;
- 10        **if**  $\mathbf{v}[m]$  and  $\mathbf{v}[\varepsilon_k]$  for  $k = 1, \dots, \#B_2$  are linearly independent **then**
- 11             $B_2 := B_2 \cup \{m = \varepsilon_{\#B_2+1}\}$ ;
- 12        **else**
- 13             $\text{LT} := \text{LT} \cup \{m\}$ ;
- 14            Let  $c_0, \dots, c_{\#B_2} \in \mathbb{K}$  such that  $c_0 \mathbf{v}_m + \sum_{k=1}^{\#B_2} c_k \mathbf{v}[\varepsilon_k] = (0, \dots, 0)$ ;
- 15             $\mathcal{G}_{>_2} := \mathcal{G}_{>_2} \cup \{m + \sum_{k=1}^{\#B_2} \frac{c_k}{c_0} \varepsilon_k\}$ ;
- 16      $L := \{x_i \varepsilon_j \mid 1 \leq i \leq n, \varepsilon_j \in B_2 \text{ s.t. } x_i \varepsilon_j \notin B_2 \cup \text{LT}\}$ ;
- 17 **return**  $\mathcal{G}_{>_2}$ ;

---

Consequently, we do not present this algorithm here but we refer the interested reader to [FM13, Mou13]. On the other side we give a detailed description of their change of ordering algorithms dedicated to *Shape Position* ideals.

In [FM11], Faugère and Mou propose a probabilistic algorithm which given the reduced Gröbner basis w.r.t. a monomial ordering  $>_1$  of an ideal  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  computes the LEX Gröbner basis – if it is in *Shape Position* – of  $\mathcal{I}$ . Later, in [FM13] they proposed a deterministic version of their algorithm. We now describe these two algorithms. As a first step, we suppose that the multiplication matrix  $T_n$  is known.

### 2.3.2 Sparse change of ordering for Shape Position ideals: the probabilistic algorithm

Let  $\mathcal{G}_{>_{\text{lex}}} = \{h_n(x_n), x_{n-1} - h_{n-1}(x_n), \dots, x_1 - h_1(x_n)\}$  be the LEX Gröbner basis of an ideal  $\mathcal{I}$  in *Shape Position*. Given the multiplication matrices  $T_1, \dots, T_n$ , an algorithm to compute the LEX Gröbner basis of  $\mathcal{I}$  has to find the  $n$  univariate polynomials  $h_1, \dots, h_n$ . For this purpose, we can proceed in two steps. First, the polynomial  $h_n$  is computed. Then, by using linear algebra techniques, one computes the other univariate polynomials  $h_1, \dots, h_{n-1}$ .

#### Computation of $h_n$

To compute  $h_n$  one has to compute the minimal polynomial of  $T_n$ . To this end, we use the first part of the Wiedemann probabilistic algorithm which succeeds with good probability if

the field  $\mathbb{K}$  is sufficiently large, see [Wie86].

We recall that  $\Phi$  denotes the isomorphism from  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  to  $\mathbb{V}_{>1}(\mathcal{I})$ , see equation (2.1). Let  $\mathbf{r}$  be a random column vector in  $\mathbb{K}^D$  and  $\mathbf{1} = \Phi(1) = (1, 0, \dots, 0)^t$ . If  $\mathbf{a} = (a_1, \dots, a_D)$  and  $\mathbf{b} = (b_1, \dots, b_D)$  are two vectors of  $\mathbb{K}^D$ , we denote by  $(\mathbf{a}, \mathbf{b})$  the dot product of  $\mathbf{a}$  and  $\mathbf{b}$  defined by  $(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^D a_i b_i$ .

Let  $S = [(\mathbf{r}, T_n^j \mathbf{1}) \mid j = 0, \dots, 2D-1]$  be a linearly recurrent sequence of size  $2D$ . By using for instance the Berlekamp-Massey algorithm [Mas69], one can compute the minimal polynomial of  $S$  denoted  $\mu$ . If  $\deg(\mu(x_n)) = D$  then we deduce that  $\mu(x_n) = h_n(x_n) \in \mathcal{G}_{>\text{lex}}$  since  $\mu$  is a divisor of  $h_n$  of maximal degree.

In order to compute efficiently  $S$ , we first notice that  $(\mathbf{r}, T_n^j \mathbf{1}) = (T^j \mathbf{r}, \mathbf{1})$  where  $T = T_n^t$  is the transpose matrix of  $T_n$ . Then, since  $T_n$  is assumed to be sparse we compute iteratively all the matrix-vector products  $T^j \mathbf{r} = T(T^{j-1} \mathbf{r})$  for  $j := 1, \dots, 2D-1$ . Then, for  $j = 0, \dots, 2D-1$  the dot product  $(\mathbf{r}, T_n^j \mathbf{1})$  is the first component of the vector  $T^j \mathbf{r}$ .

### Recovering $h_1, \dots, h_{n-1}$

We write  $h_i = \sum_{k=0}^{D-1} c_{i,k} x_n^k$  where  $c_{i,k} \in \mathbb{K}$  are unknown. We have for  $i \in \{1, \dots, n-1\}$ :

$$x_i - h_i \in \mathcal{G}_{>\text{lex}} \text{ is equivalent to } 0 = \text{NF}_{>1} \left( x_i - \sum_{k=0}^{D-1} c_{i,k} x_n^k \right) = T_i \mathbf{1} - \sum_{k=0}^{D-1} c_{i,k} T_n^k \mathbf{1}.$$

Multiplying the last equation by  $T_n^j$  for any  $j = 0, \dots, (D-1)$  and taking the scalar product with  $\mathbf{r}$  we deduce that:

$$0 = (\mathbf{r}, T_n^j (T_i \mathbf{1})) - \sum_{k=0}^{D-1} c_{i,k} (\mathbf{r}, T_n^{k+j} \mathbf{1}) = (T^j \mathbf{r}, T_i \mathbf{1}) - \sum_{k=0}^{D-1} c_{i,k} (T^{k+j} \mathbf{r}, \mathbf{1}) \quad (2.4)$$

where  $T = T_n^t$ .

Hence, we can recover  $h_1, \dots, h_{n-1}$  by solving  $n-1$  structured linear systems:

$$\underbrace{\begin{pmatrix} (T^0 \mathbf{r}, T_i \mathbf{1}) \\ (T^1 \mathbf{r}, T_i \mathbf{1}) \\ \vdots \\ (T^{D-1} \mathbf{r}, T_i \mathbf{1}) \end{pmatrix}}_{\mathbf{b}_i} = \underbrace{\begin{pmatrix} (T^0 \mathbf{r}, \mathbf{1}) & (T^1 \mathbf{r}, \mathbf{1}) & \dots & (T^{D-1} \mathbf{r}, \mathbf{1}) \\ (T^1 \mathbf{r}, \mathbf{1}) & (T^2 \mathbf{r}, \mathbf{1}) & \dots & (T^D \mathbf{r}, \mathbf{1}) \\ \vdots & \vdots & \ddots & \vdots \\ (T^{D-1} \mathbf{r}, \mathbf{1}) & (T^D \mathbf{r}, \mathbf{1}) & \dots & (T^{2D-2} \mathbf{r}, \mathbf{1}) \end{pmatrix}}_{\mathcal{H}} \underbrace{\begin{pmatrix} c_{i,0} \\ c_{i,1} \\ \vdots \\ c_{i,D-1} \end{pmatrix}}_{\mathbf{c}_i} \quad (2.5)$$

Note that the linear system (2.5) has a unique solution since from [JM89] the rank of the Hankel matrix  $\mathcal{H}$  is given by the degree of the minimal polynomial of  $S$  which is exactly  $D$  in our case. The following lemma tells that we can compute  $T_i \mathbf{1}$  without knowing  $T_i$ .

**Lemma 2.69.** *For  $i \in \{1, \dots, n-1\}$  the vector  $T_i \mathbf{1}$  can be read from  $\mathcal{G}_{>1}$ .*

*Proof.* We have to consider the two cases  $\text{NF}_{>1}(x_i) \neq x_i$  or  $\text{NF}_{>1}(x_i) = x_i$ . First, if  $\text{NF}_{>1}(x_i) \neq x_i$  then there exists  $g \in \mathcal{G}_{>1}$  such that  $\text{LT}_{>1}(g)$  divides  $x_i$ . This implies that  $g$  is a linear equation:

$$x_i + \sum_{j>i}^n \alpha_{i,j} x_j + \alpha_{i,0} \text{ with } \alpha_{i,j} \in \mathbb{K}. \quad (2.6)$$

Hence, we have  $\text{NF}_{>_1}(x_i) = -\sum_{j>i}^n \alpha_{i,j}x_j - \alpha_{i,0}$  and the vector  $T_i\mathbf{1} = \Phi(x_i)$  is given by  $T_i\mathbf{1} = -[\alpha_{i,0}, 0, \dots, 0, \alpha_{i,i+1}, \dots, \alpha_{i,n}, 0, \dots]^t$ . Otherwise,  $\text{NF}_{>_1}(x_i) = x_i$  so that  $T_i\mathbf{1} = [0, \dots, 0, 1, 0, \dots, 0]^t$ .  $\square$

Hence, once the vectors  $T^j\mathbf{r}$  have been computed for  $j = 0, \dots, (2D-1)$ , one can deduce directly the Hankel matrix  $\mathcal{H}$  with no computation, but scalar products would seem to be needed to obtain the vectors  $\mathbf{b}_i$ . However, by removing the linear equations from  $\mathcal{G}_{>_1}$  one can deduce the  $\mathbf{b}_i$  without arithmetic operations since it suffices to extract a component of the vectors  $T^j\mathbf{r}$  for  $j \in \{0, \dots, D-1\}$ .

### Linear equations in $\mathcal{G}_{>_1}$

Let denote by  $\mathbb{L}$  the set of polynomials in  $\mathcal{G}_{>_1}$  of total degree 1 (usually  $\mathbb{L}$  is empty). We define  $\mathcal{L} = \{j \in \{1, \dots, n-1\} \text{ such that } \text{NF}_{>_1}(x_j) \neq x_j\}$  and  $\mathcal{L}^c = \{1, \dots, n-1\} \setminus \mathcal{L}$  so that  $\{x_i \mid i \in \mathcal{L}\} = \text{LT}_{>_1}(\mathbb{L})$ . In other words there is no linear form in  $\mathcal{G}_{>_1}$  with leading term  $x_i$  when  $i \in \mathcal{L}^c$ .

We first solve the linear systems (2.5) for  $i \in \mathcal{L}^c$ : we know from the proof of Lemma 2.69 that  $T_i\mathbf{1} = [0, \dots, 0, 1, 0, \dots, 0]^t$ . Hence, the components  $(T^j\mathbf{r}, T_i\mathbf{1})$  of the vector  $\mathbf{b}_i$  can be extracted directly from the vector  $T^j\mathbf{r}$ . By solving the corresponding linear system we can recover  $h_i(x_n)$  for all  $i \in \mathcal{L}^c$ .

Now we can easily recover the other univariate polynomials  $h_i(x_n)$  for all  $i \in \mathcal{L}$ : by definition of  $\mathcal{L}$  we have

$$l_i = x_i + \sum_{j \in \mathcal{L}^c} \alpha_{i,j}x_j + \alpha_{i,n}x_n + \alpha_{i,0} \in \mathbb{L} \subset \mathcal{G}_{>_1} \text{ with } \alpha_{i,j} \in \mathbb{K}.$$

Hence, the corresponding univariate polynomial  $h_i(x_n)$  is simply computed by the formula:

$$h_i(x_n) = - \sum_{j \in \mathcal{L}^c} \alpha_{i,j}h_j(x_n) - \alpha_{i,n}h_n(x_n) - \alpha_{i,0}.$$

Thus, we have reduced the number of linear systems (2.5) to solve from  $n-1$  to  $n - \#\mathcal{L} - 1$ . In the case where  $\mathbb{L}$  is empty, one still has  $n-1$  linear systems to solve but from Lemma 2.69 they are freely constructed from the vectors  $T^j\mathbf{r}$  for  $j \in \{0, \dots, D-1\}$ .

We conclude by summarizing the probabilistic algorithm to compute the LEX Gröbner basis of *Shape Position* ideals in Algorithm 9.

### 2.3.3 Sparse change of ordering for Shape Position ideals: the deterministic algorithm

The part of the Wiedemann algorithm used in Algorithm 9 to compute the minimal polynomial of  $T_n$  can fail (we can find only a factor) if the random vector  $\mathbf{r}$  is badly chosen. To avoid this phenomenon and the probabilistic nature of Algorithm 9 we can use the deterministic version of the Wiedemann algorithm [Wie86]. However, recovering the other polynomials  $h_1, \dots, h_{n-1}$  is much more difficult in this case. Here, we first recall the principle of the deterministic version of the Wiedemann algorithm. Then, we present the algorithm in [FM13] which given an ideal  $\mathcal{I}$  in *Shape Position* computes the LEX Gröbner basis of  $\sqrt{\mathcal{I}}$ . Note that in the context of polynomial systems solving this is not a restriction since the solutions of  $\sqrt{\mathcal{I}}$  and  $\mathcal{I}$  are the same.

**Algorithm 9:** Probabilistic change of ordering algorithm for *Shape Position* ideals.

---

**Input** : The multiplication matrix  $T_n$  and the reduced Gröbner basis  $\mathcal{G}_{>_1}$  w.r.t.  $>_1$  of an ideal  $\mathcal{I}$  of  $\mathbb{K}[x_1, \dots, x_n]$  in *Shape Position*.

**Output:** Return the LEX Gröbner basis  $\mathcal{G}_{>_{\text{lex}}}$  of  $\mathcal{I}$  or *fail*.

```

1 Randomly choose  $\mathbf{r}$  in  $\mathbb{K}^D$ ;
2  $\mathbf{v}_0 := \mathbf{r}; T := T_n^t$ ;
3 for  $i := 1$  to  $2D$  do  $\mathbf{v}_i := T\mathbf{v}_{i-1}$ ;
4 Deduce the linearly recurrent sequence  $S = [\mathbf{v}_i[1] \mid i = 0, \dots, 2D - 1]$  and the Hankel matrix  $\mathcal{H}$ ;
5  $h_n(x_n) := \text{BerlekampMassey}(S)$ ;
6 if  $\deg(h_n) = D$  then
7    $\mathcal{L}^c := \{j \in \{1, \dots, n-1\} \text{ such that } \text{NF}_{>_1}(x_j) = x_j\}$ ;
8    $\mathcal{L} := \{1, \dots, n-1\} \setminus \mathcal{L}^c$ ;
9   for  $j \in \mathcal{L}^c$  do
10    Deduce  $T_j\mathbf{1}$  and  $\mathbf{b}_j$  then solve the structured linear system  $\mathcal{H}\mathbf{c}_j = \mathbf{b}_j$ ;
11     $h_j(x_n) := \sum_{i=0}^{D-1} c_{j,i}x_n^i$  where  $c_{j,i}$  is the  $i$ th component of the vector  $\mathbf{c}_j$ ;
12   for  $j \in \mathcal{L}$  do
13     $h_j(x_n) := -\sum_{i \in \mathcal{L}^c} \alpha_{j,i}h_i(x_n) - \alpha_{j,n}h_n(x_n) - \alpha_{j,0}$  where  $\alpha_{j,i}$  is the  $i$ th coefficient of the linear form whose leading term is  $x_j$ ;
14   return  $[x_1 - h_1(x_n), \dots, x_{n-1} - h_{n-1}(x_n), h_n(x_n)]$ ;
15 else return fail;
```

---

**Deterministic computation of  $h_n$** 

Instead of randomly choosing a vector in  $\mathbb{K}^D$ , one can use the canonical vectors

$$\mathbf{e}_1 = (1, 0, \dots, 0)^t, \mathbf{e}_2 = (0, 1, 0, \dots, 0)^t, \dots, \mathbf{e}_D = (0, \dots, 0, 1)^t.$$

At each step we consider the linear recurrent sequence  $S_i = [(\mathbf{e}_i, T_n^j \mathbf{v}_i) \mid j = 0, \dots, 2D - 1]$ . The vector  $\mathbf{v}_i$  is chosen to ensure that the minimal polynomial  $f_i$  of  $S_i$  is a factor of  $\frac{h_n}{\prod_{j=1}^{i-1} f_j}$ .

By consequence, if  $\mathbf{v}_i = \mathbf{0}$  we know for sure that  $\prod_{j=1}^{i-1} f_j = h_n$ . A more precise description of this algorithm is given in Algorithm 10.

**Algorithm 10:** Computing  $h_n$  deterministically.

---

**Input** : The multiplication matrix  $T_n$  w.r.t. some monomial ordering of an ideal  $\mathcal{I}$  in *Shape Position*.

**Output:** The univariate polynomial  $h_n$  of the LEX Gröbner basis of  $\mathcal{I}$ .

```

1  $f := 1; \mathbf{v} := (1, 0, \dots, 0)^t; i := 1; d := D$ ;
2 repeat
3    $S := [(\mathbf{e}_i, T_n^j \mathbf{v}) \mid j = 0, \dots, 2d - 1]$ ;
4    $\mu := \text{Minimal polynomial of } S$ ;
5    $f := f\mu; d := D - \deg(f); \mathbf{v} := f(T_n)\mathbf{1}; i := i + 1$ ;
6 until  $\mathbf{v} = \mathbf{0}$ ;
7 return  $f$ ;
```

---

Algorithm 10 finishes for sure with  $r \leq D$  iterations and we have  $h_n = f_1 \cdots f_r$ . Note that for  $i = 2, \dots, r$  the sequence  $S_i$  is obtained by applying the polynomial  $f_1 \cdots f_{i-1}$  to the sequence  $[(\mathbf{e}_i, T_n^j \mathbf{1}) \mid j = 0, \dots, 2(D - \sum_{j=1}^{i-1} d_1) - 1]$  which can be done by multiplying polynomials, see [Wie86]. By consequence, the only matrix-vector products required are  $T_n^j \mathbf{1}$  for  $j = 0, \dots, 2D - 1$ .

### Recovering deterministically $h_1, \dots, h_{n-1}$

Assume the deterministic Wiedemann algorithm returns  $h_n = f_1 \cdots f_r$  with  $r \leq D$  and  $\deg f_i = d_i$ . At the  $i$ th step of the algorithm  $f_i$  is the minimal polynomial of the linearly recurrent sequence

$$S_i = \left[ (\mathbf{e}_i, T_n^j \mathbf{v}_{i-1}) \mid j = 0, \dots, 2 \left( D - \sum_{k=1}^{i-1} d_k \right) - 1 \right]$$

with  $\mathbf{v}_{i-1} = \prod_{k=1}^{i-1} f_k(T_n) \mathbf{1} = M_{i-1} \mathbf{1}$  where  $M_{i-1} = \prod_{k=1}^{i-1} f_k(T_n)$ . Moreover, we have

$$S_i = \left[ (M_{i-1}^t \mathbf{e}_i, T_n^j \mathbf{1}) \mid j = 0, \dots, 2 \left( D - \sum_{k=1}^{i-1} d_k \right) - 1 \right].$$

**Proposition 2.70** ([FM13]). *Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$  in Shape Position. Let  $T_1, \dots, T_n$  be the multiplication matrices of  $\mathcal{I}$  associated to the monomial ordering  $>_1$ . Let  $\mathbf{v}$  be some column vector of  $\mathbb{K}^D$ . Let  $S = [(\mathbf{v}, T_n^j \mathbf{1}) \mid j = 0, \dots, 2D - 1]$ . Let  $f$  be the minimal polynomial of  $S$  with  $\deg(f) = d < D$ . Then,  $\mathcal{J} = \mathcal{I} + \langle f \rangle$  is also a Shape Position ideal and for  $i = 1, \dots, n - 1$  the polynomial  $g_i = x_i - \sum_{k=0}^{d-1} c_{i,k} x_n^k$  is in the LEX Gröbner basis of  $\mathcal{J}$  where  $(c_{i,0}, \dots, c_{i,d-1})$  is the unique solution of the Hankel linear system*

$$(\mathbf{v}, T_n^j T_i \mathbf{1}) = \sum_{k=0}^{d-1} c_{i,k} (\mathbf{v}, T_n^{k+j} \mathbf{1}) \text{ for } j = 0, \dots, d-1.$$

By consequence, from the previous proposition for  $i = 1, \dots, r$  the LEX Gröbner basis of  $\mathcal{I} + \langle f_i \rangle$  can be computed by solving the  $n - 1$  following Hankel linear systems

$$\underbrace{\begin{pmatrix} (\mathbf{e}_i, T_n^0 M_{i-1} T_k \mathbf{1}) \\ (\mathbf{e}_i, T_n^1 M_{i-1} T_k \mathbf{1}) \\ \vdots \\ (\mathbf{e}_i, T_n^{d_i-1} M_{i-1} T_k \mathbf{1}) \end{pmatrix}}_{\mathbf{b}_{i,k}} = \underbrace{\begin{pmatrix} (\mathbf{e}_i, T_n^0 \mathbf{v}_{i-1}) & (\mathbf{e}_i, T_n^1 \mathbf{v}_{i-1}) & \dots & (\mathbf{e}_i, T_n^{d_i-1} \mathbf{v}_{i-1}) \\ (\mathbf{e}_i, T_n^1 \mathbf{v}_{i-1}) & (\mathbf{e}_i, T_n^2 \mathbf{v}_{i-1}) & \dots & (\mathbf{e}_i, T_n^{d_i} \mathbf{v}_{i-1}) \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{e}_i, T_n^{d_i-1} \mathbf{v}_{i-1}) & (\mathbf{e}_i, T_n^{d_i} \mathbf{v}_{i-1}) & \dots & (\mathbf{e}_i, T_n^{2(d_i-1)} \mathbf{v}_{i-1}) \end{pmatrix}}_{\mathcal{H}_i} \underbrace{\begin{pmatrix} c_{i,k,0} \\ c_{i,k,1} \\ \vdots \\ c_{i,k,d_i-1} \end{pmatrix}}_{\mathbf{c}_{i,k}} \quad (2.7)$$

with  $k \in \{1, \dots, n - 1\}$ . Note that the Hankel matrices  $\mathcal{H}_i$  are deduced with no cost from  $S_i$ . From Lemma 2.69, the vector  $\mathbf{w}_k = T_k \mathbf{1}$  can be computed without knowing  $T_k$ . To compute the vectors  $\mathbf{b}_{i,k}$  for  $i \in \{1, \dots, r\}$  and  $k \in \{1, \dots, n - 1\}$  we first compute the matrix-vector products  $T_n^j \mathbf{w}_k$  for  $j = 0, \dots, D - 1$ . Then, we extract the linearly recurrent sequence  $S_{i,k} = [(\mathbf{e}_i, T_n^j \mathbf{w}_k) \mid j = 0, \dots, D - 1]$ . Finally, from [Wie86] by applying the polynomial  $f_1 \cdots f_{i-1}$  on  $S_{i,k}$  we obtain the sequence  $[(\mathbf{e}_i, T_n^j M_{i-1} \mathbf{w}_k) \mid j = 0, \dots, D - 1]$  from which we can read the vector  $\mathbf{b}_{i,k}$ .

Finally, if  $\{x_1 - g_{1,i}, \dots, x_{n-1} - g_{n-1,i}, f_i\}$  is the LEX Gröbner basis of  $\mathcal{I} + \langle f_i \rangle$  then for  $j = 1, \dots, n-1$  the polynomials  $h_j$  satisfies the following equation set:

$$\begin{cases} h_j \equiv g_{j,1} \pmod{f_1} \\ \vdots \\ h_j \equiv g_{j,r} \pmod{f_r} \end{cases} \quad (2.8)$$

which can be solved using the *Chinese Remainder Theorem*, CRT for short, if all the  $f_i$ 's are pairwise coprime. When the ideal  $\mathcal{I}$  is itself radical then  $h_n$  is square free and consequently all the  $f_i$ 's are pairwise coprime. If  $\mathcal{I}$  is not radical then the CRT could not apply directly. However one can construct equation sets solvable by the CRT whose solutions gives the LEX Gröbner basis of  $\sqrt{\mathcal{I}}$ . For more details we refer the interested reader to [FM13].

### 2.3.4 Computation of $T_n$

One can notice that the two algorithms for *Shape Position* ideals (deterministic and probabilistic) take in input only the multiplication matrix  $T_n$ . The authors of [FM13, FM11] do not investigate the issue of computing the multiplication matrices except the computation of  $T_n$  in the generic case and when the first monomial ordering is the DRL ordering. More precisely they showed the following result.

**Proposition 2.71** ([FM13]). *Let  $\mathcal{I}$  be a generic ideal. Under the **Moreno-Sociás conjecture** (Conjecture 2.40), the matrix representation of the multiplication by  $x_n$  in  $\mathbb{V}_{>\text{drl}}(\mathcal{I})$ , can be read from  $\mathcal{G}_{>\text{drl}}$  without arithmetic operation.*

The last section of this chapter is devoted to the complexity analysis of polynomial systems solving by using Gröbner bases.

## 2.4 Complexity

Two important steps in the process of solving polynomial systems by using Gröbner bases is the computation of a first Gröbner basis w.r.t. a well-chosen ordering. Then, the computation of the LEX Gröbner basis is handled by a change of ordering algorithm. For this reason, we first investigate the complexity of Gröbner bases algorithms presented in Section 2.2. Then, we study the complexity of change of ordering algorithms presented in Section 2.3. Finally, the total complexity of polynomial systems solving is summarized.

In this thesis, all the systems we want to solve are of dimension zero. Hence, from now on we consider only zero dimensional ideals. Moreover, all the complexities mentioned in this thesis are arithmetic complexity that is which counts the number of operations in the field  $\mathbb{K}$ .

### 2.4.1 Gröbner bases algorithms

Since Gröbner bases algorithms have been design to graded reverse lexicographical ordering, we investigate their complexity only in the case of the WDRL ordering (including DRL ordering). Moreover, we have seen in Section 2.2 that Gröbner bases algorithms have been designed for homogeneous ideals. For this reason, we first study the complexity of Gröbner bases algorithms when the input ideal is homogeneous. Then, we will explain how this complexity can be extended to affine ideals.

We have seen that Lazard algorithm and the Matrix  $F_5$  algorithm take as input a parameter  $d$  and return a  $d$ -Gröbner basis. In order to obtain the Gröbner basis of the ideal we need to chose  $d$  large enough to ensure that the  $d$ -Gröbner basis is actually the Gröbner basis of the ideal. In  $F_4$  or  $F_5$  algorithm this parameter is not needed since they use the principle of critical pairs whose set becomes for sure empty once the Gröbner basis is computed. Anyway, the complexity of all these algorithms depends on the maximal graduation (*i.e.* degree or weighted degree) reached by the polynomials in the expected Gröbner basis. A common tool to estimate this degree is the Hilbert Series associated to  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ .

### Hilbert series

**Definition 2.72** (Hilbert function and Hilbert series). *Let  $\mathcal{I}$  be a homogeneous ideal of  $R = \mathbb{K}[x_1, \dots, x_n]$ . Let  $\Gamma$  be a grading on  $R$ . We denote by  $R_d$  the set of homogeneous (w.r.t.  $\Gamma$ ) polynomials of graduation  $d$  of  $R$  and  $\mathcal{I}_d = \mathcal{I} \cap R_d$ . The Hilbert function of  $\mathcal{I}$  is defined by*

$$\begin{aligned} \text{HF}_{R/\mathcal{I}} : \mathbb{N} &\rightarrow \mathbb{N} \\ d &\mapsto \dim_{\mathbb{K}}(R_d/\mathcal{I}_d) = \dim_{\mathbb{K}}(R_d) - \dim_{\mathbb{K}}(\mathcal{I}_d) \end{aligned}$$

and the Hilbert series of  $\mathcal{I}$  is defined by  $\text{HS}_{R/\mathcal{I}}(z) = \sum_{d=0}^{\infty} \text{HF}_{R/\mathcal{I}}(d) z^d \in \mathbb{N}[[z]]$ .

Recall that in our context  $\mathcal{I}$  is a zero dimensional ideal. Consequently, for any monomial ordering  $>$  the canonical basis of  $R/\mathcal{I}$  seen as a  $\mathbb{K}$ -vector space is finite and its size is equal to the degree of the ideal. By consequence, it is worth noting that for  $i = 1, \dots, n$  there exists an integer  $n_i \geq 1$  such that  $x_i^{n_i} \in \text{in}_{>}(\mathcal{I})$ . Therefore, there exists an integer  $d$  such that for all  $i \geq d$  and for all monomial  $m \in R_i$ ,  $m \in \text{in}_{>}(\mathcal{I})$  *i.e.*  $\dim_{\mathbb{K}}(R_i) = \dim_{\mathbb{K}}(\text{in}_{>}(\mathcal{I})_i)$ . Moreover, from [CLO07, p.463] for any monomial ordering  $>$ , the initial ideal  $\text{in}_{>}(\mathcal{I})$  has the same Hilbert function as  $\mathcal{I}$ . Consequently, for zero-dimensional ideal the Hilbert series of  $\mathcal{I}$  is in fact a polynomial. Furthermore, since the coefficient of  $z^d$  in the Hilbert series of  $\mathcal{I}$  is the dimension of  $R_d/\text{in}_{>}(\mathcal{I})_d$  that is the number of monomials in  $R_d$  that are in the canonical basis of  $\mathbb{V}_{>}(\mathcal{I})$  we have that  $\mathcal{D}_{\mathcal{I}} = \text{HS}_{R/\mathcal{I}}(1)$  for zero dimensional ideals.

**Definition 2.73** (Degree of regularity). *Let  $\mathcal{I}$  be a homogeneous zero dimensional ideal in the polynomial ring  $R = \mathbb{K}[x_1, \dots, x_n]$  equipped with the weighted degree with weights system  $(w_1, \dots, w_n)$ . The degree of regularity of  $\mathcal{I}$ , denoted  $d_{\text{reg}}(\mathcal{I})$ , is defined as follows*

$$d_{\text{reg}}(\mathcal{I}) = \deg(\text{HS}_{R/\mathcal{I}}(z)) + \max_{i=1, \dots, n} \{w_i\}.$$

*That is to say, when using the usual degree,  $d_{\text{reg}}(\mathcal{I})$  is the minimal graduation such that  $\dim_{\mathbb{K}}(R_i) = \dim_{\mathbb{K}}(\mathcal{I}_i)$  for all  $i \geq d_{\text{reg}}(\mathcal{I})$ .*

The following proposition allows to bound the maximal graduation reached by the polynomials in the reduced Gröbner basis w.r.t. any monomial ordering of homogeneous ideals.

**Proposition 2.74** ([FSV13, Laz83]). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be a homogeneous zero dimensional ideal. For any monomial ordering, the degree of regularity  $d_{\text{reg}}(\mathcal{I})$  bounds the graduation of all the polynomials in the reduced Gröbner basis of  $\mathcal{I}$ .*

*Proof.* Let  $>$  be any monomial ordering and  $\mathcal{G}_{>}$  the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. this monomial ordering. Assume there is a polynomial  $f \in \mathcal{G}_{>}$  such that  $\Gamma(f) > d_{\text{reg}}(\mathcal{I})$  then there exists a monomial  $m$  such that  $\deg(\text{HS}_{R/\mathcal{I}}(z)) < \Gamma(m) \leq d_{\text{reg}}(\mathcal{I})$  which divides  $\text{LT}_{>}(f)$ . Moreover, since  $\deg(\text{HS}_{R/\mathcal{I}}(z)) < \Gamma(m)$  then  $m \in \text{in}_{>}(\mathcal{I})$  thus there exists  $g \in \mathcal{G}_{>}$  such that  $\text{LT}_{>}(g)$  divides  $m$  and  $\text{LT}_{>}(f)$ . This contradicts the fact that  $\mathcal{G}_{>}$  is reduced.  $\square$

The Hilbert series of ideals generated by a regular sequence of polynomials is well understood. More precisely, we have the following result.

**Theorem 2.75** ([Sta78] cor. 3.3, [Bar04, Spa12]). *Let  $\mathcal{I} = \langle f_1, \dots, f_s \rangle$  with  $s \leq n$  and  $F = (f_1, \dots, f_s)$  be a sequence of homogeneous polynomials in  $R = \mathbb{K}[x_1, \dots, x_n]$  equipped with the weighted degree with weights system  $(w_1, \dots, w_n)$  (possibly  $(1, \dots, 1)$ ). Then, the three following statements are equivalent:*

1.  $F$  is a regular sequence;
2. the dimension of  $\mathcal{I}$  is  $n - s$ ;
3. the Hilbert series of  $\mathcal{I}$  is  $\text{HS}_{R/\mathcal{I}}(z) = \frac{\prod_{i=1}^s (1 - z^{\text{wdeg}(f_i)})}{\prod_{i=1}^n (1 - z^{w_i})}$ .

**Corollary 2.76.** *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  with  $F = (f_1, \dots, f_n)$  is a sequence of homogeneous polynomials in  $R = \mathbb{K}[x_1, \dots, x_n]$  equipped with the weighted degree with weights system  $(w_1, \dots, w_n)$ . If  $\mathcal{I}$  is of dimension zero (i.e.  $F$  is a regular sequence) then,*

- (weighted) Bézout's bound:  $\mathcal{D}_I = \text{HS}_{R/\mathcal{I}}(1) = \frac{\prod_{i=1}^n \text{wdeg}(f_i)}{\prod_{i=1}^n w_i}$ ;
- (weighted) Macaulay bound:  $d_{\text{reg}}(\mathcal{I}) = \max_{i=1, \dots, n} \{w_i\} + \sum_{i=1}^n (\text{wdeg}(f_i) - w_i)$ .

### Homogeneous ideals

The usual complexity bound to compute Gröbner bases is a bound on the complexity of Lazard's algorithm which is easy to analyse. However, this bound is not tight since it does not take into account the improvements in  $F_5$  algorithm (all the matrices are of full rank). Also this complexity does not take into account the structure of the Macaulay matrices (generalization of a Sylvester matrix). A precise analysis of Matrix  $F_5$  algorithm have been done by Bardet during her PhD thesis. More precisely, with her co-authors she obtained the following result.

**Theorem 2.77** ([Bar04, BFSY05, BFS04]). *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be a homogeneous ideal of dimension zero in  $R = \mathbb{K}[x_1, \dots, x_n]$  equipped with the usual degree. The arithmetic complexity of computing the reduced Gröbner basis (for any monomial ordering) of  $\mathcal{I}$  is bounded by*

$$O \left( \sum_{d=0}^{d_{\text{reg}}(\mathcal{I})} \binom{n+d-1}{d} \left( \sum_{i=1}^n \binom{n+d-\deg(f_i)-1}{d-\deg(f_i)} \right) \left( \binom{n+d-1}{d} - \text{HF}_{R/\mathcal{I}}(d) \right)^{\omega-2} \right)$$

arithmetic operations in  $\mathbb{K}$ . This bound can be bounded by the complexity of Lazard's algorithm:

$$O \left( n d_{\text{reg}}(\mathcal{I}) \binom{n+d_{\text{reg}}(\mathcal{I})-1}{n-1}^{\omega} \right) \leq O \left( n \binom{n+d_{\text{reg}}(\mathcal{I})}{n}^{\omega} \right). \quad (2.9)$$

Note that the complexity of Matrix  $F_5$  algorithm or Lazard's algorithm strongly relies on the maximal graduation reached by the polynomials. From Proposition 2.74 for homogeneous ideals whatever the monomial ordering, the degree of regularity bounds this graduation which explains why the complexity estimates of the above theorem do not depend of the monomial ordering.

The complexity of computing a Gröbner basis when the polynomial ring is equipped with the weighted degree has been tackled in [FSV13]. They obtain rigorous bound on the complexity of Matrix  $F_5$  algorithm (analogous to that in the previous theorem for the usual degree) but here we give only the one that we will use that is the upper bound obtained from Lazard's algorithm. The difference between the complexity of Lazard's algorithm for the usual degree and the weighted degree is the size of the Macaulay matrix in graduation  $d$ . Indeed, for a fixed system of weights  $(w_1, \dots, w_n)$  there are less monomials of weighted degree  $d$  than monomials of degree  $d$  (about  $\prod_{i=1}^n w_i$  times less).

**Theorem 2.78** ([FSV13]). *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be a homogeneous ideal of dimension zero in  $R = \mathbb{K}[x_1, \dots, x_n]$  equipped with the weighted degree with the weights system  $(w_1, \dots, w_n)$ . The arithmetic complexity of computing the reduced Gröbner basis (for any monomial ordering) of  $\mathcal{I}$  is bounded by*

$$\begin{aligned} & O \left( nd_{\text{reg}}(\mathcal{I}) \left( \frac{\gcd_{i=1, \dots, n} \{w_i\}}{\prod_{i=1}^n w_i} \binom{d_{\text{reg}}(\mathcal{I}) + S_n - 1}{n-1} \right)^\omega \right) \\ & \leq O \left( n \left( \frac{\gcd_{i=1, \dots, n} \{w_i\}}{\prod_{i=1}^n w_i} \binom{d_{\text{reg}}(\mathcal{I}) + S_n}{n} \right)^\omega \right) \end{aligned} \quad (2.10)$$

arithmetic operations where  $S_n$  is defined by  $S_1 = 0$  and  $S_i = S_{i-1} + w_i \frac{\gcd_{j=1, \dots, i-1} \{w_j\}}{\gcd_{j=1, \dots, i} \{w_j\}}$  for  $i \geq 2$ .

### Affine ideals

The case of affine ideals is more difficult to handle. Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an affine ideal of dimension zero i.e.  $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$  are affine polynomials. To compute the Gröbner basis of  $\mathcal{I}$  w.r.t. some monomial ordering one can compute the Gröbner basis of the homogenization of  $\mathcal{I}$  which is  $\bar{\mathcal{I}} = \langle \bar{f}_1, \dots, \bar{f}_n \rangle \subset \mathbb{K}[x_1, \dots, x_n, x_0]$ . However, even if the sequence  $(\bar{f}_1, \dots, \bar{f}_n)$  is regular we cannot apply the result of homogeneous ideals since in that case  $\bar{\mathcal{I}}$  is of dimension 1.

Moreover, for affine systems in contrary to homogeneous systems, some polynomials of graduation  $d$  in the ideal can be obtained by combination of polynomials of higher graduation i.e.

$$f = \sum_{i=1}^n h_i f_i \text{ and } \exists i \in \{1, \dots, n\} \text{ such that } \Gamma(h_i f_i) > d \text{ and } \Gamma(f) = d. \quad (2.11)$$

As this phenomenon, called *degree fall*, is difficult to anticipate, the complexity of Gröbner bases algorithms is very hard to handle and there is no general tight bound on this complexity. Indeed, the so-called *normal strategy* in  $F_4$  or  $F_5$  algorithm consists of considering critical pairs by increasing graduation. At step  $d$  if a degree fall occurs then instead of considering next critical pairs of graduation  $d+1$  we have to restart from the graduation of the degree fall. For affine systems, one can also apply Lazard's algorithm directly (obviously with Macaulay matrices containing affine polynomials so their columns are indexed with all the monomials of graduation less than or equal to  $d$ ). However, Lazard's algorithm does not take into account degree falls hence to compute the Gröbner basis we have to consider higher graduations than using  $F_4$  or  $F_5$  algorithms and no bound are known on the minimal graduation to consider.

Nevertheless, for some classes of affine polynomial systems this phenomenon of degree falls does not occur and the complexity of Gröbner bases algorithms are well handled. In

the literature [Eis95], the definition of affine regular systems is exactly the same as for the homogeneous case. However, under this hypothesis there is no guarantee that degree falls cannot occur. Let  $f_i^{(h)}$  be the homogeneous components of highest graduation of  $f_i$ . If  $F^{(h)} = (f_1^{(h)}, \dots, f_n^{(h)})$  form a regular sequence then it is shown in [Bar04, BFS04] that no degree fall can occur when computing the Gröbner basis w.r.t. a graded ordering (obviously the grading of the order is also the same grading as that of which  $\mathbb{K}[x_1, \dots, x_n]$  is equipped i.e. which defines  $f_i^{(h)}$ ). By consequence, we use the following definition for affine regular systems.

**Definition 2.79** (Affine regular systems). *Let  $F = (f_1, \dots, f_n)$  be a sequence of non-zero affine polynomials of  $\mathbb{K}[x_1, \dots, x_n]$ . The sequence  $F$  is said to be regular if the sequence  $F^{(h)} = (f_1^{(h)}, \dots, f_n^{(h)})$  is regular. An affine polynomial system is said to be regular if it is defined by an affine regular sequence.*

In the case where no degree fall can occur, the algorithms to compute Gröbner bases perform exactly the same computations to compute the Gröbner basis of  $\mathcal{I}$  as to compute the Gröbner basis of  $\mathcal{I}^{(h)} = \langle f_1^{(h)}, \dots, f_n^{(h)} \rangle$  except that the Macaulay matrices (or sub-matrices of Macaulay matrices) are larger. That is to say their column are indexed with all the monomials of graduation less than or equal to  $d$  instead of the monomials of graduation exactly  $d$ . By consequence, we use the following definition, introduced in [Bar04], of degree of regularity for affine systems.

If  $(f_1^{(h)}, \dots, f_n^{(h)})$  is regular then  $\mathcal{I}^{(h)}$  is of dimension zero as  $\mathcal{I}$ . Hence, in [Bar04] the following definition of degree of regularity for affine systems is introduced.

**Definition 2.80** (Degree of regularity of affine systems). *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an affine ideal in  $\mathbb{K}[x_1, \dots, x_n]$  such that  $(f_1^{(h)}, \dots, f_n^{(h)})$  is a regular sequence. The degree of regularity of  $\mathcal{I}$  is  $d_{\text{reg}}(\mathcal{I}) = d_{\text{reg}}(\mathcal{I}^{(h)})$ .*

**Proposition 2.81** ([Bar04]). *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an affine ideal in  $\mathbb{K}[x_1, \dots, x_n]$  such that  $(f_1^{(h)}, \dots, f_n^{(h)})$  is a regular sequence. The degree of regularity  $d_{\text{reg}}(\mathcal{I})$  of  $\mathcal{I}$  bounds the graduation of all the polynomials in the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. a graded ordering (e.g. graded reverse lexicographical ordering).*

Consequently, the arithmetic complexity of computing Gröbner bases w.r.t. graded orderings of ideals generated by an affine regular sequence is then given by replacing  $n$  by  $n + 1$  in the complexity for homogeneous regular ideals. For the case of weighted degree the  $(n + 1)$ th weight is then 1. This is summarized in the following corollary.

**Corollary 2.82.** *Lets fix  $\Gamma$  a grading on  $R = \mathbb{K}[x_1, \dots, x_n]$ . Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle \subset R$  be an affine ideal such that  $(f_1^{(h)}, \dots, f_n^{(h)})$  is a regular sequence (i.e. from Theorem 2.75  $\mathcal{I}^{(h)}$  is of dimension zero). The arithmetic complexity of computing the reduced Gröbner basis w.r.t. a graded ordering (whose grading is  $\Gamma$ ) of  $\mathcal{I}$  is bounded by*

$$O \left( nd_{\text{reg}}(\mathcal{I}) \binom{n + d_{\text{reg}}(\mathcal{I})}{n}^\omega \right) \leq O \left( n \binom{n + d_{\text{reg}}(\mathcal{I}) + 1}{n + 1}^\omega \right) \quad (2.12)$$

when  $\Gamma$  is the usual degree and

$$O \left( nd_{\text{reg}}(\mathcal{I}) \left( \frac{1}{\prod_{i=1}^n w_i} \left( d_{\text{reg}}(\mathcal{I}) + \sum_{i=1}^n w_i - 1 \right) \right)^\omega \right)$$

$$\leq O\left(n\left(\frac{1}{\prod_{i=1}^n w_i}\left(\frac{d_{\text{reg}}(\mathcal{I}) + \sum_{i=1}^n w_i}{n+1}\right)\right)^\omega\right) \quad (2.13)$$

when  $\Gamma$  is the weighted degree equipped with the weights systems  $(w_1, \dots, w_n)$ .

Actually, it seems that the complexity of computing Gröbner bases of affine ideals w.r.t. any graded monomial ordering is given by the complexity of computing the Gröbner basis of  $\mathcal{I}^{(h)}$  without paying the price of larger matrices. This is the statement of the following theorem from a work in progress by Bardet, Faugère and Salvy.

**Theorem 2.83.** *Lets fix  $\Gamma$  a grading on  $R = \mathbb{K}[x_1, \dots, x_n]$ . Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle \subset R$  be an affine ideal such that  $(f_1^{(h)}, \dots, f_n^{(h)})$  is a regular sequence (i.e. from Theorem 2.75  $\mathcal{I}^{(h)}$  is of dimension zero). The arithmetic complexity of computing the reduced Gröbner basis w.r.t. a graded ordering (whose grading is  $\Gamma$ ) of  $\mathcal{I}$  is bounded by*

$$O\left(nd_{\text{reg}}(\mathcal{I})\left(\frac{n + d_{\text{reg}}(\mathcal{I}) - 1}{n - 1}\right)^\omega\right) \leq O\left(n\left(\frac{n + d_{\text{reg}}(\mathcal{I})}{n}\right)^\omega\right) \quad (2.14)$$

when  $\Gamma$  is the usual degree and

$$\begin{aligned} & O\left(nd_{\text{reg}}(\mathcal{I})\left(\frac{\gcd_{i=1, \dots, n}\{w_i\}}{\prod_{i=1}^n w_i}\left(\frac{d_{\text{reg}}(\mathcal{I}) + S_n - 1}{n - 1}\right)\right)^\omega\right) \\ & \leq O\left(n\left(\frac{\gcd_{i=1, \dots, n}\{w_i\}}{\prod_{i=1}^n w_i}\left(\frac{d_{\text{reg}}(\mathcal{I}) + S_n}{n}\right)\right)^\omega\right) \end{aligned} \quad (2.15)$$

when  $\Gamma$  is the weighted degree equipped with the weights systems  $(w_1, \dots, w_n)$ ,  $S_n$  is defined by  $S_1 = 0$  and  $S_i = S_{i-1} + w_i \frac{\gcd_{j=1, \dots, i-1}\{w_j\}}{\gcd_{j=1, \dots, i}\{w_j\}}$  for  $i \geq 2$ .

In contrary to the computation of a Gröbner basis, for any classes of polynomial systems, the complexity of the second step in the resolution of polynomial systems is well understood. This is what we present in the next section.

## 2.4.2 Change of ordering

### Complexity of the FGLM algorithm

As mentioned in Section 2.3.1, FGLM algorithm is split into two steps: the computation of the multiplication matrices and then the computation of the new basis.

It is not difficult to see that the complexity of computing the multiplication matrices, Algorithm 6, is dominated by the cost of computing the normal forms of monomials of the form  $x_i \epsilon_j$  that are neither in the canonical basis  $B_1$  nor in the stair of  $\mathcal{I}$  w.r.t.  $>_1$ . That is to say monomials of type (III) of Proposition 2.68. Since, each of these normal forms is computed by a matrix-vector product of size  $(\mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}) \times (\mathcal{D}_{\mathcal{I}} \times 1)$  and the number of monomials of the form  $x_i \epsilon_j$  is bounded by  $nD$  we get the following result.

**Proposition 2.84** ([FGLM93]). *Given the reduced Gröbner basis w.r.t. a monomial ordering  $>_1$  of a zero-dimensional ideal  $\mathcal{I}$  in  $\mathbb{K}[x_1, \dots, x_n]$  the complexity of computing the  $n$  multiplication matrices  $T_1, \dots, T_n$  (i.e. a matrix representation of the multiplication by  $x_1, \dots, x_n$  in  $\mathbb{V}_{>_1}(\mathcal{I})$ ) i.e. the complexity of Algorithm 6 can be bounded by  $O(n\mathcal{D}_{\mathcal{I}}^3)$  arithmetic operations. Where  $\mathcal{D}_{\mathcal{I}}$  denotes the degree of  $\mathcal{I}$ .*

Let  $\Phi$  be the isomorphism from  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  to  $\mathbb{V}_{>_1}(\mathcal{I})$ . Given the  $n$  multiplication matrices, the complexity of FGLM, Algorithm 8, is given by the complexity of testing the linear dependency of  $\Phi(m)$  with  $\Phi(\varepsilon_i)$  for all  $m$  in  $B_2$  or in  $E_{>_2}(\mathcal{I})$  and for all  $m >_2 \varepsilon_i \in B_2$ . Note that if a row echelon form of the matrix containing the vector  $\Phi(\varepsilon_i)$  for all  $\varepsilon_i \in B_2$  already computed is maintained, then one linear dependency test can be done in  $O(\mathcal{D}_{\mathcal{I}}^2)$  arithmetic operations. Moreover, from [FGLM93] the number of polynomials in any reduced Gröbner basis is bounded by  $n\mathcal{D}_{\mathcal{I}}$ . Then we obtain the following result.

**Theorem 2.85** ([FGLM93]). *Given the reduced Gröbner basis w.r.t. a monomial ordering  $>_1$  of a zero-dimensional ideal  $\mathcal{I}$  in  $\mathbb{K}[x_1, \dots, x_n]$  and a monomial ordering  $>_2$ ; the complexity of computing the reduced Gröbner basis of  $\mathcal{I}$  w.r.t.  $>_2$  i.e. the complexity of Algorithm 8 can be bounded by  $O(n\mathcal{D}_{\mathcal{I}}^3)$  arithmetic operations. Where  $\mathcal{D}_{\mathcal{I}}$  denotes the degree of  $\mathcal{I}$ .*

### Probabilistic change of ordering for Shape Position ideals

Let  $\mathcal{I}$  be an ideal in *Shape Position* and let  $\mathcal{G}_{>_1}$  be the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. the monomial ordering  $>_1$ . Let  $T_n$  be the multiplication matrix by  $x_n$  in  $\mathbb{V}_{>_1}(\mathcal{I})$ . The complexity of Algorithm 9 to compute the LEX Gröbner basis of  $\mathcal{I}$  given  $\mathcal{G}_{>_1}$  and  $T_n$  is given by the complexity of computing the matrix-vector product  $(T_n^t)^j \mathbf{r}$  for  $j = 0, \dots, 2\mathcal{D}_{\mathcal{I}} - 1$  and the complexity of solving at most  $n$  linear Hankel systems.

**Theorem 2.86** ([FM11],[FM13],[Mou13]). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal in Shape Position. Given the reduced Gröbner basis of  $\mathcal{I}$  w.r.t.  $>_1$  and the multiplication matrix by the smallest variables in  $\mathbb{V}_{>_1}(\mathcal{I})$ , there exists a probabilistic algorithm computing the LEX Gröbner basis of  $\mathcal{I}$  in  $O(\#T_n \mathcal{D}_{\mathcal{I}} + n \log_2(\mathcal{D}_{\mathcal{I}})^2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2(\mathcal{D}_{\mathcal{I}}))$  arithmetic operations. Where  $\#T_n$  denotes the number of nonzero entries in  $T_n$  and  $\mathcal{D}_{\mathcal{I}}$  the degree of  $\mathcal{I}$ .*

If the first monomial ordering is the DRL ordering, from Proposition 2.71, for generic ideals the multiplication matrix  $T_n$  can be computed without arithmetic operations. This yields the following result.

**Corollary 2.87** ([FM13],[Mou13]). *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal in Shape Position with  $(f_1, \dots, f_n)$  a generic sequence of polynomials. Given the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. DRL ordering, under the Moreno-Socias conjecture (Conjecture 2.40), there exists a probabilistic algorithm computing the LEX Gröbner basis of  $\mathcal{I}$  in  $O(\#T_n \mathcal{D}_{\mathcal{I}} + n \log_2(\mathcal{D}_{\mathcal{I}})^2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2(\mathcal{D}_{\mathcal{I}}))$  arithmetic operations. Where  $\#T_n$  denotes the number of nonzero entries in the multiplication matrix  $T_n$  and  $\mathcal{D}_{\mathcal{I}}$  the degree of  $\mathcal{I}$ . When  $f_1, \dots, f_n$  are of same degree  $d$  tending to infinity and  $n$  is fixed (still under the Moreno-Socias conjecture) the percentage of non-zero entries in  $T_n$  is  $\sim \sqrt{\frac{6}{n\pi d^2}}$ . Hence, the complexity of computing the LEX Gröbner basis is bounded by  $O\left(\sqrt{\frac{6}{n\pi}} \mathcal{D}_{\mathcal{I}}^{2+\frac{n-1}{n}}\right)$  arithmetic operations.*

### Deterministic change of ordering for Shape Position ideals

The complexity to compute the univariate polynomial  $h_n$  is given by the complexity of computing the first linear recurrent sequence  $S_1 = [(e_1, T_n^j \mathbf{1}) \mid j = 0, \dots, 2\mathcal{D}_{\mathcal{I}} - 1]$  and the complexity of the deterministic version of the Wiedemann algorithm. The cost to compute  $S_1$  is the cost to compute the matrix-vector products  $T_n^j \mathbf{1}$  for  $j = 0, \dots, 2\mathcal{D}_{\mathcal{I}} - 1$ . That is

to say  $O(\#T_n \mathcal{D}_{\mathcal{I}})$  arithmetic operations. From [Wie86], the other linearly recurrent sequence  $S_i$  for  $i = 1, \dots, r \leq \mathcal{D}_{\mathcal{I}}$  can be computed by applying the polynomial  $f_1 \cdots f_{i-1}$  to the sequence  $S_{i-1}$  in  $O(\mathcal{D}_{\mathcal{I}} \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations. Moreover, the Berlekamp-Massey algorithm computes the minimal polynomial of  $S_i$  in  $O(\mathcal{D}_{\mathcal{I}} \log_2 (\mathcal{D}_{\mathcal{I}})^2)$  arithmetic operations, see [BGY80, JM89]. Consequently, computing the polynomial  $h_n$  can be done in  $O(\#T_n \mathcal{D}_{\mathcal{I}} + \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations.

Then, if the Wiedemann algorithm returns  $h_n = f_1 \cdots f_r$ , for  $i = 1, \dots, r \leq \mathcal{D}_{\mathcal{I}}$  we have to compute  $n$  Hankel linear systems of size  $(d_i \times d_i)$  with  $d_i = \deg(f_i)$ . This can be done in  $O(\sum_{i=1}^r n d_i \log_2 (d_i)^2) \leq O(n \log_2 (\mathcal{D}_{\mathcal{I}})^2 \sum_{i=1}^r d_i) = O(n \log_2 (\mathcal{D}_{\mathcal{I}})^2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations. The Hankel matrices  $\mathcal{H}_i$  for  $i = 1, \dots, r$  are deduced with no cost from the linearly recurrent sequence  $S_i$ . The construction of the vectors  $\mathbf{b}_{i,k}$  for  $i = 1, \dots, r$  and  $k = 1, \dots, n$  follows the same idea as in the Wiedemann algorithm (except we do not need the minimal polynomial of the sequence that we compute). First we compute the matrix-vector product  $T_n^j(T_k \mathbf{1})$  for  $j = 0, \dots, \mathcal{D}_{\mathcal{I}} - 1$  and  $k = 1, \dots, n$  in  $O(n \mathcal{D}_{\mathcal{I}} \#T_n)$  arithmetic operations. Then the linearly recurrent sequences  $S_{i,k} = [(e_i, T_n^j T_k \mathbf{1}) \mid j = 0, \dots, \mathcal{D}_{\mathcal{I}} - 1]$  for  $i = 1, \dots, r$  and  $k = 1, \dots, n$  are deduced with no cost. Finally, for  $i = 1, \dots, r$  and  $k = 1, \dots, n$  the vectors  $\mathbf{b}_{i,k}$  are obtained by applying  $f_1 \cdots f_{i-1}$  on  $S_{i,k}$  in  $O(n \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations. Finally, the total complexity to compute the LEX Gröbner basis of  $\mathcal{I} + \langle f_i \rangle$  for  $i = 1, \dots, r$  is bounded by  $O(n \mathcal{D}_{\mathcal{I}} \#T_n + n \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations.

Finally, from the LEX Gröbner basis of  $\mathcal{I} + \langle f_i \rangle$  for  $i = 1, \dots, r$  recovering the LEX Gröbner basis of the radical of  $\mathcal{I}$  can be done in  $O(n \mathcal{D}_{\mathcal{I}}^2 + \mathcal{D}_{\mathcal{I}}^2 \log \mathcal{D}_{\mathcal{I}}) (+O(\mathcal{D}_{\mathcal{I}} \log_2 \frac{q}{p}))$  if  $\mathbb{K} = \mathbb{F}_q$  with  $q = p^k$  coming from the complexity of computing the square-free part of a polynomial) arithmetic operations, see [FM13, Mou13] for more details. The following theorem summarized the total complexity of the deterministic algorithm for change of ordering for *Shape Position* ideals.

**Theorem 2.88** ([FM13, Mou13]). *Let  $\mathcal{I} \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal in Shape Position. Given the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. some monomial ordering  $>_1$  and the multiplication matrix  $T_n$ , there exists a deterministic algorithm computing the LEX Gröbner basis of  $\mathcal{I}$  in*

- $O(n \mathcal{D}_{\mathcal{I}} \#T_n + n \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations if  $\mathbb{K}$  is a field of characteristic zero;
- $O\left(n \mathcal{D}_{\mathcal{I}} \#T_n + n \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}} + \mathcal{D}_{\mathcal{I}} \log_2 \frac{q}{p}\right)$  arithmetic operations if  $\mathbb{K}$  is a finite field of characteristic  $p$  and size  $q$ ;

where  $\mathcal{D}_{\mathcal{I}}$  is the degree of  $\mathcal{I}$  and  $\#T_n$  is the number of nonzero entries in  $T_n$ .

Since the deterministic algorithm still takes as input only the multiplication matrix  $T_n$ , we get the following result.

**Corollary 2.89.** *Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle \subset \mathbb{K}[x_1, \dots, x_n]$  be an ideal in Shape Position with  $(f_1, \dots, f_n)$  a generic sequence of polynomials. Given the reduced Gröbner basis of  $\mathcal{I}$  w.r.t. DRL ordering, **under the Moreno-Sociás conjecture** (Conjecture 2.40), there exists a deterministic algorithm computing the LEX Gröbner basis of  $\mathcal{I}$  in*

- $O(n \mathcal{D}_{\mathcal{I}} \#T_n + n \mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}})$  arithmetic operations if  $\mathbb{K}$  is a field of characteristic zero;

- $O\left(n\mathcal{D}_{\mathcal{I}}\#T_n + n\mathcal{D}_{\mathcal{I}}^2 \log_2 \mathcal{D}_{\mathcal{I}} \log_2 \log_2 \mathcal{D}_{\mathcal{I}} + \mathcal{D}_{\mathcal{I}} \log_2 \frac{q}{p}\right)$  arithmetic operations if  $\mathbb{K}$  is a finite field of characteristic  $p$  and size  $q$ .

Where  $\#T_n$  denotes the number of nonzero entries in the multiplication matrix  $T_n$  and  $\mathcal{D}_{\mathcal{I}}$  the degree of  $\mathcal{I}$ . When  $f_1, \dots, f_n$  are of same degree  $d$  tending to infinity and  $n$  is fixed (still under the Moreno-Socías conjecture) the percentage of non-zero entries in  $T_n$  is  $\sim \sqrt{\frac{6}{n\pi d^2}}$ . Hence, the complexity of computing the LEX Gröbner basis is bounded by  $O\left(\sqrt{\frac{6n}{\pi}} \mathcal{D}_{\mathcal{I}}^{2+\frac{n-1}{n}}\right)$  ( $+O(\mathcal{D}_{\mathcal{I}} \log_2 \frac{q}{p})$  if  $\mathbb{K} = \mathbb{F}_q$  with  $q = p^k$ ) arithmetic operations.

We conclude this chapter by summarizing our strategy and its complexity for polynomial systems solving using Gröbner bases.

### 2.4.3 Polynomial systems solving

Given a set of polynomial equations  $\mathcal{S} = \{f_1, \dots, f_s\} \subset \mathbb{K}[x_1, \dots, x_n]$  solving this system has many meanings which depends especially on the field  $\mathbb{K}$ . For this reason, in order to stick to the most general case throughout this thesis we mean by solving computing the LEX Gröbner basis of the ideal  $\langle f_1, \dots, f_s \rangle = \mathcal{I}$ . From this, since in our context  $\mathcal{I}$  is of dimension zero, the *resolution* of the system is reduced to *solve* univariate polynomials of degree at most the degree of the ideal. Hence, for completeness we give, in Table 2.1, the complexity to solve a univariate polynomial for different meanings of solving. The notation  $\tilde{O}$  means that we neglect logarithmic factors in the degree of the polynomial and depending on the field also the size of the coefficients.

| $\mathbb{K}$   | Meaning of solving                                                | Complexity                                           |
|----------------|-------------------------------------------------------------------|------------------------------------------------------|
| $\mathbb{F}_q$ | Enumerate all the solutions in $\mathbb{F}_q$                     | $O(d \log^2 d \log dq \log \log d)$ [VZGG03, p.382]  |
|                | Enumerate all the solutions in $\overline{\mathbb{F}_q}$          | $O(d^2 \log^2 d \log q \log \log d)$ [VZGG03, p.382] |
| $\mathbb{Q}$   | Enumerate all the roots in $\mathbb{Q}$                           | $\tilde{O}(d^2 s)$ [VZGG03, p.444]                   |
| $\mathbb{C}$   | Approximation with precision $O(b)$ bits of all the complex roots | $O(d \log^2 d (\log^2 d + \log b))$ [Pan02]          |

Table 2.1: Complexity to solve a univariate polynomial of degree  $d$  in number of operations in  $\mathbb{K}$ . When  $\mathbb{K} = \mathbb{Q}$ ,  $s$  denotes the size of the coefficients of the polynomial and the complexity is given in number of word operations.

To compute the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$ , we use the usual algorithm (Algorithm 11) which consists of first fixing a grading on  $\mathbb{K}[x_1, \dots, x_n]$  and computing the graded reverse lexicographical Gröbner basis of  $\langle \mathcal{S} \rangle$ . Then, using a change of ordering algorithm we can compute the lexicographical Gröbner basis of  $\langle \mathcal{S} \rangle$ . According to the complexity of  $F_5$  algorithm, this algorithm is more efficient than computing directly the LEX Gröbner basis by using  $F_5$ .

The complexity of Algorithm 11 is then given by the complexity of the algorithm used to compute the GRL Gröbner basis (e.g.  $F_5$  algorithm) and the complexity of change of ordering algorithm. Consequently, the following result is deduced from Theorems 2.77, 2.78, 2.83, 2.85 and Corollary 2.87.

**Proposition 2.90.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$ . Assume the ring  $\mathbb{K}[x_1, \dots, x_n]$  is equipped with the weighted degree with weights  $(w_1, \dots, w_n)$ . If  $(f_1, \dots, f_n)$  is a regular*

**Algorithm 11:** Polynomial systems solving**Input** : A polynomial system  $\mathcal{S} \subset \mathbb{K}[x_1, \dots, x_n]$ .**Output:** The LEX Gröbner basis of  $\langle \mathcal{S} \rangle$ .

- 1 Fix a grading on  $\mathbb{K}[x_1, \dots, x_n]$ ;
- 2 Computing the GRL Gröbner basis of  $\langle \mathcal{S} \rangle$ ;
- 3 From the GRL Gröbner basis, computing the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$ ;
- 4 **return** The LEX Gröbner basis of  $\langle \mathcal{S} \rangle$ ;

sequence then the arithmetic complexity of Algorithm 11 to compute the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$  can be bounded by

$$O \left( n \left( \frac{\gcd_{i=1, \dots, n} \{w_i\}}{\prod_{i=1}^n w_i} \binom{d_{\text{reg}}(\mathcal{I}) + S_n}{n} \right)^\omega + n \mathcal{D}_{\langle \mathcal{S} \rangle}^3 \right).$$

When  $n$  is fixed and the degrees of  $f_1, \dots, f_n$  are uniformly bounded by  $d$  which tends to infinity, this complexity can be decreased to

$$O \left( n \left( \frac{\gcd_{i=1, \dots, n} \{w_i\}}{\prod_{i=1}^n w_i} \binom{d_{\text{reg}}(\mathcal{I}) + S_n}{n} \right)^\omega + \sqrt{\frac{6}{n\pi}} \mathcal{D}_{\langle \mathcal{S} \rangle}^{2+\frac{n-1}{n}} \right)$$

if  $\langle \mathcal{S} \rangle$  is a generic ideal in Shape Position; where  $\mathcal{D}_{\langle \mathcal{S} \rangle}$  denotes the degree of  $\langle \mathcal{S} \rangle$  and  $S_n$  is defined by  $S_1 = 0$  and  $S_i = S_{i-1} + w_i \frac{\gcd_{j=1, \dots, i-1} \{w_j\}}{\gcd_{j=1, \dots, i} \{w_j\}}$  for  $i \geq 2$ .

Let  $\delta = \gcd_{i=1, \dots, n} \{w_i\}$  considering the weights system  $(w_1, \dots, w_n)$  or  $(\frac{w_1}{\delta}, \dots, \frac{w_n}{\delta})$  does not change the degree of the ideal. Moreover, the number of monomials of weighted degree  $d$  considering the weights  $(w_1, \dots, w_n)$  is exactly the same as the number of monomials of weighted degree  $\frac{d}{\delta}$  considering the weights  $(\frac{w_1}{\delta}, \dots, \frac{w_n}{\delta})$ . Consequently, whatever the weight systems  $(w_1, \dots, w_n)$ , the complexity of computing the WDRL Gröbner basis of an ideal  $\mathcal{I}$  is the same for all weights systems  $(\alpha w_1, \dots, \alpha w_n)$  with  $\alpha \in \mathbb{N}^*$ . Consequently, without loss of generality, we can assume that  $\gcd_{i=1, \dots, n} \{w_i\} = 1$ .

Furthermore, in case of homogeneous systems Corollary 2.76 gives an explicit value for  $d_{\text{reg}}(\mathcal{I})$  and  $\mathcal{D}_{\mathcal{I}}$  in terms of the degree of the input equations. Indeed, the Macaulay bound implies that  $d_{\text{reg}}(\mathcal{I}) = \max_{i=1}^n \{w_i\} + \sum_{i=1}^n (\text{wdeg}(f_i) - w_i)$  and the Bézout bound implies that  $\mathcal{D}_{\mathcal{I}} = \frac{\prod_{i=1}^n \text{wdeg}(f_i)}{\prod_{i=1}^n w_i}$ . In case of affine systems, these equalities become bounds on  $d_{\text{reg}}(\mathcal{I})$  and  $\mathcal{D}_{\mathcal{I}}$ . However, in general these bounds are reached. We mean by in general that the set of affine systems for which these bounds are reached forms a Zariski open set. In particular, generic affine ideals (Definitions 2.34 and 2.16) reached these bounds.

**Corollary 2.91.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$ . Assume the ring  $\mathbb{K}[x_1, \dots, x_n]$  is equipped with the weighted degree with weights  $(w_1, \dots, w_n)$ . Assume there exists  $i \in \{1, \dots, n\}$  such that  $w_i = 1$ . Then, we can order the  $w_i$ 's to ensure that  $S_n < \sum_{i=1}^n w_i$ . Let  $\Delta = \prod_{i=1}^n w_i$ . Assume that  $\text{wdeg}(f_1), \dots, \text{wdeg}(f_n)$  are uniformly bounded by the parameter  $d$ . If  $(f_1, \dots, f_n)$  is a regular sequence then the arithmetic complexity of Algorithm 11 to compute the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$  can be bounded by*

$$\bullet O \left( \frac{d^{\omega n}}{\Delta^\omega} + \frac{d^{3n}}{\Delta^3} \right) = O \left( \frac{d^{3n}}{\Delta^3} \right) \text{ if } d \rightarrow \infty \text{ and } n \text{ is fixed;}$$

- $O\left(\frac{ne^{\omega n}d^{\omega n}}{\Delta^\omega} + \frac{nd^{3n}}{\Delta^3}\right) = O\left(\frac{nd^{3n}}{\Delta^3}\right)$  if  $d \rightarrow \infty$ ,  $n \rightarrow \infty$  and  $w_{\max}$  is fixed;

- 

$$O\left(\frac{nd^{\omega n}e^{\omega n}}{\Delta^\omega} + \frac{nd^{3n}}{\Delta^3}\right) = \begin{cases} O\left(\frac{nd^{3n}}{\Delta^3}\right) & \text{if } d \geq e^{\frac{\omega}{3-\omega}} w_{\max} \\ O\left(\frac{nd^{\omega n}e^{\omega n}}{\Delta^\omega}\right) & \text{else} \end{cases}$$

if  $d$  and  $w_{\max} = \max\{w_1, \dots, w_n\}$  are fixed and  $n \rightarrow \infty$ .

When the Bézout bound is reached i.e.  $\mathcal{D}_{\langle S \rangle} = \frac{d^n}{\Delta}$ , these complexities can be written as  $O\left(\frac{nd^{3n}}{\Delta^3}\right) = O\left(n\mathcal{D}_{\langle S \rangle}^3\right)$  and  $O\left(\frac{nd^{\omega n}e^{\omega n}}{\Delta^\omega}\right) = O\left(n\mathcal{D}_{\langle S \rangle}^\omega e^{\omega n}\right)$  where  $\mathcal{D}_{\langle S \rangle}$  denotes the degree of  $\langle S \rangle$ .

*Proof.* Thanks to the Bézout bound, the complexity of the change of ordering step can be bounded by  $O(n\frac{d^{3n}}{\Delta^3})$  whatever the parameter which tends to infinity.

The complexity of  $F_5$  algorithm is bounded by

$$O\left(\frac{n}{\Delta^\omega} \binom{d_{\text{reg}}(\langle S \rangle) + S_n}{n}^\omega\right).$$

Thanks to the Macaulay bound  $d_{\text{reg}}(\langle S \rangle) \leq nd - \sum_{i=1}^n w_i + w_{\max}$ . Hence,

$$O\left(\frac{n}{\Delta^\omega} \binom{d_{\text{reg}}(\langle S \rangle) + S_n}{n}^\omega\right) = O\left(\frac{n}{\Delta^\omega} \binom{nd + w_{\max}}{n}^\omega\right)$$

Assume  $n$  is fixed and  $d \rightarrow \infty$  one has

$$O\left(\binom{nd + w_{\max}}{n}\right) \underset{d \rightarrow \infty}{=} O((nd + w_{\max})^n).$$

We can assume that  $d \geq w_{\max}$  thus

$$O\left(\binom{nd + w_{\max}}{n}\right) \underset{d \rightarrow \infty}{=} O(d^n).$$

Assume now  $n \rightarrow \infty$ , using Stirling formula one has

$$O\left(\binom{nd + w_{\max}}{n}\right) \underset{n \rightarrow \infty}{=} O\left(\frac{(nd + w_{\max})^{nd + w_{\max}}}{n^n (n(d-1) + w_{\max})^{n(d-1) + w_{\max}}}\right). \quad (2.16)$$

If  $w_{\max}$  is fixed then equation 2.16 implies

$$\begin{aligned} O\left(\binom{nd + w_{\max}}{n}\right) &\underset{n \rightarrow \infty}{=} O\left(\frac{d^{nd + w_{\max}}}{(d-1)^{n(d-1) + w_{\max}}}\right) \\ &\underset{n \rightarrow \infty}{=} O\left(d^n \left(\frac{d}{d-1}\right)^{n(d-1) + w_{\max}}\right). \end{aligned}$$

Let  $f(x) = \left(\frac{x}{x-1}\right)^{x-1}$  one has  $\lim_{x \rightarrow \infty} f = e$ ,  $f(2) = 2$  and  $f$  is an increasing function. Hence, for any  $x \geq 2$  one has  $f(x) \leq e$ .

By consequence, if  $w_{\max}$  is fixed whether  $d$  be fixed or tends to infinity then equation (2.16) implies

$$O\left(\binom{nd + w_{\max}}{n}\right) =_{n \rightarrow \infty} O(d^n e^n).$$

It remains to compare the complexity of  $F_5$  and FGLM algorithms when  $n \rightarrow \infty$  and  $w_{\max}$  is fixed. In that case, the complexity of  $F_5$  is given by  $O\left(\frac{n}{\Delta^\omega} d^{\omega n} e^{\omega n}\right)$  and that of FGLM is given by  $O\left(\frac{n}{\Delta^3} d^{3n}\right)$  and we have

$$\begin{aligned} \frac{d^{\omega n} e^{\omega n}}{\Delta^\omega} \leq \frac{d^{3n}}{\Delta^3} &\iff e^{\omega n} \Delta^{3-\omega} \leq d^{(3-\omega)n} \\ &\iff e^\omega \Delta^{\frac{3-\omega}{n}} \leq d^{(3-\omega)} \\ &\iff e^{\frac{\omega}{3-\omega}} \Delta^{\frac{1}{n}} \leq d. \end{aligned}$$

Moreover,  $\Delta \leq w_{\max}^n$  thus  $e^{\frac{\omega}{3-\omega}} \Delta^{\frac{1}{n}} \leq e^{\frac{\omega}{3-\omega}} w_{\max}$ . If  $d \geq e^{\frac{\omega}{3-\omega}} w_{\max}$  then the complexity of FGLM dominates the complexity of  $F_5$ .  $\square$

## CHAPTER 3

# Solving structured polynomial systems

### Contents

|            |                                                                                        |           |
|------------|----------------------------------------------------------------------------------------|-----------|
| <b>3.1</b> | <b>Systems admitting a polynomial change of variables</b>                              | <b>67</b> |
| 3.1.1      | An algorithm for solving polynomial systems admitting a polynomial change of variables | 67        |
| 3.1.2      | Complexity of $F_5$ steps                                                              | 69        |
| 3.1.3      | Complexity of change of ordering steps                                                 | 71        |
| 3.1.4      | Comparison with the usual algorithm                                                    | 72        |
| <b>3.2</b> | <b>Application to polynomial systems invariant under a linear group</b>                | <b>72</b> |
| 3.2.1      | Preliminaries on invariant theory                                                      | 73        |
| 3.2.2      | Solving systems pointwise invariant under a pseudo-reflection group $\mathbb{G}$       | 74        |
| 3.2.3      | Particular case: some examples of groups in semi-direct product with $\mathfrak{S}_n$  | 77        |

*The results presented in this chapter are from a joint work with J.-C. Faugère, P. Gaudry and G. Renault*

In this chapter we are interested in solving polynomial systems with a particular structure. The structure that we investigate is polynomial systems admitting a polynomial change of variables. Note that efficient algorithms (polynomial in the numbers of variables) have been design to decide if a polynomial system admits a polynomial change of variables, see [GGR03, FP09] or [VZG90a, VZG90b] for the univariate case. It is well known that applying an invertible *linear* change of variables leaves the complexity of solving a polynomial system unchanged. More precisely, in [Laz83] it is shown that the degree of regularity does not change when applying a linear change of variables. Moreover, it is clear that in this case the number of solutions remains unchanged.

Assume that the polynomial system  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  can be written in terms of  $\vartheta_1, \dots, \vartheta_n \in \mathbb{K}[x_1, \dots, x_n]$ . That is to say there exist  $g_1, \dots, g_n \in \mathbb{K}[y_1, \dots, y_n]$  such that  $f_i = g_i(\vartheta_1, \dots, \vartheta_n)$  for  $i = 1, \dots, n$ . Usually to solve such a system instead of solving  $\mathcal{S}$  directly one solves the system  $\mathcal{S}' = \{g_1, \dots, g_n\}$ . Then, for each solution  $(v_1, \dots, v_n)$  of  $\mathcal{S}'$  one solves the system  $\{\vartheta_1 - v_1, \dots, \vartheta_n - v_n\}$ , see for instance [Stu08, DK02]. Although this method to solve  $\mathcal{S}$  seems to be more efficient since instead of solving one system we solve many *smaller* systems, in the best of our knowledge there is no known result about the complexity of solving  $\mathcal{S}$  in this way. Consequently, one cannot estimate theoretically the gain of this method in comparison of solving  $\mathcal{S}$  directly. The aim of this chapter is to provide a complexity estimate of such an algorithm to solve polynomial systems admitting a polynomial change of variables. By consequence, we provide an estimation of the gain of such a structure in the polynomial systems solving process.

Two important parameters for polynomial systems solving are the degree of the input equations and the regularity property. If  $\mathcal{S}$  is a regular system, there is *a priori* no reason that  $\mathcal{S}'$  is regular in the sense of Definitions 2.66 and 2.79. Moreover, in general one cannot predict the degree of the  $g_i$ 's given that of  $f_1, \dots, f_n$  and  $\vartheta_1, \dots, \vartheta_n$ .

In the first part of this chapter, we tackle these two issues for *regular* polynomial change of variables.

**Definition 3.1.** *Let  $\vartheta_1, \dots, \vartheta_n \in \mathbb{K}[x_1, \dots, x_n]$  be a polynomial change of variables. We say that  $\vartheta_1, \dots, \vartheta_n$  is a regular polynomial change of variables if  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent.*

In particular, we show that if we equip the ring  $\mathbb{K}[y_1, \dots, y_n]$  of a well chosen weighted degree then for *regular* polynomial change of variables, the degree and the regularity property is conserved. More precisely we get the following result.

**Theorem 3.2.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a regular polynomial system. Assume that  $\mathcal{S}$  admits a polynomial change of variables given by  $\vartheta_1, \dots, \vartheta_n$  and that  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent. Let  $g_1, \dots, g_n \in \mathbb{K}[y_1, \dots, y_n]$  satisfying  $f_i = g_i(\vartheta_1, \dots, \vartheta_n)$ . If the ring  $\mathbb{K}[y_1, \dots, y_n]$  is equipped with the weighted degree defined by the weights system  $(\deg(\vartheta_1), \dots, \deg(\vartheta_n))$  then  $\text{wdeg}(g_i) = \deg(f_i)$  and  $(g_1, \dots, g_n)$  is a regular sequence.*

Finally, if we assume that solving the systems  $\{\vartheta_1 - v_1, \dots, \vartheta_n - v_n\}$  for all solutions  $v$  of  $\mathcal{S}'$  is negligible in comparison of solving  $\{g_1, \dots, g_n\}$  then the recent results (Theorem 2.78) about solving quasi-homogeneous systems [FSV13] can be applied. Consequently, solving  $\mathcal{S}$  by solving  $\mathcal{S}'$  decreases by a factor of about  $(\prod_{i=1}^n \deg(\vartheta_i))^\omega$  the complexity in comparison of solving  $\mathcal{S}$  directly.

In the second part of this chapter, as an application we highlight a class of polynomial systems admitting *regular* polynomial change of variables. Let  $\mathbb{G} \subset \mathbf{GL}(\mathbb{K}, n)$  be a linear group. The set of polynomials that are invariant under the action of  $\mathbb{G}$  is called the invariant ring of  $\mathbb{G}$ . The Hilbert finiteness Theorem states that the invariant ring is finitely generated see for instance [Stu08, DK02]. Hence, if  $\mathcal{S}$  is in the invariant ring of  $\mathbb{G}$  then the usual strategy in invariant theory is to write  $\mathcal{S}$  in terms of the generators of the invariant ring of  $\mathbb{G}$ . Then, one solves the obtained system  $\mathcal{S}'$ . By studying the action of  $\mathbb{G}$  on the solutions of  $\mathcal{S}$ , it is well known that the degree of  $\langle \mathcal{S}' \rangle$  is divided by  $\#\mathbb{G}$  in comparison to the degree of  $\langle \mathcal{S} \rangle$ . Hence, using the symmetries of the system allows to divide by a factor  $(\#\mathbb{G})^3$  the complexity of the change of ordering step in the process of polynomial systems solving. However, depending on the group, the number of variables and equations in the system  $\mathcal{S}'$  can be greater than that of  $\mathcal{S}$ . Moreover, we do not know how the regularity property of  $\mathcal{S}$  is handed down to  $\mathcal{S}'$ . Hence, we cannot estimate the complexity of the total solving process of  $\mathcal{S}$  using the action of  $\mathbb{G}$ .

Nevertheless, thanks to the Shephard, Todd and Chevalley Theorem [ST54, Che55] we know exactly for which groups the generators of the invariant ring gives a *regular* polynomial change of variables. More precisely, if  $\mathbb{G}$  is a pseudo-reflective group then the invariant ring of  $\mathbb{G}$  is a polynomial ring generated by  $n$  algebraically independent homogeneous polynomials. Moreover, we now exactly the product of all the degrees of the generators of the invariant ring which is  $\#\mathbb{G}$ . Consequently, the first result allows us to conclude about the complexity of solving  $\mathcal{S}$  using the action of  $\mathbb{G}$ . More precisely, we get the following result.

**Corollary 3.3.** *Let  $\mathbb{G} \subset \mathbf{GL}(\mathbb{K}, n)$  be a pseudo-reflective group. Assume we can find efficiently all the elements in an orbit of  $\mathbb{G}$ . If  $\mathcal{S}$  is a regular polynomial system in the invariant*

ring of  $\mathbb{G}$  then solving it using the action of  $\mathbb{G}$  divides by a factor of  $(\#\mathbb{G})^\omega$  the complexity of solving  $\mathcal{S}$ .

Finally, we discuss how to find efficiently all the elements in an orbit of  $\mathbb{G}$  for some groups  $\mathbb{G}$  containing the symmetric group. Note that all the groups used in this thesis contain the symmetric group, see Chapter 6 and Chapter 7.

### 3.1 Systems admitting a polynomial change of variables

Let  $\mathcal{S} = \{f_1, \dots, f_s\}$  be a polynomial system in  $\mathbb{K}[x_1, \dots, x_n]$  admitting a polynomial change of variables given by  $\vartheta_1, \dots, \vartheta_n$ . The aim of this section is to evaluate the benefit of such a change of variables for the resolution of  $\mathcal{S}$ .

#### 3.1.1 An algorithm for solving polynomial systems admitting a polynomial change of variables

Let  $\phi$  be the map which describes the change of variables associated to  $\vartheta_1, \dots, \vartheta_n$ . More precisely,  $\phi$  is defined as follows.

**Definition 3.4.** Given  $\vartheta_1, \dots, \vartheta_n \subset \mathbb{K}[x_1, \dots, x_n]$ , the one to one map  $\phi$  is defined by

$$\begin{aligned} \phi^{-1} : \mathbb{K}[y_1, \dots, y_n] &\rightarrow \mathbb{K}[x_1, \dots, x_n] \\ f &\mapsto f(\vartheta_1, \dots, \vartheta_n). \end{aligned}$$

**Example 3.5.** Let  $f \in \mathbb{F}_2[x_1, x_2]$  defined by

$$\begin{aligned} f = & x_1^8 x_2^6 + x_1^6 x_2^8 + x_1^6 x_2^4 + x_1^5 x_2^9 + x_1^4 x_2^{10} + x_1^4 x_2^8 + x_1^4 x_2^6 + x_1^4 + x_1^3 x_2^{11} + \\ & x_1^3 x_2^7 + x_1^2 x_2^{12} + x_1^2 x_2^{10} + x_1^2 x_2^8 + x_1^2 x_2^6 + x_1 x_2^9 + x_2^{10} + x_2^8 + x_2^4 \end{aligned}$$

and  $\vartheta_1 = x_1^2 + x_2^2$  and  $\vartheta_2 = x_1 x_2^3 + x_2^2$ . The polynomial  $f$  can be expressed in terms of  $\vartheta_1$  and  $\vartheta_2$  as follows  $f = \vartheta_1^3 \vartheta_2^2 + \vartheta_1^2 + \vartheta_1 \vartheta_2^3$  which implies that

$$\phi(f) = y_1^3 y_2^2 + y_1^2 + y_1 y_2^3.$$

An important parameter in the polynomial system solving complexity is the degree of the equations. Hence, the first question that arises is Except for particular change of variables like those given by the elementary symmetric polynomial when the polynomial is symmetric, *a priori* we have no information about the degree of  $\phi(f)$ . However, for a well-chosen system of weights we can relate the degree of  $f$  to the weighted degree of  $\phi(f)$ .

**Lemma 3.6.** Let  $f \in \mathbb{K}[x_1, \dots, x_n]$  be a polynomial admitting a polynomial change of variables given by  $\vartheta_1, \dots, \vartheta_n$  each of degree  $w_i$ . Assume  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent. If the ring  $\mathbb{K}[y_1, \dots, y_n]$  is equipped with the weighted degree with weights  $(w_1, \dots, w_n)$  then  $\text{wdeg}(\phi(f)) = \deg(f)$  where  $w_i = \deg(\vartheta_i)$  for  $i = 1, \dots, n$ .

*Proof.* First, assume that  $\phi(f)$  is homogeneous w.r.t. the weights system  $(w_1, \dots, w_n)$ . That is to say,  $\phi(f) = \sum_{|\alpha|_w=\delta} c_\alpha y^\alpha$  where  $\delta = \text{wdeg}(\phi(f))$ . Let denote  $\vartheta_i - \vartheta_i^{(h)}$  by  $r_i$  for all  $i = 1, \dots, n$ . We have,

$$f = \phi(f)(\vartheta_1, \dots, \vartheta_n) = \sum_{|\alpha|_w=\delta} c_\alpha (\vartheta_1^{(h)} + r_1)^{\alpha_1} \dots (\vartheta_n^{(h)} + r_n)^{\alpha_n} = H + R,$$

where  $H = \sum_{|\alpha|_w=\delta} c_\alpha (\vartheta_1^{(h)})^{\alpha_1} \dots (\vartheta_n^{(h)})^{\alpha_n}$  and  $R = f - H$  with  $\deg(R) < \delta$ .

Since  $\sum_{|\alpha|_w=\delta} c_\alpha y^\alpha$  is a homogeneous polynomial and  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  too, we then have  $\deg(H) = \delta$  or  $H = 0$ . By hypothesis,  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent hence  $\deg(H) = \delta$ . Consequently,  $\deg(f) = \deg(H) = \text{wdeg}(\phi(f))$ . If  $\phi(f) = \sum_{i=0}^\delta f_i$  where  $f_i$  is a homogeneous polynomial of weighted degree  $i$ . We just have shown that  $\deg(f_i(\vartheta_1, \dots, \vartheta_n)) = i$  for  $i = 0, \dots, \delta$ . Hence,  $\deg(f) = \text{wdeg}(\phi(f))$ .  $\square$

**Example 3.7.** • *Continuing Example 3.5,  $\vartheta_1^{(h)} = x_1^2 + x_2^2$  and  $\vartheta_2^{(h)} = x_1 x_2^3$  are algebraically independent. The weights induced by  $\vartheta_1$  and  $\vartheta_2$  are  $(2, 4)$  and we have  $\deg(f) = 14 = \text{wdeg}(\phi(f))$ .*

- *Let  $f = x_1^{24} x_2^2 + x_1^8 x_2^{16} + x_1^8 + x_1^7 + x_1^4 x_2^2 + x_1^3 x_2 + x_2^{18} + x_2^8 + x_2^3 + x_2^2 \in \mathbb{F}_2[x_1, x_2]$ . Let  $\vartheta_1 = x_1^3 + x_2^2$  and  $\vartheta_2 = x_1^4 + x_2$ . Then,  $\vartheta_1^{(h)} = x_1^3$  and  $\vartheta_2^{(h)} = x_1^4$  are algebraically dependent. Moreover,  $\phi(f) = (y_1^4 + y_2^3)(y_1^4 y_2^2 + y_2^5) + y_1 y_2 + y_2^2 = y_1^8 y_2^2 + y_1 y_2 + y_2^8 + y_2^2$ . Since,  $\phi(f)^{(h)}$  is an annihilator polynomial of  $(\vartheta_1^{(h)}, \vartheta_2^{(h)})$  we have  $\deg(f) < \text{wdeg}(\phi(f))$ . More precisely,  $\deg(f) = 26$  and  $\text{wdeg}(\phi(f)) = 32$ .*

As mentionned in Section 2.4.1, a central property for polynomial systems solving is the regularity property. Indeed, to ensure that solving a system can be done efficiently and to obtain a sharp bound on the complexity of solving this system, the regularity property is required. If a regular polynomial system  $\{f_1, \dots, f_s\}$  admits a polynomial change of variables, *a priori* there is no reason that the system  $\{\phi(f_1), \dots, \phi(f_s)\}$  be regular in the sense of Definitions 2.66 and 2.79. Nevertheless, once again for a well-chosen system of weights we can show that the regularity property is conserved. More precisely, we have the following result.

**Theorem 3.8.** *Let  $\{f_1, \dots, f_s\}$  be a regular polynomial system such that each polynomial  $f_i$  can be expressed in terms of  $\vartheta_1, \dots, \vartheta_n$ . Assume that  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent. If the ring  $\mathbb{K}[y_1, \dots, y_n]$  is equipped with the weighted degree with weights  $(w_1, \dots, w_n)$  with  $w_i = \deg(f_i)$  then the system  $\{\phi(f_1), \dots, \phi(f_s)\} \subset \mathbb{K}[y_1, \dots, y_n]$  is regular.*

*Proof.* From the proof of Lemma 3.6 one has for any  $g = \sum_{i=0}^d (\sum_{|\alpha|_w=i} c_\alpha x^\alpha) \in \mathbb{K}[y_1, \dots, y_n]$

$$(\phi^{-1}(g))^{(h)} = \sum_{|\alpha|_w=d} c_\alpha (\vartheta_1^{(h)})^{\alpha_1} \dots (\vartheta_n^{(h)})^{\alpha_n} = g^{(h)}(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}). \quad (3.1)$$

Assume now that the sequence  $(\phi(f_1), \dots, \phi(f_s))$  is not regular *i.e.* there exists  $i \in \{2, \dots, s\}$  and  $0 \neq h, h_1, \dots, h_{i-1} \in \mathbb{K}[y_1, \dots, y_n]$  such that

$$h_1 \phi(f_1)^{(h)} + \dots + h_{i-1} \phi(f_{i-1})^{(h)} - h \phi(f_i)^{(h)} = 0.$$

Thus, from equation (3.1) this implies that

$$\sum_{j=1}^{i-1} h_j^{(h)}(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}) f_j^{(h)} - h^{(h)}(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}) f_i^{(h)} = 0.$$

Since  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent we have  $h^{(h)}(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}) \neq 0$ . By consequence,  $f_i^{(h)}$  is a divisor of 0 in  $\mathbb{K}[x_1, \dots, x_n] / \langle f_1^{(h)}, \dots, f_{i-1}^{(h)} \rangle$ . This yields a contradiction, hence the sequence  $(\phi(f_1), \dots, \phi(f_s))$  is regular.  $\square$

**Example 3.9.** Consider the following system of  $\mathbb{F}_2[x_1, x_2]$

$$S : \begin{cases} f_1 &= x_1^8 + x_1^5 x_2^3 + x_1^4 x_2^2 + x_1^2 x_2^6 + x_1 x_2^7 + x_2^8 + x_2^6 + x_2^4 \\ f_2 &= x_1^{14} + x_1^{12} x_2^2 + x_1^{10} x_2^4 + x_1^6 x_2^4 + x_1^4 x_2^6 + x_1^2 x_2^8 + x_2^{14} + x_2^{10} \end{cases}$$

admitting the polynomial change of variables given by  $\vartheta_1 = x_1^2 + x_2^2$  and  $\vartheta_2 = x_1 x_2^3 + x_2^2$ . We have  $f_1^{(h)} = x_1^8 + x_1^5 x_2^3 + x_1^2 x_2^6 + x_1 x_2^7 + x_2^8$  and  $f_2^{(h)} = x_1^{14} + x_1^{12} x_2^2 + x_1^{10} x_2^4 + x_2^{14}$ . Note that,  $(f_1^{(h)}, f_2^{(h)})$  is a regular sequence and  $(\vartheta_1^{(h)}, \vartheta_2^{(h)})$  are algebraically independent. The ring  $\mathbb{F}_2[y_1, y_2]$  is then equipped with the weights system  $(2, 4)$  and we have

$$\begin{aligned} \phi(f_1) = \phi(f_1)^{(h)} &= y_1^2 y_2 + y_1^4 + y_2^2 \\ \phi(f_2) = \phi(f_2)^{(h)} &= y_1^3 y_2^2 + y_1^7 \end{aligned}$$

One can check that  $(\phi(f_1), \phi(f_2))$  is a regular sequence and that  $\phi(f_i)(\vartheta_1^{(h)}, \vartheta_2^{(h)}) = f_i^{(h)}$  for  $i \in \{1, 2\}$ .

Following the previous result, we choose to call the polynomial change of variables given by  $\vartheta_1, \dots, \vartheta_n$  regular if  $\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}$  are algebraically independent since this property allows us to conserve the regularity property of the polynomial system. If  $\phi$  is a regular polynomial change of variables, since the regularity property is conserved when applying  $\phi$  instead of solving  $\mathcal{S} = \{f_1, \dots, f_n\}$  one can solve  $\mathcal{S}_\phi = \{\phi(f_1), \dots, \phi(f_n)\}$ . Note that solving  $\mathcal{S}_\phi$  is easier than solving  $\mathcal{S}$ . Indeed, from Lemma 3.6  $\deg(f_i) = \text{wdeg}(\phi(f_i))$  hence the degree of regularity of  $\mathcal{S}_\phi$  is upper bounded by the degree of regularity of  $\mathcal{S}$ , see Macaulay bounds of Corollary 2.76. Consequently, to solve  $\mathcal{S}_\phi$  we have to consider at worst the same Macaulay matrices as for solving  $\mathcal{S}$ . Moreover, Macaulay matrices are smaller when the systems of weights is not  $(1, \dots, 1)$ . Indeed, for any weights system it is clear the the number of monomials of weighted degree  $d$  is smaller or equal to the number of monomials of degree  $d$ .

Once the solutions of  $\mathcal{S}_\phi$  are found, to recover the solutions of  $\mathcal{S}$  we need to solve  $D_\phi$  systems of the form

$$\mathcal{S}_{\vartheta, v} \begin{cases} \vartheta_1(x_1, \dots, x_n) - v_1 = 0 \\ \vdots \\ \vartheta_n(x_1, \dots, x_n) - v_n = 0 \end{cases} \quad (3.2)$$

where  $v = (v_1, \dots, v_n)$  is a solution of  $\mathcal{S}_\phi$  and  $D_\phi$  is the number of solutions of  $\mathcal{S}_\phi$ . We summarize this algorithm to solve polynomial systems admitting regular polynomial change of variables in Algorithm 12.

We now investigate the complexity of Algorithm 12. First, we study the complexity of steps 3 and 8 involving  $F_5$  algorithm. Then, we study the complexity of steps 4 and 9 involving a change of ordering algorithm. Finally, we discuss about the complexity of step 1 and we conclude by a comparison of the complexity of Algorithm 12 with the complexity of solving directly  $\mathcal{S}$  by using the usual algorithm for polynomial systems solving, Algorithm 11.

### 3.1.2 Complexity of $F_5$ steps

If the input system  $\mathcal{S}$  is regular then from Theorem 3.8, the system  $\mathcal{S}_\phi$  is regular when the ring  $\mathbb{K}[y_1, \dots, y_n]$  is equipped with the weighted degree of weights system  $(w_1, \dots, w_n)$  where  $w_i = \deg(\vartheta_i)$ . By consequence, we get the following result.

---

**Algorithm 12:** Solving polynomial systems admitting polynomial change of variables.

---

**Input** : A polynomial system  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  admitting the regular change of variables defined by  $\vartheta_1, \dots, \vartheta_n$  with  $w_i = \deg(\vartheta_i)$  for  $i = 1, \dots, n$ .

**Output:** The solutions of  $\mathcal{S}$ .

- 1 Compute  $\mathcal{S}_\phi = \{\phi(f_1), \dots, \phi(f_n)\} \subset \mathbb{K}[y_1, \dots, y_n]$ ;
  - 2 Equip the ring  $\mathbb{K}[y_1, \dots, y_n]$  with the weighted degree of weights system  $(w_1, \dots, w_n)$ ;
  - 3 Compute the WDRL Gröbner basis of  $\langle \mathcal{S}_\phi \rangle$  using  $F_5$  algorithm;
  - 4 Compute  $\mathcal{G}_{>\text{lex}}$ , the LEX Gröbner basis of  $\langle \mathcal{S}_\phi \rangle$  using a change of ordering algorithm;
  - 5 From  $\mathcal{G}_{>\text{lex}}$  recover the solutions of  $\mathcal{S}_\phi$ ;
  - 6  $L := \emptyset$ ;
  - 7 **for** all the solutions  $v$  of  $\mathcal{S}_\phi$  **do**
  - 8     Compute the DRL Gröbner basis of  $\langle \mathcal{S}_{\vartheta,v} \rangle$  using  $F_5$  algorithm;
  - 9     Compute  $\mathcal{G}_{>\text{lex}}^{(v)}$ , the LEX Gröbner basis of  $\langle \mathcal{S}_{\vartheta,v} \rangle$  using a change of ordering algorithm;
  - 10    From  $\mathcal{G}_{>\text{lex}}^{(v)}$  recover the solutions of  $\mathcal{S}_{\vartheta,v}$  and add it to  $L$ ;
  - 11 **return**  $L$ ;
- 

**Proposition 3.10.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a regular polynomial system admitting the regular change of variables defined by  $(\vartheta_1, \dots, \vartheta_n)$ . Let denote  $\deg(f_i)$  by  $d_i$ . Computing the WDRL Gröbner basis of  $\mathcal{S}_\phi \subset \mathbb{K}[y_1, \dots, y_n]$  using the weights system  $(w_1, \dots, w_n)$  can be done in*

$$O\left(n \left( \frac{1}{\prod_{i=1}^n w_i} \left( \sum_{i=1}^n (d_i - w_i) + \max_{i=1, \dots, n} \{w_i\} + S_n \right) \right)^\omega\right) \quad (3.3)$$

arithmetic operations where  $S_i = S_{i-1} + w_i \frac{\gcd_{j=1, \dots, i-1} \{w_j\}}{\gcd_{j=1, \dots, i} \{w_j\}}$  for  $i \geq 2$ . If there exists  $i \in \{1, \dots, n\}$  such that  $w_i = 1$  then the complexity in equation (3.3) becomes

- $O\left(\frac{d^{\omega n}}{\Delta^\omega}\right)$  if  $d \rightarrow \infty$  and  $n$  is fixed;
- $O\left(\frac{ne^{\omega n} d^{\omega n}}{\Delta^\omega}\right)$  if  $n \rightarrow \infty$  and  $w_{\max}$  is fixed;

where  $d = \max_{i=1, \dots, n} \{d_i\}$  and  $w_{\max} = \max_{i=1, \dots, n} \{w_i\}$ .

*Proof.* From Theorem 3.8,  $\mathcal{S}_\phi$  is a regular polynomial system. Moreover, from Lemma 3.6 one has  $\text{wdeg}(\phi(f_i)) = \deg(f_i)$  for  $i = 1, \dots, n$ . The first result follows then from Theorem 2.83 and Theorem 2.78. The second part follows from Corollary 2.91.  $\square$

In order to study the complexity of step 8 we need to assume that  $(\vartheta_1, \dots, \vartheta_n)$  is a regular sequence. That is to say that  $(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)})$  is a regular sequence. Note that if  $(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)})$  is a regular sequence then  $\{\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)}\}$  are algebraically independent. The converse is not true, see [Smi95, Theorem 6.2.1]. Note that in general, solving all the systems  $\mathcal{S}_{\vartheta,v}$  are negligible in comparison to solving  $\mathcal{S}_\phi$ . Furthermore, for particular cases there exist very efficient method to solve these systems without using Gröbner bases computations. See for instance Section 3.2.3.

**Proposition 3.11.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a regular polynomial system admitting the change of variables defined by  $(\vartheta_1, \dots, \vartheta_n)$ . Assume that  $(\vartheta_1^{(h)}, \dots, \vartheta_n^{(h)})$  is a regular sequence. Let  $w_i = \deg(\vartheta_i)$  and  $d_i = \deg(f_i)$  for  $i = 1, \dots, n$ . Computing the DRL Gröbner basis of  $\langle \mathcal{S}_{\vartheta, v} \rangle$  for any solution  $v$  of  $\mathcal{S}_\phi$  (i.e. all steps 8 of Algorithm 12) can be done in*

- $O\left(\frac{d^n w_{\max}^{n\omega}}{\prod_{i=1}^n w_i}\right)$  if  $n$  is fixed and  $w_{\max} \rightarrow \infty$ ;
- $O\left(\frac{n d^n w_{\max}^{n\omega} e^{n\omega}}{\prod_{i=1}^n w_i}\right)$  if  $n \rightarrow \infty$ ;

where  $d = \max\{d_1, \dots, d_n\}$  and  $w_{\max} = \max\{w_1, \dots, w_n\}$ .

*Proof.* From Bézout's bound (see Corollary 2.76) the number of solutions of  $\mathcal{S}_\phi$  is bounded by  $\frac{\prod_{i=1}^n d_i}{\prod_{i=1}^n w_i}$ . From Macaulay's bound (see Corollary 2.76) and Theorem 2.83 and Theorem 2.77 the complexity of solving one system  $\mathcal{S}_{\vartheta, v}$  is bounded by  $O\left(n^{(1+\sum_{i=1}^n w_i)\omega}\right)$ . The result follows from Corollary 2.91.  $\square$

### 3.1.3 Complexity of change of ordering steps

The complexity of change of ordering steps follows directly from Lemma 3.6 and Bezout's bound which allows us to bound the degree of  $\mathcal{S}_\phi$  and  $\mathcal{S}_{\vartheta, v}$  for any solutions  $v$  of  $\mathcal{S}_\phi$ .

**Proposition 3.12.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a polynomial system admitting the regular polynomial change of variables defined by  $(\vartheta_1, \dots, \vartheta_n)$ . Let  $d_i = \deg(f_i)$  and  $w_i = \deg(w_i)$  for  $i = 1, \dots, n$ . The complexity of computing the LEX Gröbner basis of  $\langle \mathcal{S}_\phi \rangle$  given its WDRL Gröbner basis w.r.t. the weights system  $(w_1, \dots, w_n)$  (i.e. step 4 of Algorithm 12) can be bounded by  $O\left(n\left(\frac{\prod_{i=1}^n d_i}{\prod_{i=1}^n w_i}\right)^3\right)$  arithmetic operations.*

*Proof.* From Lemma 3.6, if the ring  $\mathbb{K}[y_1, \dots, y_n]$  is equipped with the weighted degree defined by the weights system  $(w_1, \dots, w_n)$  then  $\text{wdeg}(\phi(f_i)) = d_i$  for  $i = 1, \dots, n$ . Then, the weighted Bézout bound and Theorem 2.85 allows us to conclude.  $\square$

In the same way we can bound the complexity of computing all the LEX Gröbner basis of  $\langle \mathcal{S}_{\vartheta, v} \rangle$ .

**Proposition 3.13.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a polynomial system admitting the regular polynomial change of variables defined by  $(\vartheta_1, \dots, \vartheta_n)$ . Let  $d_i = \deg(f_i)$  and  $w_i = \deg(w_i)$  for  $i = 1, \dots, n$ . The complexity of computing the LEX Gröbner basis of  $\langle \mathcal{S}_{\vartheta, v} \rangle$  for any solution  $v$  of  $\mathcal{S}_\phi$  given their DRL Gröbner bases (i.e. steps 9 of Algorithm 12) can be bounded by*

$$O\left(n\left(\prod_{i=1}^n d_i\right)\left(\prod_{i=1}^n w_i\right)^2\right)$$

arithmetic operations.

### 3.1.4 Comparison with the usual algorithm

As mentioned in Chapter 2 Section 2.4.3, the complexity of computing the solutions of a system given its LEX Gröbner basis depends on the field  $\mathbb{K}$ . Moreover, in general it is negligible in comparison to compute the LEX Gröbner basis. Consequently, to compare Algorithm 12 with the usual algorithm to solve polynomial systems we compare only the cost of  $F_5$  steps and change of ordering steps.

In the two previous sections for completeness, we gave the complexity of computing the LEX Gröbner basis of  $\mathcal{S}_{\vartheta,v}$  for all solutions  $v$  of  $\mathcal{S}_\phi$ . However, if the  $w_i$ 's are sufficiently small in comparison to the  $d_i$ 's solving the systems  $\mathcal{S}_{\vartheta,v}$  is negligible in comparison of solving  $\mathcal{S}_\phi$ . In particular, in this thesis all the considered polynomial change of variables have a very efficient way to solve these systems (see Section 3.2.3).

From the Bézout bound the degree of  $\langle \mathcal{S} \rangle$  is bounded by  $\prod_{i=1}^n d_i \leq d^n$  where  $d = \max_{i=1,\dots,n} \{d_i = \deg(f_i)\}$ . From the Macaulay bound,  $d_{\text{reg}}(\langle \mathcal{S} \rangle) \leq \sum_{i=1}^n (d_i - 1) + 1 \leq nd + 1$ . Consequently from Theorems 2.77, 2.83 and 2.85 the complexity of solving directly  $\mathcal{S}$  is bounded by  $O(d^{\omega n} + d^{3n})$  (respectively  $O(ne^{\omega n} d^{\omega n} + nd^{3n})$ ) arithmetic operations if  $n$  is fixed (respectively  $n \rightarrow \infty$ ). Hence, given  $\mathcal{S}_\phi$  the complexity of computing the LEX Gröbner basis of  $\mathcal{S}_\phi$  is divided by  $(\prod_{i=1}^n w_i)^\omega$  for the  $F_5$  step and  $(\prod_{i=1}^n w_i)^3$  for the change of ordering step.

One issue remains, what is the complexity of computing  $\mathcal{S}_\phi$  given  $\phi$  and  $\mathcal{S}$ ? For particular changes of variables as the one given by the elementary symmetric polynomials there exists a very efficient algorithm to compute  $\mathcal{S}_\phi$  given  $\mathcal{S}$  (see for instance [Stu08]). More generally, as mentioned in Section 2.1.4, computing  $\mathcal{S}_\phi$  can be done using Gröbner bases and elimination order by using for instance Algorithm 2 or Algorithm 3. Unfortunately, there is no general tight bound on the complexity of performing a change of variables using Gröbner bases. However, in [FP09] the authors proposed an efficient algorithm for computing  $\mathcal{S}_\phi$  when  $f_1, \dots, f_n$  and  $\vartheta_1, \dots, \vartheta_n$  are homogeneous polynomials. More precisely, when the degrees of  $f_1, \dots, f_n$  and  $\vartheta_1, \dots, \vartheta_n$  are fixed (*i.e.* do not depend on  $n$ ) their algorithm has a polynomial complexity in  $n$ . In that case, the complexity of computing  $\mathcal{S}_\phi$  is then negligible in comparison to solve  $\mathcal{S}_\phi$ . Moreover, in the whole of this thesis (in particular in Chapter 6 and Chapter 7) there is an efficient way to compute  $\mathcal{S}_\phi$  which becomes negligible. The next result summarizes the gain of solving  $\mathcal{S}$  using Algorithm 12 in comparison of using Algorithm 11.

**Theorem 3.14.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a regular polynomial system admitting a regular change of variables defined by  $\vartheta_1, \dots, \vartheta_n$ . Let  $w_i = \deg(\vartheta_i)$  for  $i = 1, \dots, n$ . Assume  $\mathcal{S}_\phi$  can be computed efficiently. If the  $w_i$ 's are sufficiently small or if there exists an efficient algorithm to solve the systems  $\mathcal{S}_{\vartheta,v}$  for any solution  $v$  of  $\mathcal{S}_\phi$ , then the complexity of solving  $\mathcal{S}$  using Algorithm 12 is divided by  $(\prod_{i=1}^n w_i)^\omega$  in comparison of solving  $\mathcal{S}$  using Algorithm 11.*

In the next section, we study the impact of such a result on the complexity of solving polynomial systems having some symmetries that is to say polynomial systems invariant under the action of a linear group.

## 3.2 Application to polynomial systems invariant under a linear group

In this Section, we consider the action of a finite linear group  $\mathbb{G}$  on polynomials. First, we need some background about invariant theory. For a more thorough reading on this subject

see for instance [Stu08, CLO07].

### 3.2.1 Preliminaries on invariant theory

We assume that the field  $\mathbb{K}$  has characteristic zero or has a positive “large enough characteristic” that is to say not dividing the cardinality of  $\mathbb{G}$ . All notions of invariant theory recalled in this section, can be generalized to an affine variety instead of the affine space.

The linear group  $\mathbb{G} \subset \mathbf{GL}(\mathbb{K}, n)$  naturally acts on the affine space  $\mathbb{A}^n$  or any  $\mathbb{K}$ -vector space of dimension  $n$  by the matrix vector multiplication. This action can be translated to polynomial rings. More precisely we have the following definition.

**Definition 3.15** (Invariant rings). *Let  $\mathbb{K}[x_1, \dots, x_n]$  be a polynomial ring in  $n$  variables with coefficients in  $\mathbb{K}$ . The action of a group  $\mathbb{G} \subset \mathbf{GL}(\mathbb{K}, n)$  on  $\mathbb{K}[x_1, \dots, x_n]$  is defined by*

$$\begin{array}{ccc} \mathbb{G} \times \mathbb{K}[x_1, \dots, x_n] & \longrightarrow & \mathbb{K}[x_1, \dots, x_n] \\ g, f & \longmapsto & g \cdot f \end{array}$$

where  $g \cdot f$  is defined by  $(g \cdot f)(v) = f(g^{-1} \cdot v)$  where  $v$  is the vector  $(x_1, \dots, x_n)$ . This definition uses the inverse of  $g$  in order to get a left action. The invariant ring of  $\mathbb{G}$  is the set of all invariant polynomials in  $\mathbb{K}[x_1, \dots, x_n]$  :

$$\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}} = \{f \in \mathbb{K}[x_1, \dots, x_n] \mid g \cdot f = f \text{ for all } g \in \mathbb{G}\}.$$

One of the fundamental results in invariant theory was proven by Hilbert in the last decade of the nineteenth century and is summarized in the following theorem.

**Theorem 3.16** (Hilbert’s finiteness theorem). *The invariant ring of  $\mathbb{G}$  is finitely generated.*

Following this theorem, many results were provided for the decomposition of invariant rings. In particular, it is proven that  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  is a finitely generated free module over  $\mathbb{K}[\theta_1, \dots, \theta_n]$  where  $\theta_1, \dots, \theta_n$  are algebraically independent homogeneous polynomials. Consequently there exist homogeneous polynomials  $\eta_1, \dots, \eta_t \in \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  such that

$$\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}} = \bigoplus_{i=1}^t \eta_i \mathbb{K}[\theta_1, \dots, \theta_n]. \quad (3.4)$$

The decomposition (3.4) is called a Hironaka decomposition of  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$ . The polynomials  $\theta_1, \dots, \theta_n$  (resp.  $\eta_1, \dots, \eta_t$ ) are the *primary invariants* (resp. *secondary invariants*) of  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  and satisfy  $t = \frac{\prod_{i=1}^n \deg(\theta_i)}{\#\mathbb{G}}$ .

To solve pointwise invariant polynomial systems (*i.e.* each polynomial in the system is in the invariant ring of the corresponding group) by using the symmetries, one has to rewrite the systems in terms of the primary and secondary invariants. If the invariant ring of  $\mathbb{G}$  is not a polynomial algebra – *i.e.* the secondary invariants are not reduced to  $\{1\}$  – considering the symmetries can complicate the resolution of the system. Actually, since secondary invariants are not independent, then considering the symmetries when these invariants are not trivial increases the number of equations and variables to consider. Consequently, the polynomial systems may be more difficult to solve. Moreover, computing a Hironaka decomposition can be a difficult task. Solving polynomial systems invariant under a non pseudo-reflective group

has been studied by Colin [Col97] and can also be tackled by using SAGBI Gröbner bases, see for instance [FR09]; we will not need this strategy.

By consequence an elementary question is to know under which conditions on  $\mathbb{G}$ , its invariant ring is a graded polynomial algebra (and thus when the set of secondary invariants is trivial). The answer is given in the following theorem.

**Theorem 3.17** (Shephard, Todd, Chevalley [Che55, ST54]). *The invariant ring of  $\mathbb{G}$  is a polynomial algebra if and only if  $\mathbb{G}$  is a pseudo-reflection group.*

A pseudo-reflection is a linear automorphism of  $\mathbb{A}^n$  that is not the identity map, but leaves a hyperplane  $H \subset \mathbb{A}^n$  pointwise invariant. The group  $\mathbb{G} \subset \mathbf{GL}(\mathbb{K}, n)$  is said to be a pseudo-reflection group if it is generated by its pseudo-reflections.

**Example 3.18.** *Coxeter groups can be represented thanks to a pseudo reflection group. In particular, the dihedral Coxeter group  $D_n = (\mathbb{Z}/2\mathbb{Z})^{n-1} \rtimes \mathfrak{S}_n$  can be represented by the action on  $\mathbb{A}^n$  defined by the rule that  $\mathfrak{S}_n$  permutes the coordinates of the vectors, whereas  $(\mathbb{Z}/2\mathbb{Z})^{n-1}$  changes the sign on an even number of its coordinates. From Theorem 3.17 the invariant ring of  $D_n$  is then a polynomial algebra. In the sequel, the dihedral Coxeter group  $D_n$  will always correspond to this representation. It is a well known group and its invariant ring too. Actually,*

$$\mathbb{K}[x_1, \dots, x_n]^{D_n} = \mathbb{K}[p_2, \dots, p_{2(n-1)}, p_n] = \mathbb{K}[s_1, \dots, s_{n-1}, e_n]$$

where  $p_i = \sum_{k=1}^n x_k^i$  is the  $i^{\text{th}}$  power sum,  $s_i = \sum_{1 \leq k_1 < \dots < k_i \leq n} \prod_{j=1}^i x_{k_j}^2$  is the  $i^{\text{th}}$  elementary symmetric polynomial in terms of  $x_1^2, \dots, x_n^2$  and  $e_n = \prod_{k=1}^n x_k$  is the  $n^{\text{th}}$  elementary symmetric polynomial in terms of  $x_1, \dots, x_n$ .

In the case where  $\mathbb{G}$  is a pseudo-reflection group, Theorem 3.17 allows us to construct an isomorphism  $\Omega_{\mathbb{G}}$  between  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  and  $\mathbb{K}[y_1, \dots, y_n]$  where  $y_1, \dots, y_n$  are new indeterminates.

**Definition 3.19.** *Let  $\mathbb{G}$  be a pseudo-reflective group and  $\theta_1, \dots, \theta_n \in \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  be the primary invariants of  $\mathbb{G}$ . We denote by  $\Omega_{\mathbb{G}}$  the ring isomorphism from  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  to  $\mathbb{K}[y_1, \dots, y_n]$  corresponding to the change of coordinates by the  $\theta_i$ 's and defined by*

$$\begin{aligned} \Omega_{\mathbb{G}}^{-1} : \mathbb{K}[y_1, \dots, y_n] &\longrightarrow \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}} \\ f &\longmapsto f(\theta_1, \dots, \theta_n). \end{aligned}$$

In the following, we denote by  $\mathbb{K}[\theta_1, \dots, \theta_n]$  the polynomial ring given by the image of  $\Omega_{\mathbb{G}}$ .

### 3.2.2 Solving systems pointwise invariant under a pseudo-reflection group $\mathbb{G}$

In the case where the group  $\mathbb{G}$  is pseudo reflective, the invariant ring is a polynomial ring. Hence, the isomorphism  $\Omega_{\mathbb{G}}$  defines a change of variables on any polynomial in  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$ . Thus, we have highlighted a class of polynomial systems admitting polynomial change of variables. Indeed, let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a polynomial system in  $\mathbb{K}[x_1, \dots, x_n]$ . If each of the  $f_i$ 's is in  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  i.e.  $\mathcal{S}$  is pointwise invariant under  $\mathbb{G}$ , then  $\mathcal{S}$  admits the polynomial

change of variables  $\Omega_{\mathbb{G}}$  defined by  $\theta_1, \dots, \theta_n$ , the primary invariant of  $\mathbb{G}$ . Moreover, since the primary invariants of  $\mathbb{G}$  are homogeneous and algebraically independent,  $\Omega_{\mathbb{G}}$  is a regular change of variables as defined in Section 3.1.1. Consequently, the complexity of solving a polynomial systems pointwise invariant under the action of a pseudo reflective group is a direct consequence of results in Section 3.1.

**Remark 3.20.** *Since  $\mathcal{S}$  is invariant under the action of  $\mathbb{G}$ , so is its variety. The usual strategy in invariant theory (see [Stu08]) to solve  $\mathcal{S}$  is to look for an ideal having for variety the set of  $\mathbb{G}$ -orbits of the variety of  $\mathcal{S}$ . In that way, the number of solutions is decreased and by consequence the complexity of the change of ordering step is also decreased.*

Let us describe how the ideal mentioned in the previous remark can be computed in order to compare this method with the one proposed in the previous sections. If  $V$  denotes the variety of  $\mathcal{S}$ , the set of  $\mathbb{G}$ -orbits of  $V$  is defined as the quotient  $V/\mathbb{G}$  defined by the equivalence relation  $\sim$  satisfying for all  $v_1, v_2$  in  $V$ ,  $v_1 \sim v_2$  if there exists  $\sigma \in \mathbb{G}$  such that  $v_1 = \sigma \cdot v_2$ . In order to find an ideal having for variety  $V/\mathbb{G}$  the usual method in invariant theory, is to express the system  $\mathcal{S}$  in terms of the primary invariants. That is to say one computes and solves the system  $\mathcal{S}_{\mathbb{G}} = \{\Omega_{\mathbb{G}}(f_1), \dots, \Omega_{\mathbb{G}}(f_n)\} \subset \mathbb{K}[\theta_1, \dots, \theta_n]$  having for variety  $V/\mathbb{G}$ . Note that this is exactly what we done in Section 3.1. Moreover, in that case the solutions of  $\mathcal{S}_{\mathbb{G}}$  are the orbits of  $V$  under the action of  $\mathbb{G}$ . Hence, if  $v$  is an orbit of  $V/\mathbb{G}$  then solving the system  $\mathcal{S}_{\theta, v}$  defined in equation (3.2) corresponds to find all the elements in  $V$  that are in the orbit  $v$ .

The class formula allows us to conclude that the degree of  $\langle \mathcal{S}_{\mathbb{G}} \rangle$  is divided by  $\#\mathbb{G}$  in comparison to the degree of  $\langle \mathcal{S} \rangle$ . Hence, the gain on change of ordering step for solving  $\mathcal{S}_{\mathbb{G}}$  instead of  $\mathcal{S}$  is about  $(\#\mathbb{G})^3$ . However, although it is usually admitted that solving  $\mathcal{S}_{\mathbb{G}}$  is more efficient than solving  $\mathcal{S}$ , in our knowledge there is no known result about the complexity of the  $F_5$  step in the solving of  $\mathcal{S}_{\mathbb{G}}$ . Nevertheless, using results of Section 3.1 we show that the overall complexity of solving  $\mathcal{S}_{\mathbb{G}}$  is divided by  $(\#\mathbb{G})^w$  in comparison of solving  $\mathcal{S}$ .

**Remark 3.21.** *The key point of our method is to equip the ring  $\mathbb{K}[\theta_1, \dots, \theta_n]$  of the weighted degree with weights system induced by the degree of the primary invariants. By this way, we conserve the regularity property. Then, the recent results about the complexity of solving quasi-homogeneous systems (see [FSV13]) allows us to estimate the gain of solving  $\mathcal{S}_{\mathbb{G}}$  instead of  $\mathcal{S}$ .*

*The idea of equipping the variables representing the invariants (primary and secondary) with a weight corresponding to the degree of the invariant was already mentioned in [GG99] but in the context of computing the relations between the secondary invariants. Indeed, assume the group  $\mathbb{G}$  is not a pseudo-reflective group then its set of secondary invariants is not reduced to  $\{1\}$ . Since the secondary invariants are not algebraically independent a fundamental issue of computational invariant theory is to find the relations between these invariants. For this purpose, it is possible to compute the Gröbner basis of*

$$\mathcal{I} = \langle \theta_1 - y_1, \dots, \theta_n - y_n, \eta_1 - y_{n+1}, \dots, \eta_t - y_{n+t} \rangle \subset \mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_{n+t}]$$

*w.r.t. an elimination order w.r.t. the set of variables  $\{y_1, \dots, y_{n+t}\}$  that is to say we eliminate the variables  $x_1, \dots, x_n$ . In Chapter 2, we have seen that computing a Gröbner basis of homogeneous ideals is easier than computing a Gröbner basis of an affine ideal. Since, the primary and secondary invariants are homogeneous polynomials, in order to ensure that  $\mathcal{I}$  is an homogeneous ideal one needs to equip the ring  $\mathbb{K}[x_1, \dots, x_n, y_1, \dots, y_{n+t}]$  with the weighted degree with weights system  $(1, \dots, 1, \deg(\theta_1), \dots, \deg(\theta_n), \deg(\eta_1), \dots, \deg(\eta_t))$ .*

**Theorem 3.22.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a regular polynomial system. Let  $\mathbb{G}$  be a pseudo-reflection group. Assume we know an efficient way to compute all the elements in an orbit of  $\mathbb{G}$ . If  $\mathcal{S} \subset \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}}$  then the complexity of solving  $\mathcal{S}$  is divided by  $(\#\mathbb{G})^\omega$  by considering the action of  $\mathbb{G}$ .*

*Proof.* From Theorem 3.17 and equation (3.4) we have that  $f_1, \dots, f_n$  can be written in terms of  $\theta_1, \dots, \theta_l$  be the primary invariants of  $\mathbb{G}$  (i.e. algebraically independent homogeneous polynomials). That is to say  $\mathcal{S}$  admits the regular polynomial change of variables  $\Omega_{\mathbb{G}}$ . Hence, from equation (3.4) we have  $\prod_{i=1}^n \deg(\theta_i) = \#\mathbb{G}$ . Finally, applying Theorem 3.14 concludes the proof.  $\square$

Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be two pseudo-reflection groups such that  $\mathbb{G}_1 \subset \mathbb{G}_2$ . Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_2} \subset \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_1}$ , we denote  $\mathcal{S}_{\mathbb{G}_1}$  the system  $\{\Omega_{\mathbb{G}_1}(f_1), \dots, \Omega_{\mathbb{G}_1}(f_n)\}$  and  $\mathcal{S}_{\mathbb{G}_2}$  the system  $\{\Omega_{\mathbb{G}_2}(f_1), \dots, \Omega_{\mathbb{G}_2}(f_n)\}$ .

**Corollary 3.23.** *If  $\mathcal{S}$  is a regular system then solving  $\mathcal{S}_{\mathbb{G}_2}$  instead of  $\mathcal{S}_{\mathbb{G}_1}$  allows to divide by  $\left(\frac{|\mathbb{G}_2|}{|\mathbb{G}_1|}\right)^\omega = (\mathbb{G}_2 : \mathbb{G}_1)^\omega$  the complexity of solving the polynomial system where  $(\mathbb{G}_2 : \mathbb{G}_1)$  denotes the index of  $\mathbb{G}_1$  in  $\mathbb{G}_2$ .*

In some applications as for instance in Chapter 6 and 7 one can be interested in the estimation of the speed up provided by the solving of  $\mathcal{S}_{\mathbb{G}_2}$  instead of  $\mathcal{S}_{\mathbb{G}_1}$  when the system  $\mathcal{S}$  is not regular. Indeed, sometimes the symmetries due to the action of the group  $\mathbb{G}_1$  are so natural that the right modeling of the problem as a polynomial system is to take into account the symmetries of  $\mathbb{G}_1$  i.e. considering the system  $\mathcal{S}_{\mathbb{G}_1}$  but using the usual degree and not the weighted degree induces by the primary invariants of  $\mathbb{G}_1$ . It is what happens in Chapter 6 and Chapter 7. In that case, we cannot apply directly the results of Section 3.1.

By consequence, in the case where  $\mathcal{S}$  is not regular but  $\mathcal{S}_{\mathbb{G}_1}$  is regular when using the usual degree, to estimate the speed up provided by the solving of  $\mathcal{S}_{\mathbb{G}_2} = \{g_1, \dots, g_n\} \subset \mathbb{K}[y_1, \dots, y_n]$  instead of  $\mathcal{S}_{\mathbb{G}_1} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  we need to highlight a polynomial change of variables  $p_1, \dots, p_n \in \mathbb{K}[x_1, \dots, x_n]$  such that  $f_i = g_i(p_1, \dots, p_n)$  for  $i = 1, \dots, n$ . In the following, we denote by  $\phi$  the map describing this change of variables defined as:

$$\begin{aligned} \phi^{-1} : \mathbb{K}[y_1, \dots, y_n] &\rightarrow \mathbb{K}[x_1, \dots, x_n] \\ f &\mapsto f(p_1, \dots, p_n). \end{aligned}$$

**Lemma 3.24.** *Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be two pseudo-reflective groups such that  $\mathbb{G}_1 \subset \mathbb{G}_2$ . Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a polynomial system in  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_2} \subset \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_1}$ . We denote by  $\mathcal{S}_{\mathbb{G}_1}$  the system  $\{\Omega_{\mathbb{G}_1}(f_1), \dots, \Omega_{\mathbb{G}_1}(f_n)\}$ . The change of variables  $p_1, \dots, p_n$  to write  $\mathcal{S}_{\mathbb{G}_1}$  in terms of the primary invariants of  $\mathbb{G}_2$  always exists.*

*Proof.* Since  $\mathbb{G}_1 \subset \mathbb{G}_2$ , we have  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_2} \subset \mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_1}$ . In particular, the primary invariants  $\vartheta_1, \dots, \vartheta_n$  of  $\mathbb{G}_2$  are in  $\mathbb{K}[x_1, \dots, x_n]^{\mathbb{G}_1}$ . By consequence, let  $\theta_1, \dots, \theta_n$  be the primary invariants of  $\mathbb{G}_1$  for  $i \in \{1, \dots, n\}$  there exists  $p_i \in \mathbb{K}[x_1, \dots, x_n]$  such that  $\vartheta_i = p_i(\theta_1, \dots, \theta_n)$  which concludes the proof.  $\square$

The next result is a direct consequence of Lemma 3.24 and Theorem 3.14.

**Corollary 3.25.** *Assume the ring  $\mathbb{K}[x_1, \dots, x_n]$  is equipped with the usual degree. If  $\mathcal{S}_{\mathbb{G}_1} \subset \mathbb{K}[x_1, \dots, x_n]$  is regular and  $p_1^{(h)}, \dots, p_n^{(h)}$  are algebraically independent then the complexity of solving  $\mathcal{S}_{\mathbb{G}_2}$  is divided by  $(\prod_{i=1}^n \deg(p_i))^\omega$  in comparison to the complexity of solving  $\mathcal{S}_{\mathbb{G}_1}$ .*

We conclude this chapter by giving some examples of linear groups  $\mathbb{G}$  encountered in applications and for which the step of finding the solutions of  $\mathcal{S}$  given that of  $\mathcal{S}_{\mathbb{G}}$  can be done very efficiently *i.e.* by solving univariate polynomials.

### 3.2.3 Particular case: some examples of groups in semi-direct product with $\mathfrak{S}_n$

Let  $\mathbb{G} = \mathfrak{S}_n$  be the Symmetric group. It is well-known that the invariant ring of  $\mathfrak{S}_n$  is generated by the elementary symmetric polynomials, see for instance [Stu08]. Given a solution  $v = (v_1, \dots, v_n)$  of  $\mathcal{S}_{\mathbb{G}}$ , to recover the solutions of  $\mathcal{S}$  we then need to solve polynomial systems of the form

$$\begin{cases} e_1(x_1, \dots, x_n) - v_1 = 0 \\ \vdots \\ e_n(x_1, \dots, x_n) - v_n = 0 \end{cases} \quad (3.5)$$

where  $e_i$  is the  $i$ th elementary symmetric polynomial. However, it is well-known that solving the system (3.5) is equivalent to solve a univariate polynomial. Indeed, we have

$$\prod_{i=1}^n (x - x_i) = x^n + \sum_{i=1}^n (-1)^i e_i(x_1, \dots, x_n) x^{n-i}. \quad (3.6)$$

Moreover, solving a univariate polynomial of the form of (3.6) can be done in quasi-linear time in  $n$  see [VZGG03, Pan02] or Section 2.4.3. By consequence, in that case solving the systems  $\mathcal{S}_{\partial, v}$  for all solutions  $v$  of  $\mathcal{S}_{\mathfrak{S}_n}$  is negligible in comparison of solving  $\mathcal{S}$ . Moreover, it is well known that writing  $\mathcal{S}$  in terms of the elementary symmetric polynomial can be done very efficiently without Gröbner bases computations, see for instance [Stu08]. The result of Theorem 3.14 can thus be applied in the case of symmetric polynomial systems.

Let  $\mathbb{G} = (\mathbb{Z}/2\mathbb{Z})^n \rtimes \mathfrak{S}_n$ , this group can be represented thanks to a pseudo-reflection group. Indeed, it can be represented by the action on  $\mathbb{A}^n$  defined by the rule that  $\mathfrak{S}_n$  permutes a chosen vector, whereas  $(\mathbb{Z}/2\mathbb{Z})^n$  changes the sign of some vector elements. The invariant ring of  $\mathbb{G}$  is generated by the elementary symmetric polynomial in terms of  $x_1^2, \dots, x_n^2$  denoted  $s_1, \dots, s_n$ , see for instance [Kan01]. By consequence, in the same way as for the Symmetric group, given a solution of  $\mathcal{S}_{\mathbb{G}}$  we can compute the corresponding solutions of  $\mathcal{S}$  by solving a univariate polynomial and computing  $n$  square roots. Indeed, we have

$$\prod_{i=1}^n (x - x_i^2) = x^n + \sum_{i=1}^n (-1)^i s_i(x_1, \dots, x_n) x^{n-i}. \quad (3.7)$$

Another group which can be similarly handled is  $\mathbb{G} = (\mathbb{Z}/2\mathbb{Z})^{n-1} \rtimes \mathfrak{S}_n$ . It can be represented thanks to a pseudo-reflection group by the action on  $\mathbb{A}^n$  which is the same as  $(\mathbb{Z}/2\mathbb{Z})^n \rtimes \mathfrak{S}_n$  except that  $(\mathbb{Z}/2\mathbb{Z})^{n-1}$  changes the sign of an even number of vector elements. The invariant ring of  $\mathbb{G}$  is generated by  $s_1, \dots, s_{n-1}, e_n$ , see for instance [Kan01]. Since, we have  $s_n = e_n^2$  to recover the solutions of  $\mathcal{S}_{\mathbb{G}}$  we can solve the polynomial in equation (3.7) and then computing  $n$  square roots.

**Example 3.26.** Let  $f_1, f_2 \in \mathbb{F}_{53}[x_1, x_2]$  defined by

$$\begin{aligned} f_1 &= 33x_1^2x_2^2 + 27x_1^2x_2 + 27x_1x_2^2 + 49x_1^2 + 37x_1x_2 + 49x_2^2 + 7x_1 + 7x_2 + 50 \\ f_2 &= 29x_1^2x_2^2 + 48x_1^2x_2 + 48x_1x_2^2 + 2x_1^2 + 16x_1x_2 + 2x_2^2 + 6x_1 + 6x_2 + 32 \end{aligned}.$$

One can note that  $f_1, f_2 \in \mathbb{F}_{53}[x_1, x_2]^{\mathfrak{S}_2}$  and

$$\begin{aligned}\Omega_{\mathfrak{S}_2}(f_1) &= 49y_1^2 + 27y_1y_2 + 33y_2^2 + 7y_1 + 45y_2 + 50 \\ \Omega_{\mathfrak{S}_2}(f_2) &= 2y_1^2 + 48y_1y_2 + 29y_2^2 + 6y_1 + 12y_2 + 32\end{aligned}$$

Assume we look for the solutions of  $\mathcal{S} = \{f_1, f_2\}$  that are in  $(\mathbb{F}_{53})^2$ . The evaluation of the elementary symmetric polynomials in these solutions are also in  $\mathbb{F}_{53}$ . Hence, we look for the solutions of  $\mathcal{S}_{\mathfrak{S}_2} = \{\Omega_{\mathfrak{S}_2}(f_1), \Omega_{\mathfrak{S}_2}(f_2)\}$  in  $(\mathbb{F}_{53})^2$ .

The unique solution of  $\mathcal{S}_{\mathfrak{S}_2}$  in  $(\mathbb{F}_{53})^2$  is  $(13, 1)$ . Thus, computing the solutions of  $\mathcal{S}$  in  $(\mathbb{F}_{53})^2$  is reduced to find the solutions in  $\mathbb{F}_{53}$  of the univariate polynomial

$$f = x^2 - 13x + 1.$$

Such solutions are given by 24 and 42. By consequence, the solutions of  $\mathcal{S}$  in  $(\mathbb{F}_{53})^2$  are  $(24, 42)$  and  $(42, 24)$ .

## CHAPTER 4

# Change of ordering

---

### Contents

---

|                                                                                                 |           |
|-------------------------------------------------------------------------------------------------|-----------|
| <b>4.1 Computing the LEX Gröbner basis given the multiplication matrices . . . . .</b>          | <b>83</b> |
| 4.1.1 Triangular set . . . . .                                                                  | 83        |
| 4.1.2 Shape Position case . . . . .                                                             | 85        |
| <b>4.2 Computing the multiplication matrices using fast linear algebra</b>                      | <b>86</b> |
| <b>4.3 Polynomial equations with fixed degree: the tame case . . . . .</b>                      | <b>88</b> |
| 4.3.1 General Complexity analysis . . . . .                                                     | 89        |
| 4.3.2 Complexity for regular systems . . . . .                                                  | 91        |
| <b>4.4 A worst case ultimately not so bad . . . . .</b>                                         | <b>93</b> |
| <b>4.5 Polynomial equations with non-fixed degree: the wild case . . . . .</b>                  | <b>94</b> |
| 4.5.1 Reading directly $T_n$ from the Gröbner basis . . . . .                                   | 94        |
| 4.5.2 Another algorithm for polynomial systems solving . . . . .                                | 96        |
| <b>4.6 Impact of Algorithm 16 on the practical solving of PoSSo in the worst case . . . . .</b> | <b>99</b> |

---

*The results presented in this chapter are from a joint work with J.-C. Faugère, P. Gaudry and G. Renault.*

We are interested in the complexity of *polynomial systems solving*. As mentioned in Chapter 2 the PoSSo problem is stated as follows.

**Problem 4.1 (PoSSo).** *Given a set of polynomial equations  $\mathcal{S} = \{f_1 = \dots = f_s = 0\}$  with  $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ . Assume the system  $\mathcal{S}$  has a finite number of solutions counted with multiplicities in the algebraic closure of  $\mathbb{K}$ . The PoSSo problem consists of computing the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$  from which one can easily compute the solutions of  $\mathcal{S}$ .*

In the best of our knowledge, for the complexity of computing the LEX Gröbner basis, there is no better bound than  $O(nD^3)$ . The main goal of this chapter is to pass over this theoretical barrier and thus providing the first algorithm with sub-cubic complexity in  $D$  to solve the PoSSo problem.

In order to reach this goal we develop new algorithms for change of ordering for Gröbner bases. As mentioned in Chapter 2, from a DRL Gröbner basis, one can compute the corresponding LEX Gröbner basis by using a change of ordering algorithm (Algorithm 11). The first step of Algorithm 11 can be done by using  $F_4$  [Fau99] or  $F_5$  [Fau02] algorithms. The complexity of these algorithms for *regular systems* (Definitions 2.66 and 2.79) is well-handled. For the particular case of the DRL order, computing a DRL Gröbner basis of a regular system

in  $\mathbb{K}[x_1, \dots, x_n]$  with equations of same degree,  $d$ , can be done in  $\tilde{O}(e^{\omega n} d^{\omega n})$  arithmetic operations (see [BFSY05, Laz83] and Chapter 2 Theorems 2.77 and 2.83). Moreover, the number of solutions  $D$  of the system can be bounded by  $d^n$  by using the Bézout's bound. Since, this bound is generically (*i.e.* almost always) reached *i.e.*  $D = d^n$ , computing a DRL Gröbner basis can be done in  $\tilde{O}(e^{\omega n} D^{\omega})$  arithmetic operations. Hence, in this case the first step of Algorithm 11 has a polynomial arithmetic complexity in the number of solutions with exponent  $\omega$ .

The second step of Algorithm 11 can be done by using a change of ordering algorithm. As mentioned in Chapter 2, change of ordering for zero dimensional ideals is closely related to linear algebra. However, from now on the complexity of change of ordering has never been related to the complexity of matrix multiplication. Indeed, the first step of the FGLM algorithm (Algorithm 8) required  $O(nD)$  dependent matrix-vector products to compute the multiplication matrices and hence has a total complexity of  $O(nD^3)$  arithmetic operations. Moreover, the second step requires to test the linear dependency of  $O(nD)$  vectors which still yields a complexity of  $O(nD^3)$  arithmetic operations. In consequence, solving regular zero-dimensional systems can be done in  $O(nD^3)$  arithmetic operations and change of ordering appears as the bottleneck of PoSSo in this case.

**Fast Linear Algebra.** Since the second half of the 20th century, an elementary issue in theoretical computer science was to decide if most of linear algebra problems can be solved by using fast matrix multiplication and consequently bound their complexities by that of multiplying two dense matrices *i.e.*  $O(m^{\omega})$  arithmetic operations where  $m \times m$  is the size of the matrix and  $2 \leq \omega < 2.3727$ . This upper bound for  $\omega$  was obtained by Vassilevska Williams in [VW12]. For instance, Bunch and Hopcroft showed in [BH74] that the inverse or the triangular decomposition can be done by using fast matrix multiplication. Baur and Strassen investigated the determinant in [BS83]. The case of the characteristic polynomial was treated by Keller-Gehrig in [KG85]. Although the link between linear algebra and the change of ordering has been highlighted for several years, relating the complexity of the change of ordering with fast matrix multiplication complexity is still an open issue.

**Main results.** The aim of this chapter is then to give an initial answer to this question in the context of polynomial systems solving *i.e.* for the special case of the DRL and LEX orderings. More precisely, our main results are summarized in the following theorems. Let  $\mathcal{S}$  be a polynomial system. First, if the equations in  $\mathcal{S}$  have bounded degree then we present a *deterministic* algorithm to compute the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$  if it is a triangular set (see Definition 2.55 in a sub-cubic complexity in  $D$  and  $d^n$ ).

**Theorem 4.2.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a polynomial system admitting a triangular set as LEX Gröbner basis. If the sequence  $(f_1, \dots, f_n)$  is a regular sequence and if the degree of each polynomial  $f_i$  ( $i = 1, \dots, n$ ) is uniformly bounded by a fixed integer  $d$  (*i.e.*  $d$  does not tend to infinity) then there exists a deterministic algorithm solving Problem 4.1 in  $\tilde{O}(e^{\omega n} d^{\omega n} + D^{\omega})$  arithmetic operations.*

Then we present a *Las Vegas* algorithm extending the result of Theorem 4.2 to polynomial systems not necessarily having a triangular set as LEX Gröbner basis and whose equations have non fixed degree *i.e.* the degree of the equations tends to infinity.

**Theorem 4.3.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a polynomial system generating a radical ideal. If the sequence  $(f_1, \dots, f_n)$  is a regular sequence where the degree of each polynomial is uniformly bounded by a non fixed parameter  $d$  then there exists a Las Vegas algorithm solving Problem 4.1 in  $\tilde{O}(e^{\omega n} d^{\omega n} + D^{\omega})$  arithmetic operations.*

As previously mentioned, the Bézout bound allows to bound  $D$  by  $d^n$  and generically (*i.e.* for generic systems) this bound is reached *i.e.*  $D = d^n$ . By consequence, Theorem 4.2 and Theorem 4.3 means that if the number of variables is fixed (respectively tends to infinity) computing the LEX Gröbner basis of generic polynomial systems can be done in  $\tilde{O}(D^{\omega})$  (respectively  $\tilde{O}(e^{\omega n} D^{\omega})$ ) arithmetic operations.

In Figure 4.1 we show the impact of our algorithms for change of ordering (denoted “Fast FGLM”) on the complexity of solving the PoSSo problem.

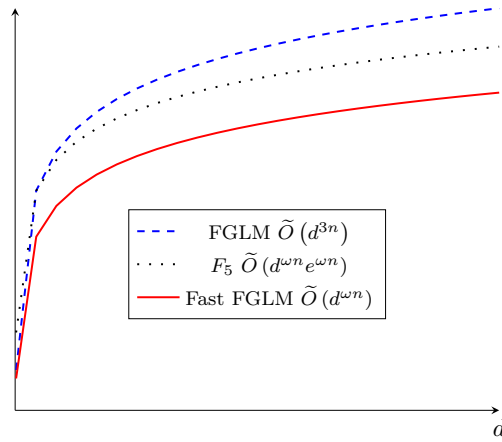


Figure 4.1: Complexity of change of ordering and  $F_5$  steps in the polynomial system solving process with  $d \rightarrow \infty$  and  $n = 20$ .

**Outline of the algorithms.** In 2011, Faugère and Mou proposed in [FM11] another kind of change of ordering algorithm to take advantage of the sparsity of the multiplication matrices. Nevertheless, when the multiplication matrices are not sparse, the complexity is still in  $O(D^3)$  arithmetic operations. Moreover, these complexities are given assuming that the multiplication matrices have already been computed and the authors of [FM11] do not investigate their computation whose complexity is still in  $O(nD^3)$  arithmetic operations. In FGLM, the matrix-vectors products (respectively linear dependency tests) are intrinsically sequential. This dependency implies a sequential order for the computation of the matrix-vectors products (respectively linear dependency tests) on which the correctness of this algorithm strongly relies. Thus, in order to decrease the complexity to  $\tilde{O}(D^{\omega})$  we need to propose new algorithms.

To achieve result in Theorem 4.2 we propose two kinds of algorithm in  $\tilde{O}(D^{\omega})$ , each of them corresponding to a step of the FGLM algorithm.

**Multiplication matrices.** We first present an algorithm to compute multiplication matrices assuming that we have already computed a Gröbner basis  $\mathcal{G}_{>_1}$ . The bottleneck of the existing algorithm [FGLM93] came from the fact that  $nD$  normal forms have to be computed in a sequential order. The key idea is to show that we can compute *simultaneously* the normal

form of all monomials *of the same degree* by computing the row echelon form of a well chosen matrix. Hence, we replace the  $nD$  normal form computations by  $\log_2(D)$  (we iterate degree by degree) row echelon forms on matrices of size  $(nD) \times (nD + D)$ . To compute simultaneously these normal forms we observe that if  $r$  is the normal form of a monomial  $m$  of degree  $d - 1$  then  $m - r$  is a polynomial in the ideal of length at most  $D + 1$ ; then we generate the Macaulay matrix of all the products  $x_i m - x_i r$  (for  $i$  from 1 to  $n$ ) together with the polynomials  $g$  in the Gröbner basis  $\mathcal{G}_{>1}$  of degree exactly  $d$ . We recall that the Macaulay matrix of some polynomials ([Laz83, Mac94] and Definition 2.61) is a matrix whose rows consist of the coefficients of these polynomials and whose columns are indexed with respect to the monomial ordering. Computing a row echelon form of the concatenation of all the Macaulay matrices in degree less or equal to  $d$  enable us to obtain all the normal forms of all monomials of degree  $d$ . This yields an algorithm to compute the multiplication matrices of arithmetic complexity  $O(\delta n^\omega D^\omega)$  where  $\delta$  is the maximal degree of the polynomials in  $\mathcal{G}_{>1}$ ; note that this algorithm can be seen as a redundant version of  $F_4$  or  $F_5$ .

In order to prove Theorem 4.3 we use the fact that, in a generic case, only the multiplication matrix by the *smallest variable* is needed. Surprisingly, we show (Theorem 4.16) that, in this situation, *no arithmetic* operation is required to build the corresponding matrix. Moreover, for non generic polynomial systems, we prove (Corollary 4.19) that a generic linear change of variables bring us back to this case.

**Computing the new basis.** The second kind of algorithm we describe is to treat the second step of Algorithm 8 *i.e.* line 2 to line 16. First, we focus on the case where the LEX Gröbner basis is a triangular set. In that case, since we know the shape of the LEX Gröbner basis, we can predict in advance the matrix-vector products to compute in the second step of FGLM algorithm and also the linear relations to find. More precisely, at step  $i$  we look for the polynomial of the form  $x_i^{d_i} + h_i(x_1, \dots, x_n)$  of the LEX Gröbner basis. To compute it we first have to compute matrix-vector products of the form  $T_i^{j_i} \cdots T_n^{j_n} \mathbf{1}$  where  $j_k = 0, \dots, d_k - 1$  and  $j_i = 0, \dots, \delta_i$  with  $\delta_i \geq d_i$ ,  $T_1, \dots, T_n$  are the  $n$  multiplication matrices,  $\mathbf{1} = (1, 0, \dots, 0)^t$  and  $\delta_i \prod_{k=i+1}^n d_k - 1 = D$ . In order to compute efficiently these matrix-vector products we generalize the algorithm of Keller-Gehrig [KG85]. We assume that the vectors  $T_{i+1}^{j_{i+1}} \cdots T_n^{j_n} \mathbf{1}$  are known for  $j_k = 0, \dots, d_k - 1$ . We denote by  $M_{i,\ell}$  the matrix containing all the vectors of the form  $T_i^\ell T_{i+1}^{j_{i+1}} \cdots T_n^{j_n} \mathbf{1}$  for  $j_k = 0, \dots, d_k - 1$ . First we compute  $T_i^2, T_i^4, \dots, T_i^{2^{\lceil \log_2 \delta_i \rceil}}$  using binary powering; then all the vectors  $T_i^{j_i} \cdots T_n^{j_n}$  for  $j_k = 0, \dots, d_k - 1$  and  $j_i = 0, \dots, \delta_i$  are computed by performing  $\log_2 \delta_i$  matrix products of the form  $T_i^{2^k} M_k$  where  $M_k$  is a matrix containing all the matrices  $M_{i,0}, \dots, M_{i,2^k-1}$  defined above. Then, to recover the corresponding polynomial in the LEX Gröbner basis we have to perform the row echelon form of a matrix of size  $D \times 2D$ . This yields an algorithm in  $O(n \log_2(D) D^\omega)$  to compute LEX Gröbner bases that are a triangular set.

Then, we focus on the particular case of *Shape Position* ideals (Definition 2.56). We present an algorithm to treat the second step of Algorithm 8 (line 2 to line 16) which is an adaptation of the algorithm given in [FM11] when the ideal is in *Shape Position*. In that case, only the multiplication matrix by the *smallest variable* is needed. When the multiplication matrix  $T$  of size  $D \times D$  is dense, the  $O(D^3)$  arithmetic complexity in [FM11] came from the  $2D$  matrix-vector products  $T^i \mathbf{r}$  for  $i = 1, \dots, 2D$  where  $\mathbf{r}$  is a column vector of size  $D$ . To decrease the complexity we follow same idea as in the previous algorithm *i.e.* we use the Keller-Gehrig algorithm [KG85]: first, we compute  $T^2, T^4, \dots, T^{2^{\lceil \log_2 D \rceil}}$  using binary powering; second,

all the products  $T^i \mathbf{r}$  are recovered by computing  $\log_2 D$  matrix multiplications. Then, in the Shape Position case, as in [FM11, FM13, Mou13] the  $n$  univariate polynomials of the lexicographical Gröbner basis are computed by solving  $n$  structured linear systems (Hankel matrices) in  $O(nD \log_2^2(D))$  operations. We thus obtain a change of ordering algorithm (DRL to LEX order) for *Shape Position* ideals whose complexity is in  $O(\log_2(D)(D^\omega + n \log_2(D)D))$  arithmetic operations.

In the following section, we first present an algorithm to compute the LEX Gröbner basis of an ideal having as LEX Gröbner basis a triangular set. Then we focus on *Shape Position* ideals. These algorithms assume the DRL Gröbner basis and the multiplication matrices to be known.

## 4.1 Computing the LEX Gröbner basis given the multiplication matrices

In this section, we present two algorithms to compute the LEX Gröbner basis given the DRL Gröbner basis. The first algorithm for ideals having a triangular set as LEX Gröbner basis follows same ideas as developed in the FGLM algorithm. The second algorithm for *Shape Position* ideals follows the one described in [FM11]. The main difference is that this new algorithm and its complexity study do not take into account any structure of the multiplication matrices (in particular any sparsity assumption).

These two algorithms share the use of Keller-Gehrig algorithm to compute particular matrix-vector products using matrix multiplication.

### 4.1.1 Triangular set

In this section, we assume the multiplicative structure of the quotient ring to be known. That is to say, the  $n$  multiplication matrices (Definition 2.32)  $T_1, \dots, T_n$  are assumed to be known. Let  $\mathcal{G}_{>_{\text{lex}}} = \{x_1^{d_1} + h_1(x_1, \dots, x_n), \dots, x_{n-1}^{d_{n-1}} + h_{n-1}(x_{n-1}, x_n), x_n^{d_n} + h_n(x_n)\}$  be the LEX Gröbner basis to compute. Where  $\deg_{x_j}(h_i) < d_j$  for any  $1 \leq i \leq j \leq n$ . The degree of  $\mathcal{I} = \langle \mathcal{G}_{>_{\text{lex}}} \rangle$  is then given by  $D = \prod_{i=1}^n d_i$ .

### Computing the polynomial $h_n$

Following FGLM algorithm, see Section 2.3.1, first we have to compute the coordinate vectors representing all the monomials  $x_n^j$  in  $\mathbb{V}_{>_{\text{drl}}}(\mathcal{I})$  for  $j = 0, \dots, d_n$ . Nevertheless we do not know in advance  $d_n$ . Thus, since  $d_n \leq D$  we compute all the coordinate vectors of  $x_n^j$  for  $j = 0, \dots, D$ . Note that this coordinate vectors can be computed as follows  $\mathbf{v}_{n,j} = T_n^j \mathbf{1}$  for any  $j \in \{0, \dots, D\}$  where  $\mathbf{1} = (1, 0, \dots, 0)^t$  is the coordinate vector of the monomial 1. In order to compute efficiently all the vectors  $\mathbf{v}_{n,j}$  we use Keller-Gehrig algorithm [KG85].

If  $\mathbf{r}_1, \dots, \mathbf{r}_k$  are column vectors then we denote by  $(\mathbf{r}_1 | \dots | \mathbf{r}_k)$  the matrix with  $D$  rows and  $k$  columns obtained by joining the vectors  $\mathbf{r}_i$  vertically. Similarly, if  $M_1, \dots, M_k$  are matrices of size  $D \times c_i$  then we denote by  $(M_1 || \dots || M_k)$  the matrix with  $D$  rows and  $\sum_{i=1}^k c_i$  columns by joining the matrices  $M_i$  vertically.

To simplify the notation let  $T$  be the transpose of  $T_n$ . First, we compute  $T^2, T^4, \dots, T^{2^{\lceil \log_2 D \rceil}}$  using binary powering with  $\lceil \log_2 D \rceil$  matrix multiplications. Similarly to [KG85],

the vectors  $T^j \mathbf{1}$  for  $j = 0, \dots, D$  are computed by induction in  $\log_2 D$  steps:

$$\begin{aligned} T^2(T\mathbf{1} \mid \mathbf{1}) &= (T^3\mathbf{1} \mid T^2\mathbf{1}) \\ T^4(T^3\mathbf{1} \mid T^2\mathbf{1} \mid T\mathbf{1} \mid \mathbf{1}) &= (T^7\mathbf{1} \mid T^6\mathbf{1} \mid T^5\mathbf{1} \mid T^4\mathbf{1}) \\ &\vdots \\ T^{2^{\lceil \log_2(D) \rceil}}(T^{2^{\lceil \log_2(D) \rceil - 1}}\mathbf{1} \mid \dots \mid \mathbf{1}) &= (T^{2^{\Delta - 1}}\mathbf{1} \mid T^{2^{\Delta - 2}}\mathbf{1} \mid \dots \mid T^{2^{\lceil \log_2(D) \rceil}}\mathbf{1}) \end{aligned} \quad (4.1)$$

where  $\Delta$  is the smallest power of two satisfying  $\Delta \geq D$ .

Then, the row echelon form  $E$  of the matrix  $M$  containing the  $D + 1$  vectors  $\mathbf{v}_{n,0}, \dots, \mathbf{v}_{n,D}$  allows to recover the polynomial  $x_n^{d_n} + h_n(x_n)$ . Indeed,  $d_n$  is given by the rank of  $M$ . Moreover, the invertible matrix  $P$  satisfying  $E = PM$  gives the linear dependency between the vectors  $\mathbf{v}_{n,0}, \dots, \mathbf{v}_{n,d_n}$  and allows to compute  $h_n$ .

### Computing the remaining polynomials $h_1, \dots, h_{n-1}$

To compute the others polynomials, we generalize the idea to compute  $h_n$ . At step  $i$ , we look for  $h_i$  and we assume that  $d_n, \dots, d_{i+1}$  are known and the coordinate vectors  $v_{i,j_n, \dots, j_{i+1}} = T_n^{j_n} \dots T_{i+1}^{j_{i+1}} \mathbf{1}$  are also known for any  $j_k \in \{0, \dots, d_k - 1\}$  for  $k = i + 1, \dots, n$ .

We know that  $d_i \leq \frac{D}{\prod_{j=i+1}^n d_j} = \delta_i$ . Hence, we compute all the coordinate vectors  $v_{i,j_n, \dots, j_{i+1}, \ell} = T_n^{j_n} \dots T_{i+1}^{j_{i+1}} T_i^\ell \mathbf{1}$  for  $j_k \in \{0, \dots, d_k - 1\}$  with  $k = i + 1, \dots, n$  and for  $\ell = 0, \dots, \delta_i$ . Note that the number of such coordinate vectors is  $(\delta_i + 1) \prod_{i=i+1}^n d_i = D + \prod_{i=i+1}^n d_i \leq 2D$ .

Let  $M_{i,\ell}$  be the matrix constructed from the vectors  $v_{i,j_n, \dots, j_{i+1}, \ell} = T_n^{j_n} \dots T_{i+1}^{j_{i+1}} T_i^\ell \mathbf{1}$  for  $j_k \in \{0, \dots, d_k - 1\}$  with  $k = i + 1, \dots, n$  i.e.

$$M_{i,\ell} = (v_{i,0, \dots, 0, \ell} \mid v_{i,1, \dots, 0, \ell} \mid v_{i,2, \dots, 0, \ell} \mid \dots \mid v_{i,d_n - 1, \dots, d_{i+1} - 1, \ell})$$

is a matrix of size  $D \times \prod_{j=i+1}^n d_j$ . Note that  $M_{i,0}$  is known and  $M_{i,1} = T_i M_{i,0}$ .

In order to compute efficiently all the coordinate vectors  $\mathbf{v}_{i,j_n, \dots, j_{i+1}, \ell}$  we first compute  $T_i^2, T_i^4, \dots, T_i^{2^{\lceil \log_2 \delta_i \rceil}}$  using binary powering with  $\lceil \log_2 \delta_i \rceil$  matrix multiplications. Then, the vectors  $T_i^\ell T_n^{j_n} \dots T_{i+1}^{j_{i+1}} \mathbf{1}$  for  $\ell = 0, \dots, \delta_i$  are computed by induction in  $\log_2 \delta_i$  steps:

$$\begin{aligned} T_i^2(M_{i,1} \parallel M_{i,0}) &= (M_{i,3} \parallel M_{i,2}) \\ T_i^4(M_{i,3} \parallel M_{i,2} \parallel M_{i,1} \parallel M_{i,0}) &= (M_{i,7} \parallel M_{i,6} \parallel M_{i,5} \parallel M_{i,4}) \\ &\vdots \\ T_i^{2^{\lceil \log_2(\delta_i) \rceil}}(M_{i,2^{\lceil \log_2(\delta_i) \rceil - 1}} \parallel \dots \parallel M_{i,0}) &= (M_{i,2^{\Delta} - 1} \parallel M_{i,2^{\Delta} - 2} \parallel \dots \parallel M_{i,2^{\lceil \log_2(\delta_i) \rceil}}) \end{aligned} \quad (4.2)$$

where  $\Delta$  is the smallest power of two satisfying  $\Delta \geq \delta_i$ .

Finally, let  $M = (M_{i,0} \parallel \dots \parallel M_{i,\delta_i})$  be a matrix of size  $(D \times D + \prod_{j=i+1}^n d_j)$ . Let  $r$  be the rank of  $M^t$ ,  $d_i$  is then given by  $d_i = \frac{r}{\prod_{j=i+1}^n d_j}$ . Let  $E$  be the row echelon form of  $M^t$  and let  $P$  be the invertible matrix satisfying  $E = PM^t$ , the linear dependency between the  $v_{i,0, \dots, 0, d_i}$  and the coordinate vectors  $v_{i,j_n, \dots, j_{i+1}, \ell}$  for  $j_k = 0, \dots, d_k$  with  $k = i + 1, \dots, n$  and for  $\ell = 0, \dots, d_i - 1$  can be read from  $P$  and gives  $h_i$ .

This algorithm to compute LEX Gröbner basis when it is a triangular set is summarized in Algorithm 13.

**Algorithm 13:** LEX Gröbner basis computation as a triangular set.

---

**Input** :  $\mathcal{I}$  be an ideal having a triangular set for LEX Gröbner basis and the  $n$  multiplication matrices  $T_1, \dots, T_n$  representing the multiplication by  $x_1, \dots, x_n$  in  $\mathbb{V}_{>\text{drl}}(\mathcal{I})$ .

**Output**: The LEX Gröbner basis of  $\mathcal{I}$ .

```

1  $M_{n,0} := \mathbf{1}; M_{n,1} := T_n \mathbf{1}; G := \emptyset; d := 1;$ 
2 for  $i := n$  to 1 do
3    $\delta_i := \frac{D}{d};$ 
4   Compute  $T_i^{2^j}$  for  $j = 1, \dots, \lceil \log_2(\delta_i) \rceil;$ 
5   Compute  $M_{i,2}, \dots, M_{i,\delta_i}$  using induction (4.2);
6    $M := (M_{i,0} \parallel \dots \parallel M_{i,\delta_i});$ 
7   Compute  $E$  the row echelon form of  $M^t$  and  $P$  such that  $E = PM^t;$ 
8    $d_i := \frac{\text{Rank}(E)}{d}; d := d \times d_i;$ 
   Read from  $P$ ,  $c, c_{j_n, \dots, j_i} \in \mathbb{K}$  s.t.  $c \cdot T_i^{d_i} \mathbf{1} + \sum_{\substack{j_k=0, \dots, d_k-1 \\ k \in \{i, \dots, n\}}} c_{j_n, \dots, j_i} \cdot T_n^{j_n} \dots T_i^{j_i} \mathbf{1} = \mathbf{0};$ 
9    $f := x_i^{d_i} + \sum_{\substack{j_k=0, \dots, d_k-1 \\ k \in \{i, \dots, n\}}} \frac{c_{j_n, \dots, j_i}}{c} x_n^{j_n} \dots x_i^{j_i};$ 
10  Append  $f$  to  $G;$ 
11  if  $i > 1$  then  $M_{i-1,0} := M; M_{i-1,1} := T_{i-1} M_{i-1,0};$ 
12
13 return  $G;$ 
```

---

The second algorithm that we present is for the particular case of ideals in *Shape Position*. Note that the *Shape Position* case is a particular case of triangular set. Hence, Algorithm 13 can be used. However, the algorithm presented in the next section is more efficient in practice. Moreover, only the multiplication matrix  $T_n$  is required as input of this algorithm. This will be useful to speed up the computation of the multiplicative structure of the quotient ring in the whole change of ordering algorithm (see Section 4.5).

#### 4.1.2 Shape Position case

The idea is the same as above but instead of following FGLM algorithm, we follow the efficient algorithm of Faugère and Mou for *Shape Position* ideals. Let  $\mathcal{G}_{>\text{lex}} = \{h_n(x_n), x_{n-1} - h_{n-1}(x_n), \dots, x_1 - h_1(x_n)\}$  be the LEX Gröbner basis of  $\mathcal{I}$ . In Section 2.3.2 (respectively Section 2.3.3) we saw that Faugère and Mou have proposed a probabilistic (respectively deterministic) change of ordering algorithm to compute the LEX Gröbner basis of *Shape Position* ideals. Indeed, given the linearly recurrent sequence  $S = [(\mathbf{r}, T_n^j \mathbf{1}) \mid j = 0, \dots, 2D - 1]$  where  $\mathbf{r}$  is a random column vector (respectively the linearly recurrent sequence  $S_i = [(\mathbf{e}_i, T_n^j \mathbf{1}) \mid j = 0, \dots, 2D - 1]$  where  $\mathbf{e}_i$  is the  $i$ th canonical vector) we saw that computing the LEX Gröbner basis is reduced to solve Hankel linear systems. Which can be done very efficiently.

In order to compute efficiently  $S$  they note that  $(\mathbf{r}, T_n^j \mathbf{1}) = (T^j \mathbf{r}, \mathbf{1})$  where  $T = (T_n)^t$ . Consequently, computing  $S$  (respectively  $S_i$ ) can be done by extracting the first (respectively the  $i$ th) component of the vectors  $T^j \mathbf{r}$  (respectively  $T_n^j \mathbf{1}$ ) for  $j \in \{0, \dots, 2D - 1\}$ .

Since they consider that the multiplication matrix  $T_n$  is sparse, they compute iteratively

the matrix-vector products *i.e.*  $T^{j+1}\mathbf{r} = T(T^j\mathbf{r})$  or  $T_n^{j+1}\mathbf{1} = T_n(T_n^j\mathbf{1})$ . However, when the matrix  $T_n$  is dense this yields an algorithm with cubic complexity in  $D$ . In order to compute these matrix-vector products using multiplication matrices we use the algorithm of Keller-Gehrig as presented in equation (4.1) where the matrix and the vector are chosen according to the wanted matrix-vector products.

**Remark 4.4.** *In the case of the deterministic algorithm, we also need to compute the matrix-vector products  $T_n^j\mathbf{w}_k$  for  $j = 0, \dots, d_i - 1$  to compute the vector  $\mathbf{b}_{i,k}$  where  $d_i \leq D$  and  $\mathbf{w}_k = T_k\mathbf{1}$  (see Section 2.3.3 for notations and description of this algorithm). Hence, we also use induction (4.1) to compute these matrix-vector products. Consequently, for the deterministic algorithm we use  $n$  times the induction 4.1.*

Following notations of Section 2.3.2, we summarize the probabilistic algorithm in Algorithm 14.

---

**Algorithm 14:** Probabilistic change of ordering for *Shape Position* ideals.

---

**Input** : The multiplication matrix  $T_n$  and the DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of an ideal  $\mathcal{I}$ .  
**Output**: Return the LEX Gröbner basis  $\mathcal{G}_{>\text{lex}}$  of  $\mathcal{I}$  or *fail*.

- 1  $T := T_n^t$ ;  $\mathbf{r} :=$  Random column vector in  $\mathbb{K}^D$ ;
- 2 Compute  $T^{2^i}$  for  $i = 0, \dots, \lceil \log_2 D \rceil$  and compute  $T^j\mathbf{r}$  for  $j = 0, \dots, (2D - 1)$  using induction (4.1). Deduce the linearly recurrent sequence  $S$  and the Hankel matrix  $\mathcal{H}$ ;
- 3  $h_n(x_n) := \text{BerlekampMassey}(S)$ ;
- 4 **if**  $\deg(h_n) = D$  **then**
- 5     Let  $\mathcal{L}^c = \{j \in \{1, \dots, n-1\} \text{ such that } \text{NF}_{>\text{drl}}(x_j) = x_j\}$  and  
        $\mathcal{L} = \{1, \dots, n-1\} \setminus \mathcal{L}^c$ ;
- 6     **for**  $j \in \mathcal{L}^c$  **do**
- 7         Deduce  $T_j\mathbf{1}$  and  $\mathbf{b}_j$  then solve the structured linear system  $\mathcal{H}\mathbf{c}_j = \mathbf{b}_j$ ;
- 8          $h_j(x_n) := \sum_{i=0}^{D-1} c_{j,i}x_n^i$  where  $c_{j,i}$  is the  $i$ th component of the vector  $\mathbf{c}_j$ ;
- 9     **for**  $j \in \mathcal{L}$  **do**
- 10          $h_j(x_n) := -\sum_{i \in \mathcal{L}^c} \alpha_{j,i}h_i(x_n) - \alpha_{j,n}h_n(x_n) - \alpha_{j,0}$  where  $\alpha_{j,i}$  is the  $i$ th coefficient  
        of the linear form whose leading term is  $x_j$ ;
- 11     **return**  $[x_1 - h_1(x_n), \dots, x_{n-1} - h_{n-1}(x_n), h_n(x_n)]$ ;
- 12 **else return** *fail*;

---

In the next section, we show how to use fast matrix multiplication to compute all the multiplication matrices.

## 4.2 Computing the multiplication matrices using fast linear algebra

Let  $B = \{\epsilon_D >_{\text{drl}} \dots >_{\text{drl}} \epsilon_1 = 1\}$  be the canonical basis w.r.t. the DRL ordering of  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  seen as a  $\mathbb{K}$ -vector space. In Section 2.3.1, we show that computing the multiplication matrices  $T_1, \dots, T_n$  consists of computing the normal form of the monomials  $\epsilon_i x_j$  for  $i = 1, \dots, D$  and  $j = 1, \dots, n$ . From Proposition 2.68 we saw that this can be done by performing at most  $nD$  matrix-vector products which yields a cubic complexity in  $D$  to compute  $T_1, \dots, T_n$ .

Another way to compute the normal form of a term  $t$  is to find the unique polynomial in the ideal whose leading term is  $t$  and the others terms correspond to monomials in  $B$ . Hence, to compute the multiplication matrices, we look for the polynomial  $t - \text{NF}_{>\text{drl}}(t)$  for any  $t$  in the frontier  $F$ . We recall that the *frontier* is the set  $F = \{x_j \epsilon_i \mid j = 1, \dots, n \text{ and } i = 1, \dots, D\} \setminus B$ . Therefore, to compute these polynomials we proceed in two steps. First, we construct a polynomial in the ideal whose leading term is  $t$ . If  $t$  is the leading term of a polynomial  $g$  in  $\mathcal{G}_{>\text{drl}}$  then the desired polynomial is  $g$  itself. Otherwise,  $t$  is of type **III** of Proposition 2.68 and  $t = x_k t'$  with  $t' \in F$  and  $\deg(t') < \deg(t)$ . We will proceed degree by degree so that we can assume that we know a polynomial  $f'$  in the ideal whose leading term is  $t'$ ; then the desired polynomial is  $f = x_k f'$ . Next, once we have all the polynomials  $f$  with all possible leading terms  $t$  of some degree  $d$ , we can recover the canonical form  $t - \text{NF}_{>\text{drl}}(t)$  by reducing  $f$  with respect to the others polynomials whose leading terms are less than  $t$ . By computing a reduced row echelon form of the Macaulay matrix of all these polynomials, we can reduce all of them simultaneously.

Following the idea presented above, we can now describe Algorithm 15 for computing all the multiplication matrices  $T_i$ . Assuming that  $F$  is sorted in increasing order w.r.t.  $>\text{drl}$  i.e.  $F = \{t_{\#F} >\text{drl} \dots >\text{drl} t_1\}$ , we define the linear map  $\phi$ :

$$\phi : \left( \begin{array}{ccc} \mathbb{A} & \rightarrow & \mathbb{K}^{D+\#F} \\ \sum_{i=1}^D \alpha_i \epsilon_i + \sum_{j=1}^{\#F} \beta_j t_j & \mapsto & (\beta_{\#F}, \dots, \beta_1, \alpha_1, \dots, \alpha_D) \end{array} \right)$$

Let  $M$  be a row indexed matrix by all the monomials in  $F$ . Let  $m$  be a monomial in  $F$  and  $i$  the position of  $m$  in  $F$ ,  $M[m]$  denotes the row of  $M$  of index  $m$  i.e. the  $(\#F - i + 1)^{\text{th}}$  row of  $M$  containing a polynomial of leading term  $m$ . If  $T$  is a matrix,  $T[* , i]$  denotes the  $i^{\text{th}}$  column of  $T$ .

**Proposition 4.5.** *Algorithm 15 is correct.*

*Proof.* The key point of the algorithm is to ensure that for each monomial in  $F$  its normal form is computed and stored in NF before we use it. We will prove the following loop invariant for all  $d$  in  $\{d_{\min}, \dots, d_{\max}\}$ .

*Loop invariant:* at the end of step  $d$ , all the normal forms of the monomials of degree  $d$  in the frontier  $F$  are computed and are stored in NF. Moreover, the  $m$ th row of the matrix  $M$  contains  $\phi(m - \text{NF}_{>\text{drl}}(m))$  for any monomial  $m \in F_d$ .

First, we assume that  $d = d_{\min}$ . Then, each monomial  $t$  of degree  $d$  in  $F$  is of type **(II)** of Proposition 2.68. Indeed, if  $t$  was of type **(III)** then there exists  $t'$  in  $F$  of degree  $d - 1$  which divides  $t$ . This is impossible because  $t' \in F_{d_{\min}-1} = \emptyset$ . Hence, the normal form of  $t$  for  $t \in F_{d_{\min}}$  is known and  $M[t]$  contains  $\phi(g)$  with  $g$  the unique element of  $\mathcal{G}_{>\text{drl}}$  such that  $\text{LT}_{>\text{drl}}(g) = t$ . Hence,  $M[t] = \phi(g) = \phi(t - \text{NF}_{>\text{drl}}(t))$ . Moreover, since  $\mathcal{G}_{>\text{drl}}$  is a reduced Gröbner basis, the matrix  $M$  is already in reduced row echelon form. Thus, the loop in Line 9 updates  $\text{NF}[t]$  for all  $t \in F_d$ .

Let  $d > d_{\min}$ , we now assume that the loop invariant is true for any degree less than  $d$ . For all  $t \in F_d$  the  $t$ th row of  $M$  contains either  $\phi(t - \text{NF}_{>\text{drl}}(t))$  if  $t$  is of type **(II)** or  $\phi(t - x_k \text{NF}[t'])$  if  $t$  is of type **(III)**. Since  $\deg(t') = d - 1$ , by induction its normal form is known and in NF. Hence  $\text{NF}[t'] = \text{NF}_{>\text{drl}}(t')$  and  $M[t] = \phi(x_k(t' - \text{NF}_{>\text{drl}}(t')))$ . A first consequence is that, before Line 8, since we sort  $F_d$  at each step,  $M$  is an upper triangular matrix with  $M[t, t] = 1$  for all  $t \in F_d$ , see Figure 4.2. Note that sorting  $F_d$  is required only to obtain this triangular form. Let  $f$  be the polynomial  $\text{NF}_{>\text{drl}}(t')$ . Writing  $f = \sum_{j=1}^D \lambda_j \epsilon_j$  we have that  $\lambda_j = 0$

---

**Algorithm 15:** Building multiplication matrices (in the following || does not mean parallel code but gives details about pseudo code on the left side).

---

**Input** : The DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of an ideal  $\mathcal{I}$ .  
**Output**: The  $n$  multiplication matrices  $T_1, \dots, T_n$ .

```

1 Compute  $B = \{\epsilon_D >_{\text{drl}} \dots >_{\text{drl}} \epsilon_1\}$  and  $F = \{x_i \epsilon_j \mid i = 1, \dots, n \text{ and } j = 1, \dots, D\} \setminus B$ ,  

 $S := \#F$ ;  

2  $d_{\min} := \min(\{\deg(t) \mid t \in F\})$ ;  $d_{\max} := \max(\{\deg(t) \mid t \in F\})$ ;  $\text{NF} := []$ ;  

3  $M :=$  the zero matrix of size  $nD \times (n+1)D$  row indexed by all the monomials in  $F$ ;  

4 for  $d = d_{\min}$  to  $d_{\max}$  do  

5    $F_d := \text{Sort}(\{t \in F \mid \deg(t) = d\}, >_{\text{drl}})$  ;  

6   for  $m \in F_d$  do  

7     Check if we can find:  

       (i)  $g \in \mathcal{G}_{>\text{drl}}$  such that  $\text{LT}_{>\text{drl}}(g) = m$   

       (ii)  $t' \in F$  such that  $m = x_k t'$   

       Add the corresponding row to the matrix  $M$ ;  

8    $M := \text{ReducedRowEchelonForm}(M)$  ;  

9   for  $m \in F_d$  do  

10    Read  $\text{NF}_{>\text{drl}}(m)$  from  $M$ ;  

11    Construct  $T_1, \dots, T_n$  from  $\text{NF}$ ;  

return  $T_1, \dots, T_n$ ;
```

|| **if**  $m = \text{LT}_{>\text{drl}}(g)$  **then**  $M[m] := \phi(g)$ ;  
|| **else**  
|| Find  $x_k$  and  $t' \in F_{d-1}$  such that  $m = x_k t'$ ;  
||  $M[m] := \phi(m - x_k \text{NF}[t'])$ ;  
|| **for**  $\epsilon$  in  $B$  **do**  $\text{NF}[\epsilon] := \epsilon$ ;  
|| **for**  $t$  in  $F \cup B$  **do**  
|| **for**  $x_i$  s.t.  $x_i$  divides  $t$  and  $\frac{t}{x_i} = \epsilon_j \in B$  **do**  
||  $T_i[* , j] := \psi(\text{NF}[t])$ ;

---

if  $\deg(\epsilon_j) \geq d$  since  $\deg(\text{NF}_{>\text{drl}}(t')) \leq \deg(t') = d - 1$ . So that  $f = \sum_{j=1}^k \lambda_j \epsilon_j$  such that  $\deg(\epsilon_j) < d$  when  $j \leq k$ . Now for all  $j \in \{1, \dots, k\}$ ,  $x_k \epsilon_j$  are in exactly one of the following cases:

1.  $x_k \epsilon_j \in B$  so that  $\text{NF}_{>\text{drl}}(x_k \epsilon_j) = x_k \epsilon_j$  is already reduced.
2.  $x_k \epsilon_j \in F$ . Since  $d' = \deg(x_k \epsilon_j) \leq d$  it implies that  $x_k \epsilon_j \in F_{d'}$  so that the row  $M[x_k \epsilon_j]$  has been added to  $M$ .

Moreover, since each row of the matrix  $M$  contains a polynomial in the ideal  $\langle \mathcal{G}_{>\text{drl}} \rangle$  after the computation of the row echelon form, the rows of the matrix  $M$  contain also polynomials in  $\langle \mathcal{G}_{>\text{drl}} \rangle$  being linear combination of the previous polynomials. Hence, after the computation of the row echelon form of  $M$ , the row  $M[t]$  is equal to  $\phi(t - \text{NF}_{>\text{drl}}(t))$ .

By induction, this finishes the proof of the loop invariant and then of the correctness of Algorithm 15.  $\square$

### 4.3 Polynomial equations with fixed degree: the tame case

The purpose of this section, is to analyze the asymptotic complexity of Algorithm 13, Algorithm 14 and Algorithm 15 when the degrees of the equations of the input system are uniformly bounded by a fixed integer  $d > 1$  and to establish the first main result of this chapter.

### 4.3.1 General Complexity analysis

First, we study the complexity of Algorithm 13 to compute LEX Gröbner basis that is a triangular set given the multiplication matrices  $T_1, \dots, T_n$ .

**Proposition 4.6.** *Given the multiplication matrices  $T_1, \dots, T_n$  and the DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of an ideal having for LEX Gröbner basis a triangular set, its LEX Gröbner basis can be deterministically computed in  $O(n \log_2(D) D^\omega)$  where  $D$  is the number of solutions. Expressed with the input parameters of the system to solve, the complexity is  $O(n^2 \log_2(d) d^{\omega n})$  where  $d > 1$  is a (fixed) bound on the degree of the input polynomials.*

*Proof.* The complexity of Algorithm 13 is dominated by the cost of Lines 4, 5 and 7. The others computation are negligible in comparison. At Line 4 one computes  $\lceil \log_2(\delta_i) \rceil$  matrix products of size  $(D, D) \times (D, D)$ . Since  $\delta_i \leq D$  the complexity of this step for all the iterations of the loop is in  $O(n \log_2(D) D^\omega)$  arithmetic operations. At Line 5 one computes  $\lceil \log_2(\delta_i) \rceil$  matrix products of size at most  $(D, D) \times (D, D + \sum_{j=i+1}^n d_j)$  i.e. of size at most  $(D, D) \times (D, 2D)$ . Hence, the complexity of this step for all the iterations of the loop is in  $O(n \log_2(D) D^\omega)$  arithmetic operations. Finally, from [KG85] the complexity of Line 7 is in  $O(D^\omega)$  since the matrix  $M$  is of size at most  $(D, 2D)$ . Moreover, Algorithm 13 is a deterministic algorithm which concludes the proof.  $\square$

Next, we analyse Algorithm 14 to compute the LEX Gröbner basis of *Shape Position* ideals given the last multiplication matrix.

**Proposition 4.7.** *Given the multiplication matrix  $T_n$  and the DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of an ideal in *Shape Position*, its LEX Gröbner basis can be probabilistically computed in  $O(\log_2(D)(D^\omega + nD \log_2(D) \log_2 \log_2(D)))$  where  $D$  is the number of solutions. Expressed with the input parameters of the system to solve, the complexity is  $O(n \log_2(d) d^{\omega n})$  where  $d > 1$  is a (fixed) bound on the degree of the input polynomials.*

*Proof.* As usual  $T = T_n^t$  is the transpose matrix of  $T_n$ . Using the induction (4.1), the vectors  $T^j \mathbf{r}$  can be computed for all  $j = 0, \dots, (2D - 1)$  in  $O(\log_2(D) D^\omega)$  field operations. Then the linear recurrent sequence  $S$  and the matrix  $\mathcal{H}$  can be deduced with no cost. The Berlekamp-Massey algorithm compute the minimal polynomial of  $S$  in  $O(D \log_2^2(D) \log_2 \log_2(D))$  field operations [JM89, BGY80].

As defined in Section 2.3.2,  $\mathcal{L} = \{j \in \{1, \dots, n-1\} \text{ such that } \text{NF}_{>\text{drl}}(x_j) \neq x_j\}$  and  $\mathcal{L}^c = \{1, \dots, n-1\} \setminus \mathcal{L}$ . The right hand sides of the linear systems  $\mathbf{b}_i$  can be computed without field operations when  $i \in \mathcal{L}^c$ . Since the matrix  $\mathcal{H}$  is a non singular Hankel matrix, the  $\#\mathcal{L}^c$  linear systems (2.5) can be solved in  $O(\#\mathcal{L}^c \log_2^2(D) \log_2 \log_2(D) D) = O(n \log_2^2(D) \log_2 \log_2(D) D)$  field operations. Then, to recover all the  $h_i(x_n)$  for  $i \in \mathcal{L}$  we perform  $O(\#\mathcal{L} \#\mathcal{L}^c D) = O(n^2 D)$  multiplications and additions in  $\mathbb{K}$ .

Since the Bézout's bound allows to bound  $D$  by  $d^n$  with  $d$  a fixed integer we have  $\log_2(D) \leq n \log_2(d)$  and the arithmetic complexity of Algorithm 14 is  $O(\log_2(D)(D^\omega + nD \log_2(D) \log_2 \log_2(D)))$  which can be expressed in terms of  $d$  and  $n$  as  $O(n \log_2(d) d^{\omega n})$ .  $\square$

As for the probabilistic algorithm (Algorithm 14), the deterministic version of change of ordering for *Shape Position* ideals presented in Section 4.1.2 has the same complexity as the deterministic algorithm of Faugère and Mou (Theorem 2.88) presented in Section 2.3.3. Except that the  $nD \#\mathcal{T}_n$  part of the complexity due to the computation of some matrix-vector

products using the sparsity of  $T_n$  is replaced by  $n \log_2(D)D^\omega$  since we consider  $T_n$  dense and we use induction (4.1) to compute these matrix-vector products.

This deterministic version computes the LEX Gröbner basis of the radical of the ideal in input when the ideal is in *Shape Position*. Hence, this is not restricting if we assume that the ideal is radical or if we are interested only in the solutions of the system (which is generally the case when speaking about *polynomial systems solving*).

**Proposition 4.8.** *Let  $T_n$  be the multiplication matrix by the smallest variable and  $\mathcal{G}_{>\text{drl}}$  be the DRL Gröbner basis of a radical ideal  $\mathcal{I}$  in Shape Position. There is a deterministic algorithm which computes the LEX Gröbner basis of  $\mathcal{I}$  in*

- $O(n \log_2(D)D^\omega)$  arithmetic operations if  $\mathbb{K}$  is a field of characteristic zero;
- $O\left(n \log_2(D)D^\omega + D \log_2 \frac{q}{p}\right)$  arithmetic operations if  $\mathbb{K}$  is a finite field of characteristic  $p$  and size  $q$ ;

(or in  $O(n^2 \log_2(d)d^{\omega n})$ ) where  $D$  is the degree of  $\mathcal{I}$ .

Now, to complete algorithms of Section 4.1, we deal with the complexity of Algorithm 15 to compute the multiplication matrices. Note that in Proposition 4.7 and 4.8 only the last matrix  $T_n$  is needed. Before considering the complexity of Algorithm 15, we first discuss the complexity of computing  $B$  and  $F$ .

**Lemma 4.9.** *Given  $\mathcal{G}_{>\text{drl}}$  (resp.  $B$ ) the construction of  $B$  (resp.  $F$ ) requires at most  $O(n^3 D^2)$  (resp.  $O(nD^2 + n^2 D)$ ) elementary operations which can be decreased to  $O(n^2 D)$  (resp.  $O(n^2 D)$ ) elementary operations if a hash table is used.*

*Proof.* It is well known that the canonical basis  $B$  can be computed in polynomial time (but no arithmetic operations). Nevertheless, in order to be self-contained we describe an elementary algorithm to compute  $B$ . We start with the monomial 1 and we multiply it by all the variables  $x_i$  which gives  $n$  new monomials to consider. If a new monomial is not divisible by a leading term of a polynomial in  $\mathcal{G}_{>\text{drl}}$  then we keep it otherwise we discard it. At each step (we iterate degree by degree) we multiply by the variables  $x_i$  only the monomials of highest degree that we have kept and we proceed until all the new monomials are discarded. Hence, we have to test the irreducibility of all the elements in  $F \cup B$  whose total number is bounded by  $(n+1)D$ . Since  $\text{LT}_{>\text{drl}}(\mathcal{G}_{>\text{drl}}) \subset F$  we can bound the number of elements of  $\mathcal{G}_{>\text{drl}}$  by  $nD$ . Therefore, to compute  $B$  we have to test the divisibility of  $(n+1)D$  monomials by at most  $nD$  monomials. Hence, the construction of  $B$  can be done in  $O(n^3 D^2)$  elementary operations.

When using hash tables, we initialize the table  $F^+$  with all the leading terms of  $\mathcal{G}_{>\text{drl}}$ . At each step (i.e. degree  $d$ ) to test the divisibility of a monomial  $m' = x_i m$  with  $m$  in  $B$  by an element in  $\text{LT}_{>\text{drl}}(\mathcal{G}_{>\text{drl}})$  we look for it in  $F^+$  in  $O(1)$  operations. If  $m'$  is in  $F^+$  we discard it and we add  $x_j m'$  for  $j = 1, \dots, n$  to  $F^+$ . In this way since  $\{m \in F \mid \deg(m) = d+1\} \setminus E_{>\text{drl}}(\mathcal{G}_{>\text{drl}}) \subset \{x_j m \mid m \in F \text{ s.t. } \deg(m) = d\}$  we ensure that  $F_{d+1} \subset F^+$  at the end of the step  $d$ . One tests if a monomial is in  $B$  for at most  $(n+1)D$  monomials ( $B \cup F$ ). The table  $F^+$  contains at most  $n^2 D$  monomials. Each of them can be computed in  $O(1)$  operations.

From  $B$ , the construction of  $F$  requires  $nD$  monomials multiplications i.e.  $n^2 D$  additions of integers. Moreover, removing  $B$  of  $F$  can be done by testing if  $(n+1)D$  monomials are in  $B$  in at most  $O(nD^2)$  elementary operations which can be decreased to  $O(nD)$  if we use a hash table.  $\square$

Now that we have seen how to construct  $B$  and  $F$ , the complexity of Algorithm 15 is treated in the following proposition.

**Proposition 4.10.** *Given the DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of an ideal, one can compute all the multiplication matrices in  $O((d_{\max} - d_{\min})n^\omega D^\omega)$  (or in  $O((d_{\max} - d_{\min})n^\omega d^{\omega n})$ ) arithmetic operations in  $\mathbb{K}$  where  $d_{\max}$  (resp.  $d_{\min}$ ) is the maximal (resp. the minimal) degree of all the polynomials in  $\mathcal{G}_{>\text{drl}}$ .*

*Proof.* Algorithm 15 computes all the multiplication matrices incrementally degree by degree. The frontier  $F$  can be written as the union of disjoint sets  $F_\delta = \{t \in F \mid \deg(t) = \delta\}$  so that we define  $s_\delta := \#F_\delta$  and  $S_\delta := s_{d_{\min}} + \dots + s_\delta$ . The cost of the loop at Line 4 is, at each step, given by the complexity of computing the reduced row echelon form of  $M$ . In degree  $\delta$  the shape of the matrix  $M$  is depicted on Figure 4.2 where  $\text{Id}(S_{\delta-1})$  is the  $S_{\delta-1} \times S_{\delta-1}$  identity matrix,  $\mathbf{0}(S_{\delta-1})$  is the  $S_{\delta-1} \times s_\delta$  zero matrix,  $T$  is a  $s_\delta \times s_\delta$  upper triangular matrix and  $B, C, D$  are dense matrices of respective size  $s_\delta \times S_{\delta-1}$ ,  $s_\delta \times D$ ,  $S_{\delta-1} \times D$ .

$$M = \begin{array}{c|ccc|ccc|ccc} & \multicolumn{4}{c}{t \in F_\delta} & \multicolumn{4}{c}{t \in F_{\delta-1} \cup \dots \cup F_{d_{\min}}} & \multicolumn{4}{c}{t \in B} \\ \hline 1 & \star & \cdots & \star & \star & \cdots & \star & \star & \cdots & \star \\ 0 & 1 & \cdots & \star & \star & \cdots & \star & \star & \cdots & \star \\ \vdots & \mathbf{T} & \ddots & \vdots & \vdots & \mathbf{B} & \vdots & \vdots & \mathbf{C} & \vdots \\ 0 & 0 & \cdots & 1 & \star & \cdots & \star & \star & \cdots & \star \\ \hline 0 & 0 & \cdots & 0 & 1 & \cdots & 0 & \star & \cdots & \star \\ \vdots & \mathbf{0}(S_{\delta-1}, s_\delta) & \vdots & \mathbf{Id}(S_{\delta-1}) & \vdots & \ddots & \vdots & \vdots & \mathbf{D} & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 & \star & \cdots & \star \end{array}$$

Figure 4.2: Shape of the matrix  $M$  of Algorithm 15.

Consequently the reduced row echelon form of  $M$  can be obtained from the following formula:

$$\text{ReducedRowEchelonForm}(M) = \left[ \begin{array}{c|cccc} \mathbf{Id}(S_\delta) & T^{-1}(C - BD) \\ \hline & D \end{array} \right].$$

Since  $s_\delta \leq S_\delta \leq S_{d_{\max}} \leq nD$  we can bound the complexity of computing the reduced row echelon form of  $M$  by  $O(n^\omega D^\omega)$ . From Lemma 4.9, the costs of the construction of  $B$  and  $F$  are negligible in comparison to the cost of loop in Line 4 which therefore gives the complexity of Algorithm 15:  $O((d_{\max} - d_{\min})n^\omega D^\omega)$  arithmetic operations. Since  $D \leq d^n$ , this complexity can be written as  $O((d_{\max} - d_{\min})n^\omega d^{\omega n})$ .  $\square$

### 4.3.2 Complexity for regular systems

Regular systems form an important family of polynomial systems. Actually, as shown in Section 2.4.1 the complexity of computing a Gröbner basis of a regular system is well understood. Since the property of being regular is a generic property this is also the typical behavior of polynomial systems. For regular systems we can bound accurately the values of  $d_{\max}$  which is the maximal degree of  $\mathcal{G}_{>\text{drl}}$  and we can prove the first main result of this chapter.

**Theorem 4.11.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\}$  be a polynomial system generating an ideal admitting a triangular set for LEX Gröbner basis. Assume that  $(f_1, \dots, f_n)$  is a regular sequence of polynomials whose degrees are uniformly bounded by a fixed integer  $d$  i.e.  $\deg(f_i) \leq d$  for*

$i = 1, \dots, n$ . The PoSSo problem (Problem 4.1) can be solved using a deterministic algorithm in  $O(ne^{\omega n} d^{\omega n} + (n^{\omega+1} + n \log_2 D) D^{\omega})$  arithmetic operations in  $\mathbb{K}$ .

*Proof.* For regular systems  $d_{\max}$  can be bounded by the Macaulay bound [Laz83, BFSY05] and Corollary 2.76:  $d_{\max} \leq \sum_{i=1}^n (\deg(f_i) - 1) + 1 \leq n(d - 1) + 1$ . Given the system  $\mathcal{S}$  the complexity of computing the DRL Gröbner basis of  $\langle \mathcal{S} \rangle$  is bounded by [BFSY05], Theorem 2.77 and Theorem 2.83:

$$O\left(n \binom{n + d_{\max}}{n}^{\omega}\right) = O\left(n \binom{nd + 1}{n}^{\omega}\right) = O(ne^{\omega n} d^{\omega n})$$

arithmetic operations (see proof of Corollary 2.91 for more details about the approximation of the binomial).

From this DRL Gröbner basis, according to Proposition 4.10, the multiplication matrices  $T_1, \dots, T_n$  can be computed in  $O(n^{\omega+1} D^{\omega})$  arithmetic operations.

Finally, from  $T_1, \dots, T_n$  and the DRL Gröbner basis, thanks to Proposition 4.6 the LEX Gröbner basis of  $\langle \mathcal{S} \rangle$  can be computed by a deterministic algorithm in  $O(n \log_2(D) D^{\omega})$  arithmetic operations. Since,  $F_4$  [Fau99],  $F_5$  [Fau02], Algorithm 13 and Algorithm 15 are deterministic algorithms this finishes the proof.  $\square$

Since the beginning of the chapter, the first Gröbner basis is assumed to be the DRL Gröbner basis. One can notice that the algorithms presented until now do not use this assumption. Hence, the result of the previous theorem can be extend to the case where the first Gröbner basis is the WDRL Gröbner basis. Indeed, the asymptotic value of  $d_{\max}$  is unchanged in comparison to the DRL case.

**Corollary 4.12.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a polynomial system generating an ideal admitting a triangular set for LEX Gröbner basis. Assume that the ring  $\mathbb{K}[x_1, \dots, x_n]$  is equipped with a weighted degree with weights system  $(w_1, \dots, w_n)$ . If  $(f_1, \dots, f_n)$  is a regular sequence of polynomials whose degrees are uniformly bounded by a fixed integer  $d$  i.e.  $\deg(f_i) \leq d$  for  $i = 1, \dots, n$ . The PoSSo problem (Problem 4.1) can be solved using a deterministic algorithm in  $O\left(n \left(\frac{e^n d^n}{\prod_{i=1}^n w_i}\right)^{\omega} + (n^{\omega+1} + n \log_2 D) D^{\omega}\right)$  arithmetic operations in  $\mathbb{K}$ .*

This corollary implies that if the degree of the equations are fixed then fast change algorithm can be used in Algorithm 12.

Among regular systems, there are generic systems (Definition 2.34). Let  $d_i = \deg(f_i) = d$  for all  $i = 1, \dots, n$ . Since the Bézout's bound (Corollary 2.76) allows to bound the number of solutions  $D$  by  $\prod_{i=1}^n d_i = d^n$  and since this bound is generically reached, we have generically that  $D = d^n$  and we get the following corollary.

**Corollary 4.13.** *Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a generic polynomial system generating an ideal  $\mathcal{I} = \langle \mathcal{S} \rangle$  of degree  $D$ . If  $\mathcal{I}$  admits a triangular set for LEX Gröbner basis and if the degree of each polynomial in  $\mathcal{S}$  is uniformly bounded by a fixed integer  $d$  then there exists a deterministic algorithm which solves the PoSSo problem in  $\tilde{O}(e^{\omega n} D^{\omega})$  arithmetic operations.*

In the next section, we study a first step towards the generalization of Theorem 4.11 to polynomial systems with equations of non fixed degree. More precisely, we are going to discuss what happens if one polynomial have a non fixed degree i.e. its degree depends on a parameter (for instance the number of variables). In this case, Theorem 4.11 does not apply but we present other arguments in order to obtain a similar complexity results for computing  $\mathcal{G}_{>\text{lex}}$  given  $\mathcal{G}_{>\text{drl}}$  and new ideas for its generalization.

## 4.4 A worst case ultimately not so bad

We consider the following pathological case:  $\deg(f_1) = \dots = \deg(f_{n-1}) = 2$  and  $\deg(f_n) = 2^n$ . Then,  $D = 2^{2n-1}$ ,  $d_{\min} = 2$  and  $d_{\max} = 2^n + n - 1$ . In this context, the complexity of computing  $\mathcal{G}_{>\text{lex}}$  given  $\mathcal{G}_{>\text{drl}}$  seems to be in  $O(\log_2^\omega(D)D^{\omega+\frac{1}{2}})$  arithmetic operations. However, we will show that an adaptation of Algorithm 15 allows to decrease this complexity.

In [MS03], Moreno-Socias studied the basis of the residue class ring  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ , w.r.t. the DRL ordering, for generic ideals. In particular, he shows that when the smallest variable  $x_n$  is in abscissa any section of the stairs of  $\mathcal{I}$  has steps of height one and of depth two. That is to say, for any variable  $x_i$  with  $i < n$  and for all instantiations of the others variables  $(\{x_1, \dots, x_{n-1}\} \setminus \{x_i\})$  the associated section of the stairs of  $\mathcal{I}$  has the shape in Figure 2.1. This shape is summarized in Theorem 2.38.

Following the notations of Theorem 2.38, in our case we have  $d_{\max} = \delta + 1$ ,  $\delta^* = n - 1$ ,  $\delta = 2^n + n - 2$ ,  $\sigma = n - 1$  and  $\mu = 2^n - n$ . We can note that in this particular case,  $\mu$  is very large, which implies that a large part of the monomials of the form  $\epsilon_i x_j$  are actually in  $B$ . We will show that in Algorithm 15 instead of computing the loop in Line 4 for  $d = d_{\min}, \dots, d_{\max}$  we can perform it only on the restricted subset  $d = d_{\min}, \dots, \sigma(n-1) + 1, \mu + 1, \dots, d_{\max}$ . By consequence, the complexity of computing  $\mathcal{G}_{>\text{lex}}$  given  $\mathcal{G}_{>\text{drl}}$  will be in  $O((d_{\max} - \mu + \sigma(n-1) - d_{\min})n^\omega D^\omega) = O(\log_2^{\omega+2}(D)D^\omega)$  with  $d_{\max} - \mu + \sigma(n-1) - d_{\min} = n^2 - 2 \sim \log_2^2(D)$ .

**Lemma 4.14.** *Given the normal form of all monomials in  $F$  of degree less than or equal to  $\sigma(n-1) + 1$  we can compute all the normal forms of all monomials in  $F$  of degree less than or equal to  $\mu$  in  $O(nD^2)$  arithmetic operations.*

Suppose that we know the normal form of the monomials of the forms  $\epsilon_i x_j$  of degree less than  $\mu$  which are not divisible by  $x_n$ . From these normal forms, the idea of the proof is to show that the normal form of all the monomials of the form  $\epsilon_i x_j$  of degree less than  $\mu$  and of degree  $\alpha_n > 0$  in  $x_n$  is given by  $x_n^{\alpha_n} \text{NF}_{>\text{drl}}(t)$  where  $\text{NF}_{>\text{drl}}(t)$  is assumed to be known.

*Proof.* Let  $t \in F$  of degree less than or equal to  $\mu$ . First, assume that  $x_n$  does not divide  $t$ . As  $\mathcal{I}$  is zero-dimensional, there exist  $\eta_1, \dots, \eta_{n-1} \in \mathbb{N}$  such that  $x_i^{\eta_i}$  is a leading term of a polynomial in  $\mathcal{G}_{>\text{drl}}$ . Moreover, from Theorem 2.38,  $\eta_i \leq \bar{\sigma}$ . Hence, for all  $\epsilon \in \tilde{B}_0$ ,  $\deg(\epsilon) \leq \sigma(n-1)$ . The monomials in  $F$  not divisible by  $x_n$  are all of the form  $x_i \epsilon$  with  $i = 1, \dots, n-1$  and  $\epsilon \in \tilde{B}_0$ . Thus  $\deg(t) \leq \sigma(n-1) + 1$  and by hypothesis, its normal form is known.

Suppose now that  $x_n$  divides  $t$  and  $t$  is of type III of Proposition 2.68. We can write  $t = x_n^\alpha t'$  where  $\alpha \in \mathbb{N}^*$  such that  $x_n \nmid t'$ . From Theorem 2.38 item (d),  $t'$  is a leading term of a polynomial in  $\langle \mathcal{G}_{>\text{drl}} \rangle$ . Moreover,  $t \in F$  so  $t = x_i \epsilon$  with  $\epsilon \in B$ . Suppose that  $i = n$  hence,  $\frac{t}{x_n} = \epsilon = x_n^{\alpha-1} t' \in \langle \mathcal{G}_{>\text{drl}} \rangle$  which is impossible. Thus,  $i \neq n$  and we have,  $t' = \frac{t}{x_n^\alpha} = x_i \epsilon' \in F$  with  $\epsilon' = \frac{\epsilon}{x_n^\alpha} \in B$ . Therefore, from the first part of this proof,  $\text{NF}_{>\text{drl}}(t') = \sum_{i=1}^s \alpha_i \epsilon_i$ ,  $\alpha_i \in \mathbb{K}$  is known. Finally,  $\text{NF}_{>\text{drl}}(t) = \sum_{i=1}^s \alpha_i \text{NF}_{>\text{drl}}(x_n^\alpha \epsilon_i)$  with  $\deg(x_n^\alpha \epsilon_i) \leq \mu$ . Let  $k_i$  be such that  $x_n^{k_i} | \epsilon_i$  and  $x_n^{k_i+1} \nmid \epsilon_i$  as  $\tilde{B}_{k_i} = \tilde{B}_{k_i+\alpha}$  then  $x_n^\alpha \epsilon_i \in B$  and  $\text{NF}_{>\text{drl}}(t) = \sum_{i=1}^s \alpha_i x_n^\alpha \epsilon_i$ .

By consequence, computing the normal form of  $t$  can be done in less than  $D$  arithmetic operations. As usual, we can bound the size of  $F$  by  $nD$  which finishes the proof.  $\square$

One can notice that Algorithm 14 and its deterministic version take as input only the multiplication matrix by the smallest variable. Thus in the proof of Theorem 4.11 we did not fully take advantage of this particularity. Hence, the next section is devoted to study if this

matrix can be computed more efficiently than computing all the multiplication matrices. By studying the structure of the basis of the  $\mathbb{K}$ -vector space  $\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$  we will show that, up to a linear change of variables,  $T_n$  can be deduced from  $\mathcal{G}_{>\text{drl}}$ . In the previous results, the algorithm restricting the order of magnitude of the degrees of the equations is Algorithm 15 to compute the multiplication matrices. Since, we need only  $T_n$  which can be computed very efficiently, the impact of such a result is that there exists a Las Vegas algorithm extending the result of Theorem 4.11 to polynomial systems whose equations have non fixed degree.

## 4.5 Polynomial equations with non-fixed degree: the wild case

In this section, in order to obtain our second main result, we consider *initial* and *generic* ideals. To compute the multiplication matrix  $T_n$  we need to compute the normal forms of all monomials  $\epsilon_i x_n$  for  $i = 1, \dots, D$  with  $\epsilon_i \in B$ . From Proposition 2.68 a monomial of the form  $\epsilon_i x_n$  can be either in  $B$  or in  $E_{>\text{drl}}(\mathcal{I})$  or in  $\text{in}_{>\text{drl}}(\mathcal{I}) \setminus E_{>\text{drl}}(\mathcal{I})$ . As previously shown, the difficulty to compute  $T_n$  lies in the computation of the normal forms of monomials  $\epsilon_i x_n$  that are in  $\text{in}_{>\text{drl}}(\mathcal{I}) \setminus E_{>\text{drl}}(\mathcal{I})$ . In this section, thanks to the study of the stairs, *i.e.*  $B$ , of generic ideals by Moreno-Socias, see Section 2.1.3, we first show that for generic ideals (Definition 2.34), all monomials of the form  $\epsilon_i x_n$  are in  $B$  or in  $E_{>\text{drl}}(\mathcal{I})$ . Hence, the multiplication matrix  $T_n$  can be computed very efficiently. Then, we show that, up to a linear change of variables, this result can be extended to any ideal. According to these results, we finally propose an algorithm for solving the PoSSo problem whose complexity allows to obtain the second main result of this chapter.

### 4.5.1 Reading directly $T_n$ from the Gröbner basis

In the sequel, the arithmetic operations will be the addition or the multiplication of two operands in  $\mathbb{K}$  that are different from  $\pm 1$  and 0. In particular we do not consider the change of sign as an arithmetic operation.

**Proposition 4.15.** *Let  $\mathcal{I}$  be a generic ideal. Let  $t$  be a monomial in  $E_{>\text{drl}}(\mathcal{I})$  *i.e.* a leading term of a polynomial in the DRL Gröbner basis of  $\mathcal{I}$ . If  $x_n$  divides  $t$  then for all  $k \in \{1, \dots, n-1\}$ ,  $\frac{x_k t}{x_n} \in \text{in}_{>\text{drl}}(\mathcal{I})$ .*

*Proof.* This result is deduced from the shape of the stairs of  $\mathcal{I}$  (see Figure 2.1 for a representation in dimension 2). Let  $t = x_1^{\alpha_1} \dots x_n^{\alpha_n}$  be a leading term of a polynomial in  $\mathcal{G}_{>\text{drl}}$  divisible by  $x_n$  *i.e.*  $\alpha_n > 0$  and  $m = x_1^{\alpha_1} \dots x_{n-1}^{\alpha_{n-1}}$ . We use the same notations as in Theorem 2.38.

From Theorem 2.38 item (d), since  $t \in E_{>\text{drl}}(\mathcal{I})$  and  $\alpha_n > 0$  we have  $\alpha_n > \mu$  and  $\alpha_n \not\equiv \delta \pmod{2}$ . Then, from Theorem 2.38 item (c),  $\deg(m)$  is the maximal degree reached by the monomials in  $\tilde{B}_{\alpha_{n-1}}$ . Thus  $x_k m \notin \tilde{B}_{\alpha_{n-1}}$  for all  $k \in \{1, \dots, n-1\}$ . As a consequence, for all  $k \in \{1, \dots, n-1\}$  we have  $\frac{x_k t}{x_n} \in \text{in}_{>\text{drl}}(\mathcal{I})$ .  $\square$

Consequently, from the previous proposition, we obtain the following result.

**Theorem 4.16.** *Given  $\mathcal{G}_{>\text{drl}}$  the DRL Gröbner basis of a generic ideal  $\mathcal{I}$ , the multiplication matrix  $T_n$  can be read from  $\mathcal{G}_{>\text{drl}}$  with no arithmetic operation.*

*Proof.* Suppose that there exists  $i \in \{1, \dots, D\}$  such that  $t = x_n \epsilon_i$  is of type (III). Hence,  $t = m \text{LT}_{\text{drl}}(g)$  for some  $g \in \mathcal{G}_{>\text{drl}}$  and  $\deg(m) > 1$  with  $x_n \nmid m$  (otherwise  $\epsilon_i \notin B$ ). Then, there exists  $k \in \{1, \dots, n-1\}$  such that  $x_k \mid m$ . By consequence, from Proposition 4.15,

we have  $\epsilon_i = \frac{m}{x_k} \cdot \frac{x_k \text{LT}_{>\text{drl}}(g)}{x_n} \in \text{in}_{>\text{drl}}(\mathcal{I})$  which yields a contradiction. Thus, all monomials  $t = x_n \epsilon_i$  are either in  $B$  or in  $\text{E}_{>\text{drl}}(\mathcal{I})$  and their normal forms are known and given either by  $t$  (if  $t \in B$ ) or by changing the sign of some polynomial  $g \in \mathcal{G}_{>\text{drl}}$  and removing its leading term. Note that by using a linked list representation (for instance), removing the leading term of a polynomial does not require arithmetic operation.  $\square$

Thanks to the previous theorem, Algorithm 14 or its deterministic version can be used to compute the LEX Gröbner basis of a generic ideal:

**Corollary 4.17.** *Let  $\mathcal{I}$  be a generic ideal in Shape Position. From the DRL Gröbner basis  $\mathcal{G}_{>\text{drl}}$  of  $\mathcal{I}$ , its LEX Gröbner basis  $\mathcal{G}_{>\text{lex}}$  can be computed in  $O(\log_2(D)(D^\omega + nD \log_2(D) \log_2 \log_2(D)))$  arithmetic operations with a probabilistic algorithm or  $O(n \log_2(D) D^\omega)$  ( $+O\left(D \log_2 \frac{q}{p}\right)$  if  $\mathbb{K} = \mathbb{F}_{p^k}$ ) arithmetic operations with a deterministic algorithm.*

However, polynomial systems coming from applications are usually not generic. Nevertheless, this difficulty can be bypassed by applying a linear change of variables. By studying the structure of the *generic initial ideal* of  $\mathcal{I}$  (Definition 2.43) – that is to say, the initial ideal of  $g \cdot \mathcal{I}$  for a generic choice of  $g$  – we will show that the results of Proposition 4.15 and Theorem 4.16 can be generalized to non-generic ideals, up to a random linear change of variables. Indeed, in [Gal73] Galligo shows that for the characteristic zero fields, the generic initial ideal of any homogeneous ideal satisfies a more general property, Property 2.47, than Proposition 4.15. Later, Pardue [Par94] extends this result to fields of positive characteristic, see Theorem 2.49 and Corollary 2.50.

Nevertheless, systems coming from applications are usually not homogeneous and results of Theorem 2.49 and Corollary 2.50 do not apply directly. Let  $\mathcal{I} = \langle f_1, \dots, f_n \rangle$  be an affine ideal i.e.  $f_1, \dots, f_n$  are affine polynomials. In the next proposition we highlight an homogeneous ideal having the same initial ideal than  $\mathcal{I}$ . This allows to extend the result of Theorem 2.49 and Corollary 2.50 to affine ideals.

**Proposition 4.18.** *Let  $\mathcal{I} = \langle f_1, \dots, f_s \rangle$  be an affine ideal. If  $(f_1, \dots, f_s)$  is a regular sequence, then there exists a Zariski open set  $U_a \subset \mathbf{GL}(\mathbb{K}, n)$  such that for all  $g \in U_a$ ,  $\text{E}_{>\text{drl}}(g \cdot \mathcal{I}) = \text{E}_{>\text{drl}}(\mathbf{Gin}(\mathcal{I}^h))$ .*

*Proof.* Let  $f$  be a polynomial. We denote by  $f^h$  the homogeneous component of highest degree of  $f$  and  $f^a = f - f^h$ . Let  $t \in \text{in}_{>\text{drl}}(\mathcal{I})$ , there exists  $f \in \mathcal{I}$  such that  $\text{LT}_{>\text{drl}}(f) = t$ . Since,  $f \in \mathcal{I}$  and  $(f_1^h, \dots, f_s^h)$  is assumed to be a regular sequence then there exist  $h_1, \dots, h_s \in \mathbb{K}[x_1, \dots, x_n]$  such that  $f = \sum_{i=1}^s h_i f_i = \sum_{i=1}^s h_i f_i^h + \sum_{i=1}^s h_i f_i^a$  with  $\deg(h_i f_i) \leq \deg(f)$  for all  $i \in \{1, \dots, s\}$  and there exists  $j \in \{1, \dots, s\}$  such that  $\deg(h_j f_j) = \deg(f)$ . By consequence,  $0 \neq \sum_{i=1}^s h_i f_i^h \in \mathcal{I}^h$  where  $\mathcal{I}^h$  is the ideal generated by  $\{f_1^h, \dots, f_s^h\}$  and  $\text{LT}_{>\text{drl}}(f) = \text{LT}_{>\text{drl}}(\sum_{i=1}^s h_i f_i^h)$ . Thus,  $\text{in}_{>\text{drl}}(\mathcal{I}) \subset \text{in}_{>\text{drl}}(\mathcal{I}^h)$ . It is straightforward that  $\text{in}_{>\text{drl}}(\mathcal{I}^h) \subset \text{in}_{>\text{drl}}(\mathcal{I})$  hence  $\text{in}_{>\text{drl}}(\mathcal{I}^h) = \text{in}_{>\text{drl}}(\mathcal{I})$ .

For all  $g \in \mathbf{GL}(\mathbb{K}, n)$ , since  $g$  is invertible the sequence  $(g \cdot f_1, \dots, g \cdot f_s)$  is also regular. Indeed, if there exists  $i \in \{1, \dots, s\}$  such that  $g \cdot f_i$  is a divisor of zero in the quotient ring  $\mathbb{K}[x_1, \dots, x_n] / \langle g \cdot f_1, \dots, g \cdot f_i \rangle$  then  $f_i$  is a divisor of zero in  $\mathbb{K}[x_1, \dots, x_n] / \langle f_1, \dots, f_i \rangle$ . Hence,

$$\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \text{in}_{>\text{drl}}((g \cdot \mathcal{I})^h).$$

Moreover,  $g$  is a linear change of variables thus it preserves the degree. Hence, for all  $f \in \mathcal{I}$ , we have  $(g \cdot f)^h = g \cdot f^h$ . Finally, let  $U_a$  be a Zariski open subset of  $\mathbf{GL}(\mathbb{K}, n)$  such that for

all  $g \in U_a$ , we have the equality  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}^h) = \mathbf{Gin}(\mathcal{I}^h)$ . Thus, for all  $g \in U_a$ , we then have  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \text{in}_{>\text{drl}}((g \cdot \mathcal{I})^h) = \text{in}_{>\text{drl}}(g \cdot \mathcal{I}^h) = \mathbf{Gin}(\mathcal{I}^h)$ .  $\square$

Hence, from the previous proposition, for a random linear change of variables  $g \in \mathbf{GL}(\mathbb{K}, n)$  we have  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \mathbf{Gin}(\mathcal{I}^h)$ . Thus from Corollary 2.50, for all generators  $m$  of the monomial ideal  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I})$  (i.e.  $m$  is a leading term of a polynomial in the DRL Gröbner basis of  $g \cdot \mathcal{I}$ ) if  $x_n^t$  divides  $m$  and  $x_n^{t+1}$  does not divide  $m$  then for all  $j < n$  we have  $\frac{x_j}{x_n}m \in \text{in}_{>\text{drl}}(g \cdot \mathcal{I})$  if  $t \not\equiv 0 \pmod{p}$ . Therefore, in the same way as for generic ideals, the multiplication matrix  $T_n$  of  $g \cdot \mathcal{I}$  can be read from its DRL Gröbner basis. Moreover, the Shape Lemma (Lemma 2.57) states that radical ideals have, up to a generic linear change of variables, a LEX Gröbner basis in *Shape Position*. Hence, one can compute very efficiently the multiplication matrix  $T_n$  and then use Algorithm 14 to compute the LEX Gröbner basis of  $g \cdot \mathcal{I}$ . This is summarized in the following corollary.

**Corollary 4.19.** *Let  $\mathbb{K}$  be an infinite field of characteristic  $p \geq 0$ . Let  $\mathcal{I}$  be an ideal of  $\mathbb{K}[x_1, \dots, x_n]$ . There exists a Zariski open subset  $U$  of  $\mathbf{GL}(\mathbb{K}, n)$  such that for all  $g \in U$ , the arithmetic complexity of computing the multiplication matrix by  $x_n$  of  $g \cdot \mathcal{I}$  given its DRL Gröbner basis can be done without arithmetic operation. If  $p > 0$  this is true only if  $\deg_{x_n}(m) \not\equiv 0 \pmod{p}$  for all  $m \in E_{>\text{drl}}(g \cdot \mathcal{I})$ . Consequently, under the same hypotheses and if  $\mathcal{I}$  is a radical ideal, the complexity of computing the LEX Gröbner basis of  $g \cdot \mathcal{I}$  given its DRL Gröbner basis can be bounded by  $O(\log_2(D)(D^\omega + nD \log_2(D) \log_2 \log_2(D)))$  arithmetic operations.*

Following this result, we propose another algorithm for polynomial systems solving.

#### 4.5.2 Another algorithm for polynomial systems solving

Let  $\mathcal{S} \subset \mathbb{K}[x_1, \dots, x_n]$  be a polynomial system generating a radical ideal denoted  $\mathcal{I}$ . For any  $g \in \mathbf{GL}(\mathbb{K}, n)$ , from the solutions of  $g \cdot \mathcal{I}$  one can easily recover the solutions of  $\mathcal{I}$ . Let  $U$  be the Zariski open subset of  $\mathbf{GL}(\mathbb{K}, n)$  such that for all  $g \in U$ ,  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \mathbf{Gin}(\mathcal{I}^h)$ . If  $g$  is chosen in  $U$  then the multiplication matrix  $T_n$  can be computed very efficiently. Indeed, from Section 4.5.1 all monomials of the form  $\epsilon_i x_n$  for  $i = 1, \dots, D$  are in  $B$  or in  $E_{>\text{drl}}(g \cdot \mathcal{I})$  and their normal are easily known. Moreover, from the Shape Lemma (Lemma 2.57), there exists  $U'$  a Zariski open subset of  $\mathbf{GL}(\mathbb{K}, n)$  such that for all  $g \in U'$  the ideal  $g \cdot \mathcal{I}$  admits a LEX Gröbner basis in *Shape Position*. If  $g$  is also chosen in  $U'$  then we can use Algorithm 14 or its deterministic version to compute the LEX Gröbner basis of  $g \cdot \mathcal{I}$ . Hence, we propose in Algorithm 16 a Las Vegas algorithm to solve the PoSSo problem. A Las Vegas algorithm is a randomized algorithm whose output (which can be *fail*) is always correct. The end of this section is devoted to evaluate its complexity and its probability of success i.e. when the algorithm does not return *fail*.

Algorithm 16 succeeds if the three following conditions are satisfied

1.  $g \in \mathbf{GL}(\mathbb{K}, n)$  is chosen in a non empty Zariski open set  $U'$  such that for all  $g \in U'$ ,  $g \cdot \mathcal{I}$  has a LEX Gröbner basis in *Shape Position*;
2.  $g \in \mathbf{GL}(\mathbb{K}, n)$  is chosen in a non empty Zariski open set  $U$  such that for all  $g \in U$ ,  $\text{in}_{>\text{drl}}(g \cdot \mathcal{I}) = \mathbf{Gin}(\mathcal{I}^h)$ ;
3.  $p = 0$  or  $p > 0$  and for all  $m \in E_{>\text{drl}}(g \cdot \mathcal{I})$ ,  $\deg_{x_n}(m) \not\equiv 0 \pmod{p}$ .

**Algorithm 16:** Another algorithm for PoSSo.

---

**Input** : A polynomial system  $\mathcal{S} \subset \mathbb{K}[x_1, \dots, x_n]$  generating a radical ideal.  
**Output**:  $g$  in  $\mathbf{GL}(\mathbb{K}, n)$  and the LEX Gröbner basis of  $\langle g \cdot \mathcal{S} \rangle$  or *fail*.

- 1 Choose randomly  $g$  in  $\mathbf{GL}(\mathbb{K}, n)$ ;
- 2 Compute  $\mathcal{G}_{>\text{drl}}$  the DRL Gröbner basis of  $g \cdot \mathcal{S}$ ;
- 3 **if**  $T_n$  can be read from  $\mathcal{G}_{>\text{drl}}$  **then**
- 4     Extract  $T_n$  from  $\mathcal{G}_{>\text{drl}}$ ;
- 5     From  $T_n$  and  $\mathcal{G}_{>\text{drl}}$  compute  $\mathcal{G}_{>\text{lex}}$  using Algorithm 14;
- 6     **if** Algorithm 14 succeeds **then return**  $g$  and  $\mathcal{G}_{>\text{lex}}$ ;
- 7     **else return fail**;
- 8 **else return fail**;

---

The existence of the non empty Zariski open subset  $U'$  is proven in [GM89] (see proof of Lemma 2.57). Conditions (1) and (2) are satisfied if  $g \in U \cap U'$ . Since,  $U$  and  $U'$  are open and dense,  $U \cap U'$  is also a non empty Zarisky open set.

**Probability of success of Algorithm 16**

Assume that one can randomly choose an element in  $\mathbf{GL}(\mathbb{K}, n)$  with  $\mathbb{K} = \mathbb{C}$  or  $\mathbb{R}$ . Then, in that case the probability of choosing an element in  $U \cap U'$  i.e. that the condition (1) and (2) be satisfied is 1. By consequence, the probability of success of Algorithm 16, if  $\mathbb{K} = \mathbb{C}, \mathbb{R}$  is 1.

Contrary to finite sets, there is no effective way of randomly choosing an element in  $\mathbb{C}$  or  $\mathbb{R}$ . Moreover, usually the coefficient field of the polynomials is the field of rational numbers or a finite field. Assume that  $\mathbb{K} = \mathbb{F}_q$  or  $\mathbb{K} = \mathbb{Q}$  and we randomly choose in a finite subset of  $\mathbb{Q}$  of size  $q$ . The Schwartz-Zippel lemma [Sch80, Zip79] allows to bound the probability that the conditions (1) and (2) do not be satisfied by  $\frac{d}{q}$  where  $d$  is the degree of the polynomial defining  $U \cap U'$ . Thus, in order to bound this failure probability we need to estimate the degree of the polynomials defining  $U$  and  $U'$ .

**Construction of  $U'$ .** From proof of the Shape Lemma (Lemma 2.57) the polynomial  $P_{U'}$  defining  $U'$  is constructed as the determinant of a Vandermonde matrix associated to  $D$  indeterminates. Then, the degree of  $P_{U'}$  is  $\frac{D(D-1)}{2}$ .

**Construction of  $U$ .** From proof of Theorem 2.42, the Zariski open subset  $U$  is constructed as the intersection of Zariski open subsets  $U_1, \dots, U_\delta$  of  $\mathbf{GL}(\mathbb{K}, n)$  where  $\delta$  is the maximum degree of the generators of  $\mathbf{Gin}(\mathcal{I}^h)$ . Let  $d$  be a fixed degree. Let  $\mathbb{K}[x_1, \dots, x_n]_d = R_d$  be the set of homogeneous polynomials of degree  $d$  of  $\mathbb{K}[x_1, \dots, x_n]$ . Let  $\{f_1, \dots, f_{t_d}\} \subset R_d$  be a vector basis of  $\mathcal{I}_d^h = \mathcal{I}^h \cap R_d$ . Let  $\mathbf{g} = (\mathbf{g}_{i,j})$  be a  $(n \times n)$  matrix of unknowns and let  $M$  be a matrix representation of the map  $\mathcal{I}_d^h \rightarrow \mathbf{g} \cdot \mathcal{I}_d^h$  defined as follows:

$$M = (M_{i,j}) = \begin{array}{ccc|c} m_1 & \cdots & m_N & \\ \hline \star & \cdots & \star & \mathbf{g} \cdot f_1 \\ \vdots & \ddots & \vdots & \vdots \\ \star & \cdots & \star & \mathbf{g} \cdot f_{t_d} \end{array}$$

where  $M_{i,j}$  is the coefficient of  $m_j$  in  $\mathbf{g} \cdot f_i$  and  $\{m_1, \dots, m_N\}$  is the set of monomials in  $R_d$ . In [BS87b, Eis95] (proof of Theorem 2.42), the polynomial  $P_{U_d}$  defining  $U_d$  is constructed as a particular minor of size  $t_d$  of  $M$ . Since each coefficient in  $M$  is a polynomial in  $\mathbb{K}[\mathbf{g}_{1,1}, \dots, \mathbf{g}_{n,n}]$  of degree  $d$ , the degree of  $P_{U_d}$  is  $d \cdot t_d$ . Finally, since  $U_d$  is open and dense for all  $d = 1, \dots, \delta$  we deduce that  $U = \cap_{i=1}^{\delta} U_d$  is a non empty Zariski open set whose defining polynomial,  $P_U$ , is of degree  $\sum_{d=1}^{\delta} d \cdot t_d \leq \delta \sum_{i=1}^{\delta} t_d$ . Moreover, we have  $t_d = \dim_{\mathbb{K}}(\mathcal{I}_d^h)$ . Since  $\delta$  is the minimal degree such that  $\dim_{\mathbb{K}}(R_d/\mathcal{I}_d^h) = 0$  for any  $d \geq \delta$  and since  $(f_1, \dots, f_n)$  is assumed to be a regular sequence we have that  $\text{HS}_{\mathbb{K}[x_1, \dots, x_n]/\mathcal{I}^h}(1) = \sum_{d=1}^{\delta} (\dim_{\mathbb{K}}(R_d) - \dim_{\mathbb{K}}(\mathcal{I}_d^h)) = D$ , see Corollary 2.76. Hence,  $\sum_{d=1}^{\delta} \dim_{\mathbb{K}}(\mathcal{I}_d^h) = \sum_{d=1}^{\delta} \dim_{\mathbb{K}}(R_d) - D = \binom{n+\delta}{n} - D$ .

For ideals generated by a regular sequence  $(f_1, \dots, f_n)$ , thanks to the Macaulay's bound (Corollary 2.76),  $\delta$  can be bounded by  $\sum_{i=1}^n (\deg(f_i) - 1) + 1$ . Note that the Macaulay's bound gives also a bound on  $\deg_{x_n}(m)$  for all  $m \in E_{>\text{drl}}(g \cdot \mathcal{I})$ . To conclude, the probability that conditions (1) and (2) be satisfied is greater than

$$1 - \frac{1}{q} \left( \frac{D(D-1)}{2} + \left( \sum_{i=1}^n (\deg(f_i) - 1) + 1 \right) \left( \binom{\sum_{i=1}^n \deg(f_i) + 1}{n} - D \right) \right)$$

and if  $p = 0$  or  $p > \sum_{i=1}^n (\deg(f_i) - 1) + 1$  then condition (3) is satisfied.

### Complexity of Algorithm 16

As previously mentioned, the matrix  $T_n$  can be read from  $\mathcal{G}_{>\text{drl}}$  (test in Line 3 of Algorithm 16) if all the monomials of the form  $\epsilon_i x_n$  are either in  $B$  or in  $E_{>\text{drl}}(\langle \mathcal{G}_{>\text{drl}} \rangle)$ . Let  $F_n = \{\epsilon_i x_n \mid i = 1, \dots, D\}$ , the test in Line 3 is equivalent to test if  $F_n \subset B \cup E_{>\text{drl}}(\langle \mathcal{G}_{>\text{drl}} \rangle)$ . Since  $F_n$  contains exactly  $D$  monomials and  $B \cup E_{>\text{drl}}(\langle \mathcal{G}_{>\text{drl}} \rangle)$  contains at most  $(n+1)D$  monomials; in a similar way as in Lemma 4.9 testing if  $F_n \subset B \cup E_{>\text{drl}}(\langle \mathcal{G}_{>\text{drl}} \rangle)$  can be done in at most  $O(nD^2)$  elementary operations which can be decreased to  $O(D)$  elementary operations if we use a hash table. Hence, the cost of computing  $B$ ,  $F_n$  (see Lemma 4.9) and the test in Line 3 of Algorithm 16 are negligible in comparison to the complexity of Algorithm 14. Hence, the complexity of Algorithm 16 is given by the complexity of  $F_5$  algorithm to compute the DRL Gröbner basis of  $g \cdot \mathcal{I}$  and the complexity of Algorithm 14 to compute the LEX Gröbner basis of  $g \cdot \mathcal{I}$ . From [Laz83], the complexities of computing the DRL Gröbner basis of  $g \cdot \mathcal{I}$  or  $\mathcal{I}$  are the same. Since it is straightforward to see that the number of solutions of these two ideals are also the same we obtain the second main result of the chapter.

**Theorem 4.20.** *Let  $\mathbb{K}$  be a field of characteristic zero or a finite field  $\mathbb{F}_q$  of sufficiently large characteristic  $p$ . Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a polynomial system generating a radical ideal  $\mathcal{I} = \langle \mathcal{S} \rangle$  of degree  $D$ . If the sequence  $(f_1, \dots, f_n)$  is a regular sequence such that the degree of each polynomial is uniformly bounded by a fixed or non fixed parameter  $d$  then there exists a Las Vegas algorithm which solves the PoSSo problem in  $O(ne^{\omega n} d^{\omega n} + \log_2(D)(D^{\omega} + nD \log_2(D) \log_2 \log_2(D)))$  (respectively  $O(d^{\omega n} + \log_2(D)(D^{\omega} + D \log_2(D) \log_2 \log_2(D)))$ ) arithmetic operations if  $n \rightarrow \infty$  (respectively  $n$  is fixed).*

As previously mentioned, the Bézout's bound allows to bound the number of solutions  $D$  by the product of the degrees of the input equations. Since this bound is generically reached we get the following corollary.

**Corollary 4.21.** *Let  $\mathbb{K}$  be a field of characteristic zero or a finite field  $\mathbb{F}_q$  of sufficiently large characteristic  $p$ . Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_n]$  be a generic polynomial system generating a radical ideal. If the degree of each polynomial in  $\mathcal{S}$  is equal to a non fixed parameter  $d$  then there exists a Las Vegas algorithm which solves the PoSSo problem in  $\tilde{O}(D^\omega)$  (respectively  $\tilde{O}(e^{\omega n} D^\omega)$ ) arithmetic operations if  $n$  is fixed (respectively if  $n \rightarrow \infty$ ).*

**Remark 4.22.** *Note that Algorithm 16 cannot be used to solve polynomial systems admitting a polynomial change of variables. Indeed, applying the linear change of variables will break the quasi-homogeneous structure of these systems. However, assume the multiplication matrix  $T_n$  associated to the WDRL Gröbner basis can be computed with no arithmetic operations. The complexity of Corollary 4.12 can then be extended to systems whose equations have non fixed degree.*

## 4.6 Impact of Algorithm 16 on the practical solving of the PoSSo problem in the worst case

In this section we discuss the impact of Algorithm 16 on the practical resolution of the PoSSo problem. Note that Algorithm 14 to compute the LEX Gröbner basis given the multiplication matrix  $T_n$  is of theoretical interest. Indeed, although in theory  $\omega$  is bounded by 2.3727 in practice in our knowledge the best implementation of the matrix product uses Strassen algorithm [Str69]. For instance this algorithm is implanted in MAGMA [BCP97] or in LINBOX [DGG<sup>+</sup>02]. Thus, in practice  $\omega = \log_2(7) \sim 2.8073$ .

As a consequence, in practice the sparse version of Faugère and Mou [FM11] (see Chapter 2) is much more efficient than the fast version using dense matrix multiplication. Hence, in the following experiments we use the *sparse* version of change of ordering. In Table 4.1, we give the time to compute the LEX Gröbner basis using the usual algorithm (Algorithm 11) and Algorithm 16. This time is divided into three steps, the first is the time to compute the DRL Gröbner basis using  $F_5$  algorithm, the second is the time to compute the multiplication matrix  $T_n$  and the last part is the time to compute the LEX Gröbner basis given  $T_n$  using the algorithm in [FM11]. Since, this algorithm takes advantage of the sparsity of the matrix  $T_n$  we also give its density. We also give the number of normal forms to compute (*i.e.* the number of terms of the form  $\epsilon_i x_n$  that are not in  $B$  or in  $E_{>\text{drl}}(\mathcal{I})$  (or in  $E_{>\text{drl}}(g \cdot \mathcal{I})$ ).

The experiments are performed on a worst case for our algorithm in the sense that the system in input is already a DRL Gröbner basis. Thus, while the usual algorithm does not have to compute the DRL Gröbner basis, our algorithm needs to compute the DRL Gröbner basis of  $g \cdot \mathcal{I}$ . The system in input is of the form  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{F}_{65521}[x_1, \dots, x_n]$  with  $\text{LT}_{>\text{drl}}(f_i) = x_i^2$ . Hence, the monomials in the basis  $B$  are all the monomials of degree at most one in each variable. The degree of the ideal  $D$  is then  $2^n$ . The monomials  $\epsilon_i x_n$  that are not in  $B$  or in  $E_{>\text{drl}}(\langle \mathcal{S} \rangle)$  are of the form  $x_n^2 m$  where  $m$  is a monomial in  $x_1, \dots, x_{n-1}$  of total degree greater than zero and linear in each variable. By consequence, using the usual algorithm we have to compute  $2^{n-1} - 1$  normal forms to compute only  $T_n$ .

One can note that in the usual algorithm the bottleneck of the resolution of the PoSSo problem is the change of ordering due to the construction of the multiplication matrix  $T_n$ . Since our algorithm allows to compute very efficiently the matrix  $T_n$  (for instance for  $n = 11$ , 0 seconds in comparison to 7544 seconds for the usual algorithm), the most time consuming step becomes the computation of the DRL Gröbner basis. However, the total running time of

| n  | D     | Algorithm | First GB | Build $T_n$ | # NF  | Density | Compute $h_1, \dots, h_n$ | Total PoSSo |
|----|-------|-----------|----------|-------------|-------|---------|---------------------------|-------------|
| 7  | 128   | usual     | 0s       | 0s          | 63    | 34.20%  | 0s                        | 0s          |
|    |       | This work | 0s       | 0s          | 0     | 26.57%  | 0s                        | 0s          |
| 9  | 512   | usual     | 0s       | 13s         | 255   | 32.81%  | 0s                        | 13s         |
|    |       | This work | 0s       | 0s          | 0     | 23.68%  | 0s                        | 0s          |
| 11 | 2048  | usual     | 0s       | 7521s       | 1023  | 31.93%  | 23s                       | 7544s       |
|    |       | This work | 5s       | 0s          | 0     | 21.53%  | 0s                        | 5s          |
| 13 | 8192  | usual     | 0s       | > 2 days    | 4095  |         |                           | > 2 days    |
|    |       | This work | 157s     | 2s          | 0     | 19.86%  | 26s                       | 185s        |
| 15 | 32768 | usual     | 0s       | > 2 days    | 16383 |         |                           | > 2 days    |
|    |       | This work | 5786s    | 46s         | 0     | 18.52%  | 1886s                     | 7718s       |
| 16 | 65536 | usual     | 0s       | > 2 days    | 32767 |         |                           | > 2 days    |
|    |       | This work | 38067s   | 195s        | 0     | 18.33%  | 14297s                    | 52559s      |

Table 4.1: A worst case example: comparison of the usual algorithm for solving the PoSSo problem and Algorithm 16, the proposed algorithm. Computation with FGb on a 3.47 GHz Intel Xeon X5677 CPU.

our algorithm is far less than that of the usual algorithm. For instance, for  $n = 13$  the PoSSo problem can now be solved in approximately three minutes whereas we could not solve this instance of the PoSSo problem using the usual algorithm.

Moreover, using Algorithm 16 the density of the matrix  $T_n$  is decreased (which implies that the running time of Faugère and Mou algorithms is also decreased). This can be explained by the fact that the dense columns of the matrix  $T_n$  come from monomials of the form  $x_n \epsilon_i$  that are not in  $B$  *i.e.* in the frontier. Since Algorithm 16 allows to ensure that the monomials  $x_n \epsilon_i$  are either in  $B$  or in  $E_{>\text{drl}}(g \cdot \mathcal{I})$  then the number of dense columns in  $T_n$  is potentially decreased.

## Part II

# Algebraic Cryptanalysis of the Elliptic Curves Discrete Logarithm



## CHAPTER 5

# Elliptic curves

---

### Contents

---

|            |                                                                                            |            |
|------------|--------------------------------------------------------------------------------------------|------------|
| <b>5.1</b> | <b>Definitions</b>                                                                         | <b>104</b> |
| <b>5.2</b> | <b>Elliptic curves representations</b>                                                     | <b>106</b> |
| 5.2.1      | Short Weierstrass form                                                                     | 106        |
| 5.2.2      | Twisted Jacobi intersection curves                                                         | 107        |
| 5.2.3      | Twisted Edwards curves                                                                     | 108        |
| 5.2.4      | Universal Edwards model of elliptic curves                                                 | 109        |
| <b>5.3</b> | <b>Discrete logarithm problem and generic algorithms</b>                                   | <b>111</b> |
| 5.3.1      | Pohlig Hellman reduction                                                                   | 111        |
| 5.3.2      | Baby step giant step                                                                       | 112        |
| 5.3.3      | Pollard $\rho$ method                                                                      | 113        |
| <b>5.4</b> | <b>Semaev summation polynomials</b>                                                        | <b>114</b> |
| 5.4.1      | Computing summation polynomials                                                            | 115        |
| 5.4.2      | Twisted Jacobi intersection curves                                                         | 116        |
| 5.4.3      | Twisted Edwards curves                                                                     | 117        |
| 5.4.4      | Universal Edwards model of elliptic curves                                                 | 117        |
| <b>5.5</b> | <b>Gaudry's index calculus attack for ECDLP solving</b>                                    | <b>117</b> |
| 5.5.1      | Presentation of the algorithm                                                              | 117        |
| 5.5.2      | Complexity analysis                                                                        | 121        |
| 5.5.3      | Balancing relation search and linear algebra using the <i>double large prime variation</i> | 122        |
| 5.5.4      | Variant " $n - 1$ "                                                                        | 123        |
| 5.5.5      | Diem's variant of the index calculus attack                                                | 124        |
| <b>5.6</b> | <b>Using symmetries to improve the ECDLP solving</b>                                       | <b>124</b> |
| 5.6.1      | Solving the point decomposition problem                                                    | 125        |
| 5.6.2      | Computation of summation polynomials                                                       | 127        |

---

In this chapter we give definitions and properties about elliptic curves needed in the two following chapters. Moreover, we briefly present generic algorithms to solve the discrete logarithm problem. We recall that an algorithm to solve the DLP is said to be generic if it does not take advantage of the structure of the group. Finally, we recall the principle of index calculus attack to solve the elliptic curve discrete logarithm. For a more thorough reading on elliptic curves and algorithms to solve the discrete logarithm problem see for instance [CF05, CP05, Coh93, Sil09].

## 5.1 Definitions

First, we give a representation of an elliptic curve as a projective variety.

**Definition 5.1** (Projective space). *Let  $\mathbb{K}$  be a field. The projective space of dimension  $n$  on  $\mathbb{K}$  is denoted  $\mathbb{P}_{\mathbb{K}}^n$  and is defined as the quotient  $\mathbb{K}^{n+1}/\{0\}$  with the equivalence relation  $\sim$  defined by  $(x_0, \dots, x_n) \sim (y_0, \dots, y_n)$  if there exists  $0 \neq \lambda \in \mathbb{K}$  such that  $x_i = \lambda y_i$  for  $i = 0, \dots, n$ . We denote by  $(x_0 : \dots : x_n)$  the equivalence class of  $(x_0, \dots, x_n)$  which is also called a projective point.*

A projective variety is defined as a subset of  $\mathbb{P}_{\mathbb{K}}^n$  in a similar way that affine variety (Definition 2.3) are defined as a subset of  $\mathbb{K}^n$ .

**Definition 5.2** (Projective variety). *Let  $f_1, \dots, f_s \in \mathbb{K}[x_0, \dots, x_n]$  be homogeneous polynomials. The projective variety associated to  $f_1, \dots, f_s$  is the set*

$$\mathbf{V}_{\mathbb{P}_{\mathbb{K}}^n}(f_1, \dots, f_s) = \{(a_0 : \dots : a_n) \in \mathbb{P}_{\mathbb{K}}^n \mid f_i(a_0, \dots, a_n) = 0 \text{ for } i = 1, \dots, s\}.$$

A variety  $V$  is irreducible if there are no  $V_1, V_2 \subset V$  such that  $V_1, V_2 \neq V$  and  $V = V_1 \cup V_2$ .

An elliptic curve is a curve (i.e. a projective variety of dimension one) of genus one that admits a rational point. Since, we do not need a formal definition of the genus we stick to a more basic equivalent definition. For a formal definition of the genus and thus of elliptic curves see for instance [Sil09, chapter 2].

**Definition 5.3** (Elliptic curve). *An elliptic curve defined over  $\mathbb{K}$  is an irreducible projective variety of dimension one with no singularity and which is birationally equivalent to a projective Weierstrass curve defined by*

$$E_w^p : y^2z + a_1xyz + a_3yz^2 - x^3 - a_2x^2z - a_4xz^2 - a_6z^3 = 0 \quad (5.1)$$

where  $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$  are such that  $E_w^p$  has no singularity.

An elliptic curve defined over  $\mathbb{K}$  by the equation  $E_w^p$  has no singularity if the system  $\mathcal{S} = \left\{ E_w^p = 0, \frac{\partial E_w^p}{\partial x} = 0, \frac{\partial E_w^p}{\partial y} = 0, \frac{\partial E_w^p}{\partial z} = 0 \right\} = \{ y^2z + a_1xyz + a_3yz^2 - x^3 - a_2x^2z - a_4xz^2 - a_6z^3 = 0, a_1yz - 3x^2 - 2a_2xz - a_4z^2 = 0, 2yz + a_1xz + a_3z^2 = 0, y^2 + a_1xy + 2a_3yz - a_2x^2 - 2a_4xz - 3a_6z^2 = 0 \}$  does not have any solution in  $\overline{\mathbb{K}}$ .

Let two projective varieties  $V \subset \mathbb{P}_{\mathbb{K}}^n$  and  $W \subset \mathbb{P}_{\mathbb{K}}^m$ . Let us recall that a rational map from  $V$  to  $W$  is a  $m$ -tuple  $(r_1, \dots, r_m)$  of rational fractions with  $n$  variables. That is to say,  $r_i$  can be written as  $\frac{f_i}{g_i}$  with  $f_i, g_i \in \mathbb{K}[x_1, \dots, x_n]$ . The two projective varieties  $V$  and  $W$  are *birationally equivalent* if there exist two rational maps  $\mathfrak{R}_1$  and  $\mathfrak{R}_2$  respectively from  $V$  to  $W$  and from  $W$  to  $V$  such that  $\mathfrak{R}_1 \circ \mathfrak{R}_2$  (respectively  $\mathfrak{R}_2 \circ \mathfrak{R}_1$ ) is equivalent to the identity map on  $W$  (respectively  $V$ ).

An elliptic curve is a projective variety however it also admits an affine representation. That is to say, an elliptic curve can be seen as an affine variety to which we add a point at infinity. The affine equation corresponding to the homogeneous equation (5.1) is obtained by taking  $z = 1$ :

$$E_w : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0. \quad (5.2)$$

Note that,  $E_w^p$  corresponds to the homogenization (Definition 2.62) of  $E_w$ . The condition of non-singularity thus becomes that the system  $\left\{ E_w = 0, \frac{\partial E_w}{\partial x} = 0, \frac{\partial E_w}{\partial y} = 0 \right\}$  admits no solution in  $\overline{\mathbb{K}}$ .

In the affine case, we omit the projective points with  $z = 0$  called points at infinity. One can note that  $E_w^p$  has a unique point at infinity  $(0 : 1 : 0)$ . Hence, the whole elliptic curve defined as an affine variety is the set of points given by  $E_w$  plus the point at infinity  $P_\infty$  corresponding to the projective point  $(0 : 1 : 0)$ .

One of the properties of elliptic curves that makes them very useful is that they are naturally equipped with a group law. In particular, an elliptic curve is an abelian variety *i.e.* the set of points of an elliptic curve forms an abelian group.

The group law  $\oplus$  of an elliptic curve may be seen geometrically. We present one possible construction of the group law. In particular, this construction sets the point at infinity as the neutral element.

Let  $P$  and  $Q$  be two points of the curve. Let  $R \neq P, Q$  be the third point of  $E_w$  intersecting the curve and the line through  $P$  and  $Q$  (or the tangent of the curve at  $P$  if  $P = Q$ ). The point  $P \oplus Q$  is constructed as the third point intersecting the curve and the line through  $R$  and  $P_\infty$ . Let  $P$  be a point of  $E_w$  the inverse of  $P$  denoted  $\ominus P$ , is the third point intersecting the curve and the line through  $P$  and  $P_\infty$ . Hence, the point at infinity is the neutral element, denoted  $\mathcal{O}$  of the group law of  $E_w$ . This geometric construction of the group law is depicted on Figure 5.1.

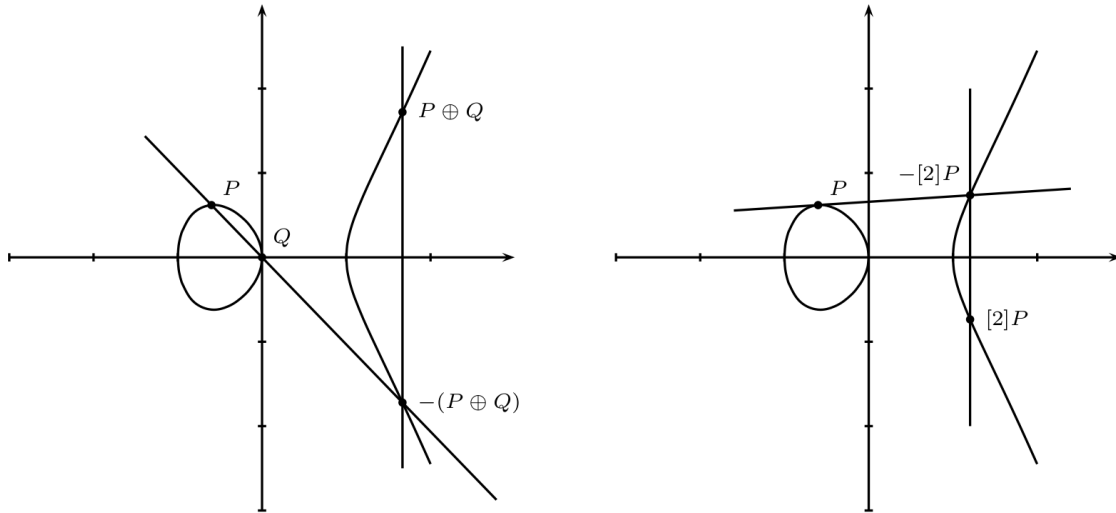


Figure 5.1: Group law of elliptic curves.

This geometric construction can be translated into algebraic equations. More precisely, the group law of  $E_w$  is given by rational fractions in terms of the coordinates of the points we want to sum. Let  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$  the coordinates of the point  $P \oplus Q$  is given by

- $P = (x_1, y_1)$  if  $Q = \mathcal{O}$ ;
- $Q = (x_2, y_2)$  if  $P = \mathcal{O}$ ;
- $(x_R, -(\lambda + a_1)x_R - \mu - a_3)$  where  $x_R = \lambda(\lambda + a_1) - x_1 - x_2 - a_2$ ,  $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$  and  $\mu = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$  if  $P \neq Q$  and  $P, Q \neq \mathcal{O}$ ;

- $(x_R, -(\lambda + a_1)x_R - \mu - a_3)$  where  $x_R = \lambda(\lambda + a_1) - 2x_1 - a_2$ ,  $\lambda = \frac{3x_1^2 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}$  and  $\mu = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}$  if  $P = Q$  and  $P \neq \mathcal{O}$ .

Moreover, the point  $\ominus P$  is given by  $(x_1, -y_1 - a_1x_1 - a_3)$ .

The efficiency of the arithmetic of elliptic curves is a central issue in cryptography. While any elliptic curves can be represented by a Weierstrass equation some of them share common properties that allow to choose another form of equation. In particular, considering this new representation can speed up the arithmetic of elliptic curves. A lot of publications about this subject have been done, see for instance [CC86, Mon87, Sma01, DIK06, Duq07, BL07, BBJ+08, FNW10]. For a more exhaustive listing of elliptic curve representations and their corresponding arithmetic see [BL].

## 5.2 Elliptic curves representations

In this section we study some elliptic curve representations. First, we present the well known short Weierstrass form. Then, we focus on three representations of elliptic curves that all have a two-torsion point (*i.e.* a point of order two) with a simple action. We will show in Chapter 6 and Chapter 7 that the action of their two-torsion induces some symmetries when solving the elliptic curve discrete logarithm problem.

### 5.2.1 Short Weierstrass form

#### Characteristic greater than 3

If the field  $\mathbb{K}$  is of characteristic different from 2 and 3 then the change of coordinates  $Y = y + \frac{a_1x + a_3}{2}$  and  $X = x + \frac{b_2}{12}$  where  $b_2 = a_1^2 + 4a_2$  allows to write the Weierstrass equation (5.2) as follows

$$E_w : Y^2 = X^3 + aX + b \quad (5.3)$$

where  $a = \frac{24b_4 - b_2^2}{48}$  and  $b = b_2^3 + 216b_6 - 36b_2b_4$  with  $b_4 = a_1a_3 + 2a_4$  and  $b_6 = a_3^2 + 4a_6$ . Thus any elliptic curve defined over a field of characteristic greater than 3 can be represented thanks to an equation of the form (5.3). Note that the curve defined by equation (5.3) is non singular and hence elliptic if and only if  $-16(4a^3 + 27b) \neq 0$ . See for instance [Coh93, chapter 7] for more details.

The group law of such a curve is simply obtained by replacing  $a_1 = 0, a_2 = 0, a_3 = 0, a_4 = a$  and  $a_6 = b$  in the formula for the group law of elliptic curve defined by the general Weierstrass equation.

#### Characteristic 2

If the field  $\mathbb{K}$  is of characteristic two then  $-16(4a^3 + 27b) = 0$  whatever  $a$  and  $b$  are. Hence, an elliptic curve defined over a field of characteristic two cannot be represented with an equation of the form (5.3). In that case there does not exist a unique short Weierstrass equation to represent all binary elliptic curves. However, there exists a short Weierstrass form for any ordinary elliptic curves and another short Weierstrass form for any supersingular elliptic curve.

**Definition 5.4** (Supersingular/ordinary elliptic curve). *Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_{p^k}$  with  $p$  a prime. Let  $E(\mathbb{K})$  be the set of points of  $E$  defined over  $\mathbb{K}$ . The*

curve  $E$  is called supersingular if the set of points  $P \in E(\overline{\mathbb{F}_{p^k}})$  such that  $[p]P = \mathcal{O}$  is reduced to the neutral element  $\mathcal{O}$ . The notation  $[n]P$  means adding  $n$  times  $P$ . Otherwise, the curve is called ordinary.

Thus, if  $a_1 = 0$  and  $\mathbb{K}$  is a field of characteristic two then  $a_3 \neq 0$  (otherwise the curve is singular) and the Weierstrass equation (5.2) can be written as

$$E_{w,s} : Y^2 + a_3Y = X^3 + \alpha X + \beta \quad (5.4)$$

where  $Y = y$ ,  $X = x + a_2$ ,  $\alpha = a_2^2 + a_4$  and  $\beta = a_2(a_2^2 + a_4) + a_6$ . In that case, if  $P = (x, y)$  is a point of  $E$  then  $\ominus P = (x, y + a_3)$ . Hence,  $P$  is a point of order two if and only if  $P = \ominus P$  which is impossible since  $a_3 \neq 0$ . Consequently, any supersingular binary elliptic curves can be represented by a short Weierstrass equation of the form (5.4).

On the contrary, if  $a_1 \neq 0$  then elliptic curves given by the Weierstrass equation (5.2) have at least one two-torsion point. Hence, the curve is ordinary. In that case the change of variables  $y = a_1^3Y + \frac{a_3^2 + a_1^2a_4}{a_1^3}$  and  $x = a_1^2X + \frac{a_3}{a_1}$  followed by a division by  $a_1^6$  gets the following short Weierstrass equation

$$E_{w,o} : Y^2 + XY = X^3 + \alpha X^2 + \beta \quad (5.5)$$

where  $\alpha = \frac{a_1a_2 + a_3}{a_1^3}$  and  $\beta = \frac{a_1^5a_6 + a_1^5a_3a_4 + a_1^4(a_2a_3^2 + a_4^2) + a_1^3a_3^3 + a_3^4}{a_1^{12}}$ . Hence, any ordinary binary elliptic curve can be represented thanks to a short Weierstrass equation of the form (5.5) when  $\beta \neq 0$ . Indeed, the curve defined by equation (5.5) is non singular when  $\beta \neq 0$ .

Due to the efficiency of pairings computation for supersingular curves, for similar size of field  $\mathbb{K}$  the elliptic curve discrete logarithm problem is easier to solve when considering supersingular curves, see for instance [CF05, chapter 24]. More precisely, thanks to pairing computations the discrete logarithm problem in  $E(\mathbb{F}_q)$  can be reduced to solve the discrete logarithm problem in  $\mathbb{F}_{q^k}$  for some integer  $k > 0$ . In general,  $k$  is large enough so that the complexity of the elliptic curve discrete logarithm is not affected. However, for supersingular curves,  $k$  is particularly small. More precisely, in characteristic two we have  $k \leq 4$ . Moreover, recent breakthrough algorithms [Jou13a, GGMZ13, Jou13b, BGJT13] have considerably improved the complexity of solving the discrete logarithm problem in finite fields of small characteristic which can now be solved in quasi-polynomial time [BGJT13]. Consequently, since the elliptic curve discrete logarithm of binary supersingular elliptic curve can be solved efficiently, in the following of this thesis we consider only binary ordinary elliptic curves defined by the short Weierstrass equation (5.5).

**Remark 5.5.** For fields of characteristic three there also exist short Weierstrass equations, see for instance [CF05, p. 274].

### 5.2.2 Twisted Jacobi intersection curves

This form of elliptic curves was introduced in 2010 in [FNW10]. It is a generalization of Jacobi intersection curves (which are the intersection of two quadratic surfaces defined in a 3-dimensional space) proposed by D.V. and G.V. Chudnovsky in [CC86].

The twisted Jacobi intersection curves are defined over a non binary field  $\mathbb{K}$  by

$$E_{a,b} : \begin{cases} ax^2 + y^2 = 1 \\ bx^2 + z^2 = 1 \end{cases}$$

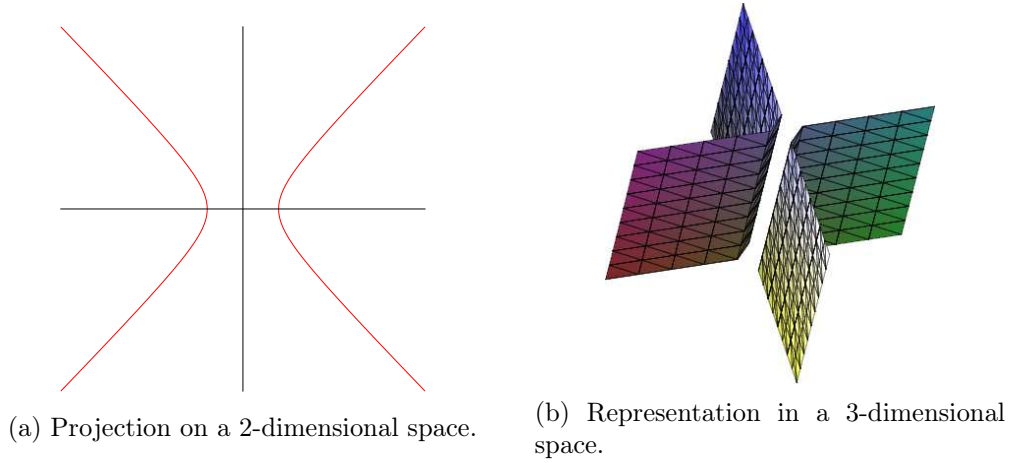


Figure 5.2: Jacobi intersection curve over the real numbers.

where  $a, b \in \mathbb{K}$ ,  $a, b \neq 0$  and  $a \neq b$ . If  $a = 1$ ,  $E_{1,b}$  is a Jacobi intersection curve.

The family of twisted Jacobi intersection curves contains all curves having three rational 2-torsion points. These three 2-torsion points are  $T_2 = (0, 1, -1)$ ,  $(0, -1, 1)$  and  $(0, -1, -1)$ . The neutral element is  $\mathcal{O} = (0, 1, 1)$  and the negative of a point  $P = (x, y, z) \in E_{a,b}(\mathbb{K})$  is given by  $\ominus P = (-x, y, z)$ . Adding one of the 2-torsion point to  $P$  gives respectively the points  $(-x, y, -z)$ ,  $(-x, -y, z)$  and  $(x, -y, -z)$ . The group law is given by

$$(x_1, y_1, z_1) \oplus (x_2, y_2, z_2) = \left( \frac{x_1 y_2 z_2 + x_2 y_1 z_1}{y_2^2 + a z_1^2 x_2^2}, \frac{y_1 y_2 - a x_1 z_1 x_2 z_2}{y_2^2 + a z_1^2 x_2^2}, \frac{z_1 z_2 - b x_1 y_1 x_2 y_2}{y_2^2 + a z_1^2 x_2^2} \right).$$

Jacobi intersection curves can have zero, four or eight 4-torsion points :

- $\left( \pm \frac{1}{\sqrt{b}}, \pm \sqrt{\frac{b-a}{b}}, 0 \right)$ , if  $a \neq 1$  non square or  $a = 1$  and  $-1$  non square and  $b$  and  $b-a$  are squares in  $\mathbb{K}$ .
- $\left( \pm \frac{1}{\sqrt{a}}, 0, \pm \sqrt{\frac{a-b}{a}} \right)$ , if  $b \neq 1$  non square or  $b = 1$  and  $-1$  non square and  $a$  and  $a-b$  are squares in  $\mathbb{K}$ .
- $\left( \pm \frac{1}{\sqrt{b}}, \pm \sqrt{\frac{b-a}{b}}, 0 \right), \left( \pm \frac{1}{\sqrt{a}}, 0, \pm \sqrt{\frac{a-b}{a}} \right)$ , if  $a, b, -1$  and  $a-b$  are squares in  $\mathbb{K}$ .

### 5.2.3 Twisted Edwards curves

This family of elliptic curves was introduced in 2008 in cryptography [BBJ<sup>+</sup>08]. Similarly to twisted Jacobi intersection curves, this representation of elliptic curves is a generalization of the representation proposed by Edwards in [Edw07]. These curves were deeply studied by the cryptology community, especially by Bernstein and Lange [BL07], for their efficient arithmetic. In [BBJ<sup>+</sup>08] the authors show that the family of twisted Edwards curves is isomorphic to the family of Montgomery curves [Mon87]. In particular these curves always have a rational 2-torsion point  $T_2 = (0, -1)$  (and a rational 4-torsion point for Edwards curves). A twisted

Edwards curve is defined over a field  $\mathbb{K}$  of characteristic  $> 2$  by

$$E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2 \quad (5.6)$$

where  $a, d \neq 0$  and  $a \neq d$ . If  $a = 1$ ,  $E_{1,d}$  is an (untwisted) Edwards curve.

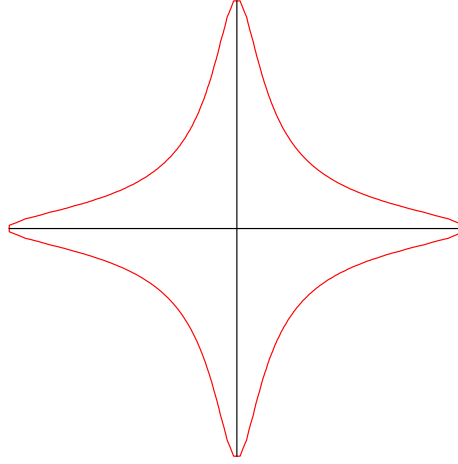


Figure 5.3: Edwards curve over the real numbers.

The group law of a twisted Edwards curve is given by

$$(x_1, y_1) \oplus (x_2, y_2) = \left( \frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

with neutral element  $\mathcal{O} = (0, 1)$ . The opposite of a point  $P = (x, y) \in E_{a,d}(\mathbb{K})$  is  $\ominus P = (-x, y)$ , and adding  $T_2$  to  $P$  gives  $P + T_2 = (-x, -y)$ . Therefore, the symmetries can be interpreted in terms of the group law. If  $a$  is a square in  $\mathbb{K}$  then a twisted Edwards curve has two 4-torsion points  $T_4 = (a^{-\frac{1}{2}}, 0)$  or  $(-a^{-\frac{1}{2}}, 0)$ .

### 5.2.4 Universal Edwards model of elliptic curves

In [DF12], the authors introduce a new representation of elliptic curves that they call Edwards model of elliptic curves. The advantage of such a representation is that it has good reduction modulo two. That is to say, the group law and the equation defining the curve are independent from the characteristic of the field the curve is defined over. Hence, in the whole of this thesis we call this representation the universal Edwards model.

Let  $\mathbb{K}$  be a field of any characteristic. An universal Edwards model of elliptic curves is defined by

$$E : 1 + x^2 + y^2 + x^2y^2 = \lambda xy \quad (5.7)$$

where  $\lambda = 4t + \frac{1}{t} \in \mathbb{K}^*$  with  $t \in \mathbb{K}^*$ .

For fields of characteristic different from two, a curve in universal Edwards model is isomorphic to an Edwards curves. In characteristic two, an elliptic curve in universal Edwards model is birationally equivalent to an elliptic curve in Weierstrass representation defined by  $y^2 + xy = x^3 + \frac{1}{\lambda^4}$ .

The group law on universal Edwards model is defined by  $(x_1, y_1) \oplus (x_2, y_2) = (x_3, y_3)$  where

$$\begin{aligned} (x_3, y_3) &= (\phi_x(x_1, y_1, x_2, y_2), \phi_y(x_1, y_1, x_2, y_2)) \\ &= \left( \frac{x_1 + y_1 x_2 y_2 - 2t(y_1 + x_1 x_2 y_2)}{y_2 + x_1 y_1 x_2 - 2t(x_2 + x_1 y_1 y_2)}, \frac{x_1 x_2 + y_1 y_2 - 2t(x_1 y_2 + y_1 x_2)}{1 + x_1 y_1 x_2 y_2 - 2t(x_1 y_1 + x_2 y_2)} \right). \end{aligned}$$

The neutral element of this group law is the point  $\mathcal{O} = (2t, 1)$  and the opposite of a point  $P = (x, y)$  is given by  $\ominus P = \left(x, \frac{1}{y}\right)$ . An elliptic curve in universal Edwards model has three two-torsion points  $T_{2,1} = \left(\frac{1}{2t}, 1\right)$ ,  $T_{2,2} = (-2t, -1)$  and  $T_{2,3} = \left(-\frac{1}{2t}, -1\right)$ . Let  $P = (x, y)$  be a point of  $E$ . The addition of  $P$  with a two-torsion point gives:

$$\begin{aligned} P \oplus T_{2,1} &= \left(\frac{1}{x}, \frac{1}{y}\right) \\ P \oplus T_{2,2} &= (-x, -y) \\ P \oplus T_{2,3} &= \left(-\frac{1}{x}, -\frac{1}{y}\right). \end{aligned}$$

In characteristic two, only one two-torsion point remains. Actually, the neutral element  $\mathcal{O}$  and the two-torsion point  $T_{2,2}$  coincide with the neutral element of the binary model  $\mathcal{O} = (0, 1)$ . The two torsion points  $T_{2,1}$  and  $T_{2,3}$  coincide with the unique two torsion point being the point at infinity  $P_\infty$ .

So that the two-torsion point remains a rational point in characteristic two we consider the following change of variables:  $x \mapsto \frac{1}{X}$  and  $y \mapsto Y$ . The equation defining the curve does not change. Indeed, equation (5.7) implies that

$$1 + \frac{1}{X^2} + Y^2 + \frac{Y^2}{X^2} = \lambda \frac{Y}{X}. \quad (5.8)$$

Multiplying both sides of equation (5.8) by  $X^2$  we obtain

$$\mathcal{E} : X^2 + 1 + X^2 Y^2 + Y^2 = \lambda X Y. \quad (5.9)$$

The neutral element becomes  $\mathcal{O} = \left(\frac{1}{2t}, 1\right)$  and the three two-torsion points are  $T_{2,1} = (2t, 1)$ ,  $T_{2,2} = \left(-\frac{1}{2t}, -1\right)$  and  $T_{2,3} = (-2t, -1)$ .

In characteristic two the neutral element and the two-torsion point  $T_{2,2}$  coincide with the point at infinity  $P_\infty$  which is now the neutral element of the binary model. The two-torsion points  $T_{2,1}$  and  $T_{2,3}$  coincide with the unique two-torsion point  $T_2 = (0, 1)$ .

The group law on  $\mathcal{E}$  is defined by  $(X_1, Y_1) \oplus (X_2, Y_2) = (X_3, Y_3)$  where

$$\begin{aligned} (X_3, Y_3) &= \left( \frac{1}{\phi_x\left(\frac{1}{X_1}, Y_1, \frac{1}{X_2}, Y_2\right)}, \phi_y\left(\frac{1}{X_1}, Y_1, \frac{1}{X_2}, Y_2\right) \right) \\ &= \left( \frac{Y_2 X_1 X_2 + Y_1 - 2t(X_1 + X_2 Y_1 Y_2)}{X_2 + X_1 Y_1 Y_2 - 2t(Y_1 X_1 X_2 + Y_2)}, \frac{1 + X_1 X_2 Y_1 Y_2 - 2t(X_2 Y_2 + X_1 Y_1)}{X_1 X_2 + Y_1 Y_2 - 2t(X_2 Y_1 + X_1 Y_2)} \right). \end{aligned}$$

The opposite of a point  $P = (X, Y)$  is still given by  $\ominus P = \left(X, \frac{1}{Y}\right)$ . Indeed,  $(X, Y) \oplus \left(X, \frac{1}{Y}\right) =$

$(\frac{1}{2t}, 1) = \mathcal{O}$ . Moreover, we have

$$\begin{aligned} P \oplus \mathcal{O} &= P \\ P \oplus T_{2,1} &= \left( \frac{1}{X}, \frac{1}{Y} \right) \\ P \oplus T_{2,2} &= (-X, -Y) \\ P \oplus T_{2,3} &= \left( -\frac{1}{X}, -\frac{1}{Y} \right). \end{aligned}$$

A curve in universal Edward model can have 4-torsion points. Indeed, assume that  $\mathbb{K} = \mathbb{F}_{q^n}$ . If  $q^n \equiv 1 \pmod{4}$  then  $-1$  is a square in  $\mathbb{K}$ . In that case the curve has four 4-torsion points  $T_{4,1} = (0, \sqrt{-1})$ ,  $T_{4,2} = (0, -\sqrt{-1})$ ,  $T_{4,3} = (\sqrt{-1}, 0)$  and  $T_{4,4} = (-\sqrt{-1}, 0)$ .

### 5.3 Discrete logarithm problem and generic algorithms

Given a finite cyclic group  $\mathbb{G} = \langle g \rangle$  of group law  $\oplus$ , the discrete logarithm problem, DLP for short, is defined as follows: given  $h \in \mathbb{G}$  to find an integer  $x$  such that

$$h = [x]g = \underbrace{g \oplus \cdots \oplus g}_{x \text{ times}}.$$

Given any finite cyclic group  $\mathbb{G}$  there exist algorithms solving the DLP without knowing any structure of  $\mathbb{G}$ . These algorithms are called generic algorithms. In this section, we briefly present some of these algorithms.

#### 5.3.1 Pohlig Hellman reduction

Given a finite cyclic group  $(\mathbb{G}, \oplus) = (\langle g \rangle, \oplus)$  of order  $n$  and  $h \in \mathbb{G}$ , the DLP consists of finding  $x \pmod{n}$  satisfying  $h = [x]g$ . Let  $n = \prod p_i^{\alpha_i}$  be the prime factorization of  $n$ . The idea of the Pohlig-Hellman reduction, see [PH78] or [MVOV10, chapter 3], is to reduce the computation of the logarithm in  $\mathbb{G}$  to logarithms in  $\mathbb{Z}/p_i\mathbb{Z}$ . To this end, one computes  $x_i = x \pmod{p_i^{\alpha_i}}$  for each prime  $p_i$  such that  $\alpha_i > 0$ . Then, finding  $x$  is reduced to use the *Chinese Remainder Theorem*. To find  $x_i$  one computes  $\epsilon_0, \dots, \epsilon_{\alpha_i-1}$ , the digits of the  $p_i$ -ary representation of  $x_i$  i.e.  $x_i = \sum_{j=0}^{\alpha_i-1} \epsilon_j p_i^j$ . Assume we know  $\epsilon_0, \dots, \epsilon_j$ . To compute  $\epsilon_{j+1}$  one proceeds as follows. Let  $\mathbf{g} = \left[ \frac{n}{p_i} \right] g$ , note that since  $g$  is a generator of  $\mathbb{G}$  the order of  $\mathbf{g}$  is  $p_i$ . Let  $\gamma = \left[ \epsilon_0 + \cdots + \epsilon_j p_i^j \right] g$ . The key point of the Pohlig-Hellman reduction is to observe that  $\mathbf{h} = \left[ \frac{n}{p_i^{j+2}} \right] (h \ominus \gamma) = [\epsilon_{j+1}] \mathbf{g}$ . Hence, using an algorithm to solve the DLP in  $\mathbb{Z}/p_i\mathbb{Z}$  (see the two following Sections 5.3.2 and 5.3.3) one can compute  $\epsilon_{j+1}$  be the logarithm of  $\mathbf{h}$  to the base  $\mathbf{g}$ . This algorithm is summarized in Algorithm 17.

**Theorem 5.6.** *Let  $\mathbb{G} = \langle g \rangle$  be a finite cyclic group of order  $n$ . Let  $c_i$  be the complexity of solving the DLP in  $\mathbb{Z}/p_i\mathbb{Z}$ . Given  $h \in \mathbb{G}$  and the prime factorization of  $n = \prod_{i=1}^r p_i^{\alpha_i}$  with  $\alpha_i > 0$  the Pohlig-Hellman reduction (Algorithm 17) computes the discrete logarithm of  $h$  in base  $g$  in  $O(\sum_{i=1}^r \alpha_i(\log n + c_i))$  arithmetic operations in  $\mathbb{G}$ .*

While in the worst case, i.e.  $n$  is prime, this reduction does not reduce the complexity of solving the DLP, this shows that solving the DLP in groups with smooth order  $n$  can be done efficiently. Thus, these groups are weak from a cryptographic point of view.

**Algorithm 17:** Pohlig-Hellman reduction.

---

**Input** : A finite cyclic group  $\mathbb{G}$  of order  $n$ ,  $g$  a generator of  $\mathbb{G}$  and  $h$  in  $\mathbb{G}$ .  
**Output**: An integer  $x$  such that  $h = [x]g$ .

- 1 Compute  $p_i$  and  $\alpha_i$  for  $i = 1, \dots, r$  such that  $n = \prod_{i=1}^r p_i^{\alpha_i}$  is the prime factorization of  $n$ ;
- 2 **for**  $i := 1$  **to**  $r$  **do**
- 3      $\gamma := 1$ ;  $\epsilon_{-1} := 0$ ;  $\mathbf{g} := \left\lfloor \frac{n}{p_i} \right\rfloor g$ ;
- 4     **for**  $j := 0$  **to**  $\alpha_i - 1$  **do**
- 5          $\gamma := \gamma \oplus \left\lfloor \epsilon_{j-1} p_i^{j-1} \right\rfloor g$ ;  $\mathbf{h} := \left\lfloor \frac{n}{p_i^{j+1}} \right\rfloor (h \ominus \gamma)$ ;
- 6         Find  $\epsilon_j$  such that  $\mathbf{h} = [\epsilon_j] \mathbf{g}$  using Algorithm 18;
- 7      $x_i := \epsilon_0 + \dots + \epsilon_{\alpha_i-1} p_i^{\alpha_i-1}$ ;
- 8 Compute  $x$  using the *Chinese Remainder Theorem*;
- 9 **return**  $x$ ;

---

**5.3.2 Baby step giant step**

The baby step giant step algorithm of Shanks [Sha71] (see for instance [CP05, chapter 5] or [MVOV10, chapter 3]) is a trade-off between exhaustive search and memory requirement. Let  $n$  be the order of  $\mathbb{G}$ . Since  $\mathbb{G}$  is a cyclic group one looks for the integer  $x \in \{0, \dots, n-1\}$  such that  $h = [x]g$ . Let  $m$  be an integer in  $\{0, \dots, n-1\}$ . The Euclidean division of  $x$  by  $m$  can be written as  $x = qm + r$  where  $r \in \{0, \dots, m-1\}$  and  $q \in \{0, \dots, \lceil \frac{n}{m} \rceil\}$ . Moreover, we have  $h = [x]g = [qm + r]g$ . Hence,  $h \oplus [-qm]g = [r]g$ . Let us consider the two sequences  $a_i = h \oplus [-im]g$  for  $i = 0, \dots, \lceil \frac{n}{m} \rceil$  and  $b_j = [j]g$  for  $j = 0, \dots, m-1$ . Then, since the Euclidean division is unique there exist unique  $i$  and  $j$  such that  $a_i = b_j$ . In that case we have  $h \oplus [-im]g = [j]g$  which implies that  $x = im + j$ . The number of operations in  $\mathbb{G}$  or the memory required by this algorithm is in  $O(m + \frac{n}{m})$ . Hence, the best trade-off consists of choosing  $m = \lceil \sqrt{n} \rceil$ . This algorithm is summarized in Algorithm 18.

**Algorithm 18:** Baby step giant step for DLP.

---

**Input** : A finite cyclic group  $\mathbb{G}$  of order  $n$ ,  $g$  a generator of  $\mathbb{G}$ ,  $\mathcal{O}$  the neutral element of  $\mathbb{G}$  and  $h$  in  $\mathbb{G}$ .  
**Output**: An integer  $x$  such that  $h = [x]g$ .

- 1  $m := \lceil \sqrt{n} \rceil$ ;  $b_0 := \mathcal{O}$ ;  $a := h$ ;  $\alpha := [-m]g$ ;
- 2 **for**  $j := 1$  **to**  $m-1$  **do**  $b_j := g \oplus b_{j-1}$  and store  $(j, b_j)$  in a hash table;
- 3 **for**  $i := 0$  **to**  $m-1$  **do**
- 4     **if**  $a$  is the second component of an entry in the hash table **then**
- 5         Let  $(r, b_r)$  such an entry; **return**  $im + r$ ;
- 6      $a := a \oplus \alpha$ ;

---

**Theorem 5.7.** *Given a finite cyclic group  $\mathbb{G} = \langle g \rangle$  of order  $n$  and an element  $h \in \mathbb{G}$ . The baby step giant step algorithm (Algorithm 18) computes the discrete logarithm  $x$  of  $h$  in base  $g$  in  $O(\sqrt{n})$  arithmetic operations in  $\mathbb{G}$  and requires  $O(\sqrt{n})$  memory space.*

### 5.3.3 Pollard $\rho$ method

The Pollard  $\rho$  method [Pol78] (see also [MVOV10, chapter 3] or [CF05, chapter 19]) is a randomized algorithm. Its arithmetic complexity is the same as that of the baby step giant step however its memory requirement is in  $O(1)$ . The Pohlig-Hellman reduction allows, without loss of generality, to assume that the order  $n$  of the group  $\mathbb{G}$  is prime.

The principle of the Pollard  $\rho$  method is to iterate a function  $f : \mathbb{G} \mapsto \mathbb{G}$  whose behavior is close from that of a random function. Since, the group is finite one can find a collision which gives the discrete logarithm. The principle of Pollard  $\rho$  method is depicted on Figure 5.4.

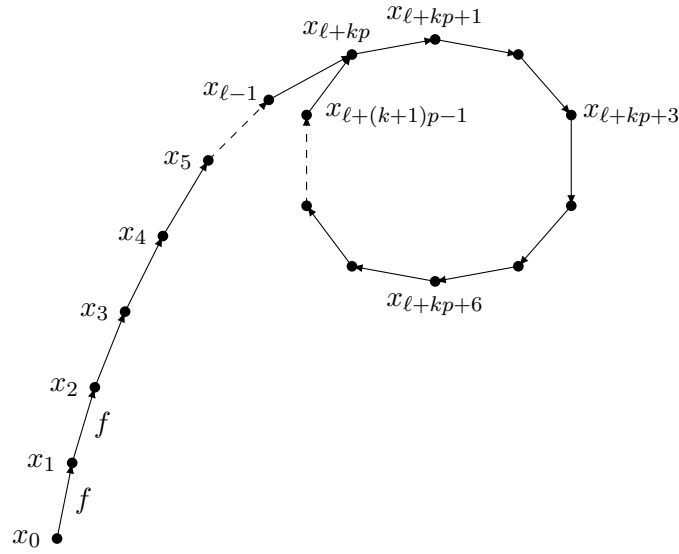


Figure 5.4: Pollard  $\rho$  method.

More precisely,  $\mathbb{G}$  is partitioned into three sets  $S_1, S_2$  and  $S_3$  of approximately same size. After ensuring that  $\mathcal{O} \notin S_2$ , the function  $f$  is defined as follows

$$x_{i+1} = f(x_i) = \begin{cases} h \oplus x_i & \text{if } x_i \in S_1 \\ [2]x_i & \text{if } x_i \in S_2 \\ g \oplus x_i & \text{if } x_i \in S_3 \end{cases} \quad (5.10)$$

The sequence  $(x_i)_{i \in \mathbb{N}}$  is then defined iteratively by  $f$  and  $x_0 = \mathcal{O}$ . Hence, for any  $i \geq 0$  we have  $x_i = [\alpha_i]g \oplus [\beta_i]h$ . Thus, the sequence  $(x_i)_{i \in \mathbb{N}}$  determines two others sequences  $(\alpha_i)_{i \in \mathbb{N}}$  and  $(\beta_i)_{i \in \mathbb{N}}$  defined as follows

$$a_{i+1} = \begin{cases} a_i & \text{if } x_i \in S_1 \\ 2a_i \mod n & \text{if } x_i \in S_2 \\ a_i + 1 \mod n & \text{if } x_i \in S_3 \end{cases} \quad b_{i+1} = \begin{cases} b_i + 1 \mod n & \text{if } x_i \in S_1 \\ 2b_i \mod n & \text{if } x_i \in S_2 \\ b_i & \text{if } x_i \in S_3 \end{cases}.$$

Assume a collision  $x_i = x_j$  is found. Then, we have  $[\alpha_i]g \oplus [\beta_i]h = [\alpha_j]g \oplus [\beta_j]h$ . Hence, if  $\beta_i - \beta_j \neq 0 \mod n$  then the discrete logarithm is given by  $x = \frac{\alpha_j - \alpha_i}{\beta_i - \beta_j} \mod n$ . Consequently, solving the DLP is reduced to finding a collision. In order to find such a collision efficiently in time and in memory one can use Floyd's cycle finding algorithm [Flo67] (see for instance [MVOV10, chapter 3]) whose complexity is in  $O(\sqrt{n})$  operations in  $\mathbb{G}$  and requires memory in  $O(1)$ .

**Theorem 5.8.** *Let  $\mathbb{G} = \langle g \rangle$  be a finite cyclic group of order  $n$  and  $h \in \mathbb{G}$ . Assuming the function  $f$  (equation (5.10)) behaves like a random function, the Pollard  $\rho$  method is a probabilistic algorithm computing the discrete logarithm of  $h$  in base  $g$  in  $O(\sqrt{n})$  arithmetic operations in  $\mathbb{G}$  and requiring  $O(1)$  memory space.*

Surprisingly, a result of Shoup [Sho97] shows that in general this is the best complexity that one can expect using a generic algorithm. However, using the structure of the group  $\mathbb{G}$  depending on the nature of the group, one can design more efficient algorithms. In particular when  $\mathbb{G} = \mathbb{F}_q^*$  is the multiplicative group constructed from a finite field, index calculus algorithm allows to obtain sub-exponential complexity, see [AD94]. More recently, new index calculus algorithms have been design improving the complexity of solving the DLP in  $\mathbb{F}_q^*$ . First, Joux proposes in [Jou13b] a new index calculus algorithm whose better complexity but still sub-exponential. Shortly after, inspired by Joux's algorithm, Barbulescu *et al* [BGJT13] present a quasi-polynomial time algorithm for solving the DLP in  $\mathbb{F}_q^*$  when  $q = p^k$  and  $p$  is small.

In this thesis, we focus on groups constructed from the set of rational points of an elliptic curve. Note that for this particular case, this problem is denoted ECDLP for elliptic curve discrete logarithm problem. In 2004, Semaev [Sem04] attempts to design an index calculus algorithm for solving the ECDLP for elliptic curves defined over a prime field  $\mathbb{F}_p$ . The difficulty in index calculus method to solve the DLP is to find a factor base *i.e.* a subset of elements in the group for which we know the complexity and the probability of decomposing any element of the group w.r.t. this factor base. Moreover, this complexity and this probability should be *good* enough to ensure to obtain an efficient algorithm. In [Sem04], Semaev does not highlight such an efficient factor base. Hence, his algorithm does not really apply theoretically and experimentally. However, he introduces a useful tool for decomposing points of elliptic curves: *summation polynomials*. Later, Gaudry and Diem independently propose more efficient index calculus algorithms for solving the ECDLP. Both of them are using the summation polynomials to solve the underlying problem that follows.

**Point Decomposition Problem (PDP).** *Given a point  $R$  of an elliptic curve defined over a field  $\mathbb{K}$ , denoted  $E(\mathbb{K})$ ; and given a factor base  $\mathcal{F} \subset E(\mathbb{K})$  find  $P_1, \dots, P_m \in \mathcal{F}$  such that  $R = P_1 \oplus \dots \oplus P_m$ .*

## 5.4 Semaev summation polynomials

Originally, the summation polynomials were introduced by Semaev as a projection of the PDP over the set of  $x$ -coordinates of the points.

**Definition 5.9.** *Let  $E$  be an elliptic curve defined by a planar equation over a field  $\mathbb{F}_{q^n}$  and let  $\overline{\mathbb{F}_{q^n}}$  be an algebraic closure of this field. For all  $m \geq 2$ , the  $m$ th summation polynomial of  $E$  is defined by  $f_m(x_1, \dots, x_m)$  such that for all  $x_1, \dots, x_m$  in  $\overline{\mathbb{F}_{q^n}}$ , its evaluation  $f_m(x_1, \dots, x_m)$  is zero if and only if there exist  $y_1, \dots, y_m \in \overline{\mathbb{F}_{q^n}}$  such that  $(x_i, y_i)$  is in  $E(\overline{\mathbb{F}_{q^n}})$  and  $(x_1, y_1) \oplus \dots \oplus (x_m, y_m)$  is the neutral element of  $E$ .*

More generally the summation polynomials can be defined as a projection over the set of any coordinate. In the context of Definition 5.9 and if  $E$  is in Weierstrass representation we have the following result.

**Theorem 5.10** (Semaev [Sem04]). *Let  $E$  be an elliptic curve defined over a field of characteristic  $> 3$  by a Weierstrass equation*

$$E : y^2 = x^3 + a_4x + a_6. \quad (5.11)$$

*The summation polynomials of  $E$  are given by*

$$\begin{cases} S_2(x_1, x_2) &= x_1 - x_2 \\ S_3(x_1, x_2, x_3) &= (x_1 - x_2)^2 x_3^2 - 2((x_1 x_2 + a_4)(x_1 + x_2) + 2a_6)x_3 + \\ &\quad (x_1 x_2 - a_4)^2 - 4a_6(x_1 + x_2) \\ S_m(x_1, \dots, x_n) &= \text{Res}_X(S_{m-k}(x_1, \dots, x_{m-k-1}, X), S_{k+2}(x_{m-k}, \dots, x_m, X)) \\ &\quad \text{for all } m \geq 4 \text{ and for all } m-3 \geq k \geq 1 \end{cases}$$

where  $\text{Res}_X(f_1, f_2)$  is the resultant of  $f_1$  and  $f_2$  with respect to  $X$ . Moreover, for all  $m \geq 3$  the  $m$ th summation polynomial is symmetric and of degree  $2^{m-2}$  in each variable. Furthermore, summation polynomials are irreducible.

A proof that summation polynomials are symmetric can be deduced from a more general result proven in Chapter 6.

**Remark 5.11.** *If the field  $\mathbb{K}$  is of characteristic two and the elliptic curve  $E$  is an ordinary curve defined over  $\mathbb{K}$  by a short Weierstrass equation*

$$E : y^2 + xy = x^3 + \alpha x^2 + \beta$$

with  $\alpha, \beta \in \mathbb{K}$ . Then, the third summation polynomial is given by

$$S_3(x_1, x_2, x_3) = x_1^2 x_2^2 + x_1^2 x_3^2 + x_2^2 x_3^2 + x_1 x_2 x_3 + \beta.$$

One can notice that in characteristic two,  $S_3$  is more sparse than in characteristic  $> 3$ . This is a general fact that we observe for  $S_m$  with  $m \geq 3$ . In Chapter 7, we will see how to take advantage of sparsity of summation polynomials in characteristic two to speed up their computation.

We now detail how the third summation polynomial is constructed in [Sem04]. The others being constructed recursively from the third.

### 5.4.1 Computing summation polynomials

First note that since summation polynomials for Weierstrass curves are a projection of the PDP over the set of the  $x$ -coordinate, the solutions of the  $m$ th summation polynomial are in fact  $(x_1, \dots, x_m)$  such that there exist  $(x_1, y_1), \dots, (x_m, y_m) \in E(\overline{\mathbb{K}})$  verifying  $\pm(x_1, y_1) \pm \dots \pm (x_m, y_m) = \mathcal{O}$ . Indeed, for Weierstrass curves of the form (5.11), we have  $\ominus(x, y) = (x, -y)$ . Hence, the solutions of summation polynomials are up to sign the solutions of the PDP. Let  $P_1 = (x_1, y_1)$  and  $P_2 = (x_2, y_2)$  be two points of the curve  $E$ . Let  $P_3 = (x_3, y_3)$  be the point of  $E$  such that  $P_3 = \pm P_1 \pm P_2$ . Since, the  $x$ -coordinate of a point, denoted  $x(P)$ , and its negative are equal then  $x_3$  is either  $x(P_1 \oplus P_2)$  or  $x(P_1 \ominus P_2)$ . Consequently,  $x_3$  is a solution of  $(x - x(P_1 \oplus P_2))(x - x(P_1 \ominus P_2))$ . Moreover, from Section 5.1  $x(P_1 \oplus P_2)$  (respectively  $x(P_1 \ominus P_2)$ ) is given by a rational fraction in terms of  $x_1, y_1, x_2, y_2$ . By consequence, let  $\frac{N}{D}$  be the irreducible form of the rational fraction  $(x_3 - x(P_1 \oplus P_2))(x_3 - x(P_1 \ominus P_2))$ . Then,

the solutions of the third summation polynomial are the projection of the solutions of the following system  $\mathcal{S}$  ensuring that  $D \neq 0$  into the variables  $x_1, x_2$  and  $x_3$ .

$$\mathcal{S} = \left\{ \begin{array}{l} N(x_1, y_1, x_2, y_2, x_3) \\ y_1^2 - (x_1^3 + a_4 x_1 + a_6) \\ y_2^2 - (x_2^3 + a_4 x_2 + a_6) \end{array} \right\}. \quad (5.12)$$

To ensure that  $D \neq 0$ , we consider the saturation ideal  $\mathcal{I}$  of  $\langle \mathcal{S} \rangle$  by  $D$ . Then, we project onto the variables  $x_1, x_2, x_3$  by computing the elimination ideal  $\mathcal{J}$  of  $\mathcal{I}$  in order to eliminate the variables  $y_1$  and  $y_2$  (see Section 2.1.4). Finally, the ideal  $\mathcal{J}$  is a principal ideal generated by the third summation polynomial.

**Remark 5.12.** Let  $\iota$  be the automorphism of degree 2 of  $E$  which associates to a point its negation:

$$\begin{array}{ccc} \iota : E(\mathbb{K}) & \longrightarrow & E(\mathbb{K}) \\ (x, y) & \longmapsto & \ominus(x, y) = (x, -y). \end{array}$$

Let  $\pi_x$  and  $\pi_y$  be respectively, the projection on  $x$  and  $y$ . We can note that  $\pi_x(x, y) = \pi_x(\iota(x, y))$  and  $\pi_y(x, y) \neq \pi_y(\iota(x, y))$ . Clearly,  $\pi_x(E) \simeq E/\iota$  and the PDP in  $m$  points have more solutions in  $E^m$  than in  $(E/\iota)^m$ . This is not true for  $\pi_y$ . By consequence, by projecting on  $x$ , we obtain summation polynomials with smaller degree. In the following, for non-Weierstrass equations we choose to project on the coordinate  $c$ , if it exists, such that there exists an automorphism  $\psi$  of  $E$  such that  $\pi_c(E) \simeq E/\psi$  and for all  $P$ ,  $\pi_c(P) = \pi_c(\psi(P))$ . For each studied representation, this automorphism exists and will be  $\iota$ .

Following this construction we give the summation polynomials corresponding to the curve representations presented in Section 5.2.

### 5.4.2 Twisted Jacobi intersection curves

As said in Remark 5.12, we compute the summation polynomials as a projection of the PDP to the coordinate which is invariant under the  $\ominus$  action. For twisted Jacobi intersection curves the  $y$  and  $z$  coordinates are invariant under the action of  $\ominus$ . Hence we can compute the summation polynomials for these curves as a projection of the PDP to the  $y$  or  $z$  coordinate. In fact the two summation polynomials for  $n$  fixed are the same up to permutation of  $a$  and  $b$ , so we give only the polynomials obtained by projection to  $y$ :

$$\left\{ \begin{array}{lcl} S_2(y_1, y_2) & = & y_1 - y_2 \\ S_3(y_1, y_2, y_3) & = & (y_1^2 y_2^2 - y_1^2 - y_2^2 + \frac{b-a}{b}) y_3^2 + 2\frac{a}{b} y_1 y_2 y_3 + \\ & & \frac{b-a}{b} (y_1^2 + y_2^2 - 1) - y_1^2 y_2^2 \\ S_n(y_1, \dots, y_n) & = & \text{Res}_Y(S_{n-k}(y_1, \dots, y_{n-k-1}, Y), S_{k+2}(y_{n-k}, \dots, y_n, Y)) \\ & & \text{for all } n \geq 4 \text{ and for all } n-3 \geq k \geq 1 \end{array} \right.$$

As in the case of Weierstrass representation, for all  $n \geq 3$  the  $n$ th summation polynomial is symmetric (see proof in Section 6.2.2) and of degree  $2^{n-2}$  in each variable. Moreover, the proof of irreducibility of summation polynomials by Semaev does not depend on the representation of the curve or the coordinate we project to. Hence, it can be applied *mutatis mutandis* for twisted Edwards or Jacobi intersection summation polynomials.

### 5.4.3 Twisted Edwards curves

For this representation, the  $y$ -coordinate is invariant under the action of  $\ominus$ . Thus, the  $n$ th summation polynomial is constructed as a projection of the PDP on the  $y$ -coordinate. For twisted Edwards curves, it is then given by

$$\left\{ \begin{array}{lcl} S_2(y_1, y_2) & = & y_1 - y_2 \\ S_3(y_1, y_2, y_3) & = & (y_1^2 y_2^2 - y_1^2 - y_2^2 + \frac{a}{d}) y_3^2 + 2 \frac{d-a}{d} y_1 y_2 y_3 + \\ & & \frac{a}{d} (y_1^2 + y_2^2 - 1) - y_1^2 y_2^2 \\ S_n(y_1, \dots, y_n) & = & \text{Res}_Y(S_{n-k}(y_1, \dots, y_{n-k-1}, Y), S_{k+2}(y_{n-k}, \dots, y_n, Y)) \\ & & \text{for all } n \geq 4 \text{ and for all } n-3 \geq k \geq 1 \end{array} \right.$$

As for Weierstrass and twisted Jacobi intersection curves, these summation polynomials are irreducible and for all  $n \geq 3$  the  $n$ th summation polynomial is symmetric and of degree  $2^{n-2}$  in each variable.

### 5.4.4 Universal Edwards model of elliptic curves

For this representation, the  $x$ -coordinate is invariant under the action of  $\ominus$ . Thus, the  $n$ th summation polynomial is given by

$$\left\{ \begin{array}{lcl} S_2(x_1, x_2) & = & x_1 - x_2 \\ S_3(x_1, x_2, x_3) & = & x_1^2 x_2^2 x_3^2 - \alpha (x_1^2 x_2^2 + x_1^2 x_3^2 + x_2^2 x_3^2) + (\frac{\alpha}{t} - 4t) x_1 x_2 x_3 + \\ & & x_1^2 + x_2^2 + x_3^2 - \alpha \\ S_n(x_1, \dots, x_n) & = & \text{Res}_X(S_{n-k}(x_1, \dots, x_{n-k-1}, X), S_{k+2}(x_{n-k}, \dots, x_n, X)) \\ & & \text{for all } n \geq 4 \text{ and for all } n-3 \geq k \geq 1 \end{array} \right.$$

where  $\alpha = \frac{1}{4t^2}$ . As for Weierstrass, twisted Jacobi intersection curves and twisted Edwards curves, these summation polynomials are irreducible and for all  $n \geq 3$  the  $n$ th summation polynomial is symmetric and of degree  $2^{n-2}$  in each variable.

## 5.5 Gaudry's index calculus attack for solving the elliptic curve discrete logarithm problem

In this section, we present the index calculus attack of Gaudry [Gau09] for solving elliptic curve discrete logarithm problem. Originally, his algorithm has been designed for solving the DLP in any abelian variety of fixed dimension  $n \geq 2$ . The elliptic curve case arises as a particular case of Gaudry's algorithm. For this particular case, the use of summation polynomials instead of the general method of Gaudry's algorithm for decomposing points allows to speed up the relation search step. Since, in this thesis we focus on ECDLP, we present this index calculus attack in this context.

### 5.5.1 Presentation of the algorithm

Usually, index calculus algorithm proceeds in three steps. Assume we want to compute  $x$  such that  $Q = [x]P$ . The first step consists of finding an *efficient* factor base. Then, the second step that we call *relation search* consists of computing relations between  $P, Q$  and the elements in the factor base. Finally, when enough relations are computed, the third step uses linear algebra to recover the discrete logarithm  $x$ .

First, we describe how the elliptic curves case appears as a particular case of abelian variety of dimension greater than one.

### Elliptic curves defined over non prime finite field as abelian variety of dimension $n > 1$

Let  $E$  be an elliptic curve in Weierstrass representation defined over  $\mathbb{F}_{q^n}$  with  $n > 1$ . Note that we assume the curve given by a Weierstrass representation to follow the presentation in [Gau09]. However, same reasoning works *mutatis mutandis* with the other representations.

Writing  $\mathbb{F}_{q^n} = \mathbb{F}_q[X]/\mu(X) = \mathbb{F}_q[\alpha]$  where  $\mu(X)$  is an irreducible polynomial over  $\mathbb{F}_q$  of degree  $n$  and  $\alpha$  is a root of  $\mu(X)$  in  $\mathbb{F}_{q^n}$ , we can see  $\mathbb{F}_{q^n}$  as a vector space over  $\mathbb{F}_q$  for which  $\{1, \alpha, \dots, \alpha^{n-1}\}$  is a basis. At each element  $x$  in  $\mathbb{F}_{q^n}$  we can thus associate a unique  $n$ -tuple of  $\mathbb{F}_q^n$  by the following isomorphism

$$\begin{aligned} \phi : \quad \mathbb{F}_{q^n} &\rightarrow \mathbb{F}_{q^n} \\ (x_0, \dots, x_{n-1}) &\mapsto \sum_{i=0}^{n-1} x_i \alpha_i \end{aligned}$$

Hence, the set of points of  $E(\mathbb{F}_{q^n})$  can be seen as an abelian variety defined over  $\mathbb{F}_q$  of dimension  $n$ . Let  $A$  be such an abelian variety defined by  $A = \{(x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1}) \in (\mathbb{F}_q)^{2n} \mid (\phi(x_0, \dots, x_{n-1}), \phi(y_0, \dots, y_{n-1})) \in E(\mathbb{F}_{q^n})\}$ . The abelian variety  $A$  is called the *Weil restriction* of  $E(\mathbb{F}_{q^n})$  from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$ . Note that  $A$  is of dimension  $n$  since the  $y_i$ 's are algebraic over  $x_0, \dots, x_{n-1}$ .

**Remark 5.13.** *The notion of Weil restriction is more general than as presented above. Indeed, a Weil restriction can be applied to all geometric objects defined over separable fields  $\mathbb{L}$  of degree  $d$  of a ground field  $\mathbb{K}$  and not only to affine variety. The principle is to relate objects of dimension  $n$  over  $\mathbb{L}$  to  $nd$ -dimensional objects over  $\mathbb{K}$ . See for instance [CF05] for a general definition of Weil restriction. We do not need such a definition in this thesis.*

Frey [Fre01] showed that any instance of the ECDLP can be mapped to an instance of the DLP in the Weil restriction of  $E(\mathbb{F}_{q^n})$  from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$ . In the same way, the PDP over any elliptic curve defined over a non prime finite field can be mapped to the PDP over  $A$ .

### Factor base

The factor base is defined as in [Gau09] by

$$\mathcal{F} = \{P \in A \cap H_1 \cap \dots \cap H_{n-1}\},$$

where  $A$  is the Weil restriction of  $E(\mathbb{F}_{q^n})$  from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$ . The hyperplane  $H_i$  is defined by the equation  $x_i = 0$ . In other words, the factor base  $\mathcal{F}$  is defined as a subset of  $E(\mathbb{F}_{q^n})$  as follow

$$\mathcal{F} = \{P = (x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}.$$

Thus, the factor base  $\mathcal{F}$  is an algebraic variety since it is defined as intersection of algebraic varieties. Since, the  $y_i$ 's are algebraic over the  $x_i$ 's,  $\mathcal{F}$  is of dimension 1. Generically (possibly by applying a random linear change of variables),  $\mathcal{F}$  is an absolutely irreducible variety. Hence, the number of points in  $\mathcal{F}$  can be estimated thanks to the Hasse-Weil bound (see for instance [CF05, chapter 5]) which is a consequence of the Weil conjecture [Wei49].

**Hasse-Weil bound.** *Let  $C$  be a smooth projective curve of genus  $g$ . The number of points of  $C$  defined in  $\mathbb{F}_q$ , denoted  $\#C(\mathbb{F}_q)$  satisfies*

$$|\#C(\mathbb{F}_q) - q - 1| \leq 2g\sqrt{q}.$$

*In other words,  $\#C(\mathbb{F}_q) = q + O(\sqrt{q})$  where the constant depends on the genus of the curve.*

Consequently, if  $\mathcal{F}$  is smooth (*i.e.* has no singularity) then its cardinality is  $\#\mathcal{F} = q + O(\sqrt{q})$ . In the case where  $\mathcal{F}$  does not contain enough points *i.e.* its cardinality is not  $q + O(\sqrt{q})$  we can choose any factor base of the form

$$\mathcal{F}_i = \{P \in A \cap H_0 \cap \cdots \cap H_{i-1} \cap H_{i+1} \cap \cdots \cap H_{n-1}\}.$$

### Relation collection

Let  $P, Q \in A$  such that  $Q = [x]P$  where we want to compute  $x$ . Let  $m$  be the order of  $P$ . As usual in index calculus algorithm, we look for relations between  $P$  and  $Q$ . More precisely, such a relation is of the form

$$R_i = [a_i]P \oplus [b_i]Q = P_{i,1} \oplus \cdots \oplus P_{i,n} \quad (5.13)$$

where  $P_{i,1}, \dots, P_{i,n}$  are some points in  $\mathcal{F}$  and  $a_i$  and  $b_i$  are randomly chosen in  $\mathbb{Z}/m\mathbb{Z}$ .

Given a point  $R$  of  $E(\mathbb{F}_{q^n})$  the probability that it can be written as a sum of  $n$  points in  $\mathcal{F}$  is about  $\frac{1}{n!}$ . Indeed, consider the map  $\phi$  defined as follow

$$\begin{aligned} \phi : \quad \mathcal{F}^n / \mathfrak{S}_n &\rightarrow A \\ (P_1, \dots, P_n) &\mapsto P_1 \oplus \cdots \oplus P_n \end{aligned}$$

where  $\mathfrak{S}_n$  is the  $n$ th symmetric group. The probability that a *generic* point  $R$  can be decomposed w.r.t.  $\mathcal{F}$  is then given by

$$\frac{1}{\#A} \sum_{R \in A} \#\phi^{-1}(R) = \frac{\#(\mathcal{F}^n / \mathfrak{S}_n)}{\#A} = \frac{\#\mathcal{F}^n}{n! \#A}.$$

As previously mentioned,  $\#\mathcal{F}$  is about  $q$  and the Hasse-Weil bound allows to estimate the cardinality of  $A$  which is about  $q^n$ .

**Hypothesis 5.14.** *There exist approximately  $\frac{q^n}{n!}$  points of  $E(\mathbb{F}_{q^n})$  which can be decomposed as the sum of  $n$  points in  $\mathcal{F}$ . Thus, a relation of the form (5.13) can be found with probability  $\frac{1}{n!}$ .*

The group law of  $E$  gives a group law on  $A$  which is given by rational fractions depending on the coordinates of the summed points. Consequently we can construct  $2n$  rational fractions  $\lambda_j$  in terms of the  $n(n+1)$  variables  $x_{i,0}, y_{i,0}, \dots, y_{i,n-1}$  for  $i = 1, \dots, n$  such that

$$P_1 \oplus \cdots \oplus P_n = (\lambda_1, \dots, \lambda_{2n})$$

where  $P_i = (x_{i,0}, 0, \dots, 0, y_{i,0}, \dots, y_{i,n-1}) \in \mathcal{F}$ .

To solve the PDP, we write  $P_1 \oplus \cdots \oplus P_n = R$  which gives  $2n$  equations in  $\mathbb{F}_q$ . Adding the equations describing  $P_i \in E$  for  $i = 1, \dots, n-1$ , we obtain a polynomial system with  $n(n+1)$  variables and  $n(n+1)$  equations in  $\mathbb{F}_q$ . It is not necessary to add the equation for

$P_n \in E$  because this information is already in the system. Indeed, we have  $P_1, \dots, P_{n-1} \in E$  and  $P_n = R \ominus (P_1 \oplus \dots \oplus P_{n-1})$  with  $R \in E$  and by consequence  $P_n$  too. The system has as many unknowns as equations then under regularity assumptions, it is of dimension 0. In practice, we observe that this system is of dimension zero. By consequence, we assume the following hypothesis.

**Hypothesis 5.15.** *Polynomial systems coming from the resolution of equation (5.13) are of dimension zero.*

In order to solve this system, we use Gröbner bases. As shown in Chapter 2 the complexity of Gröbner bases computations depends on the number of variables which is quadratic in  $n$ . To speed up the resolution, one can reduce the number of variables by using the summation polynomials. Indeed, using summation polynomial allows to *remove* the variables  $y_{i,j}$  for  $i = 1, \dots, n$  and  $j = 0, \dots, n-1$ .

We now detail how to use the summation polynomials to solve the PDP. By definition, if the points  $P_1, \dots, P_n \in \mathcal{F}$  verify

$$S_{n+1}(x_{P_1}, \dots, x_{P_n}, x_R) = 0_{\mathbb{F}_{q^n}} \quad (5.14)$$

then, up to signs, they give a solution of the PDP for  $R$ . By applying a Weil restriction, we obtain

$$S_{n+1}(x_{P_1}, \dots, x_{P_n}, x_R) = 0_{\mathbb{F}_{q^n}} \iff \sum_{k=0}^{n-1} \varphi_{R,k}(x_{P_1}, \dots, x_{P_n}) \cdot \alpha^k = 0_{\mathbb{F}_{q^n}}$$

where the  $\varphi_{R,k}(x_{P_1}, \dots, x_{P_n})$  are polynomials in  $\mathbb{F}_q[x_{P_1}, \dots, x_{P_n}]$ . Thus, solving equation (5.14) is equivalent to solve the polynomial system  $\mathcal{S} = \{\varphi_{R,k}(x_{P_1}, \dots, x_{P_n}), k = 0, \dots, n-1\}$  in  $\mathbb{F}_q$ .

**Remark 5.16.** *According to Remark 5.12 depending on the representation we do not use the same coordinate to construct summation polynomial. Hence, to adapt this algorithm to another representation we need to adjust the factor base: let  $c$  be the chosen coordinate to construct summation polynomials,  $\mathcal{F}$  has to be the set of all points of the curve with  $c$  in  $\mathbb{F}_q$  instead of  $\mathbb{F}_{q^n}$ . The probability of decomposing a point w.r.t.  $\mathcal{F}$  still follows the Hypothesis 5.14.*

Note that since  $P$  and  $\ominus P$  share the same abscissa we have that  $P \in \mathcal{F}$  implies that  $\ominus P \in \mathcal{F}$  too. Moreover, summation polynomials are a projection of the PDP on the  $x$ -coordinate of the summed point. Thus, the solutions of equation (5.14) allow to construct any decomposition of the form

$$R = \pm P_1 \pm \dots \pm P_n.$$

Let  $S_1, S_2 \subset E(\mathbb{F}_{q^n})$  be such that  $\mathcal{F} = S_1 \cup S_2$ ,  $S_1 \cap S_2 = \{P \in \mathcal{F} \mid [2]P = \mathcal{O}\}$  and  $S_i = \iota(S_j)$  with  $i \neq j$ . By consequence, considering the factor base  $S_1$  instead of  $\mathcal{F}$  and searching for relations of the form  $R = P_1 \oplus \dots \oplus P_n$  such that  $P_i$  or  $\ominus P_i$  are in  $S_1$  allows to divide the size of the factor base by a factor two without decreasing the probability of decomposing a point. Moreover, the process of relation search by using summation polynomials does not change since they do not distinguish  $P$  and  $\ominus P$ .

Although this trick has no impact in the asymptotic complexity it can give significant improvements in practice.

### Computing the discrete logarithm

Assume we have computed enough relations of the form (5.13) such that there exists a non trivial linear combination of these relations vanishing on  $A$ . In other words, there exists  $v_1, \dots, v_N \neq 0$  such that

$$\sum_{i=1}^N [v_i] R_i = \sum_{i=1}^N [v_i] (P_{i,1} \oplus \dots \oplus P_{i,n}) = \mathcal{O}.$$

That is to say  $\sum_{i=1}^N [v_i]([a_i]P \oplus [b_i]Q) = \mathcal{O}$ . Hence,  $\sum_{i=1}^N [v_i](a_i + xb_i)P = \mathcal{O}$ . Consequently, if  $m$  denotes the order of  $P$  in  $A$  we have  $\sum_{i=1}^N v_i(a_i + xb_i) = 0 \pmod{m}$ . Thus, if  $\sum_{i=1}^N v_i b_i$  is invertible modulo  $m$  we have

$$x = -\frac{\sum_{i=1}^N v_i a_i}{\sum_{i=1}^N v_i b_i} \pmod{m}. \quad (5.15)$$

Index calculus attack as presented in [Gau09] for solving the elliptic curve discrete logarithm is summarized in Algorithm 19.

---

**Algorithm 19:** Index calculus attack for ECDLP.

---

**Input** : Two points  $P$  and  $Q$  in  $E(\mathbb{F}_{q^n})$  with  $n > 1$  such that  $Q \in \langle P \rangle$  and  $m$  the order of  $P$ .

**Output:** The integer  $x$  such that  $Q = [x]P$  and  $0 \leq x < m$ .

- 1 Compute the factor base  $\mathcal{F} = \{(x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}$ ;
  - 2 Compute  $\#\mathcal{F} + 1$  relations ( $\#\mathcal{F}$  independent relations and any other) of the form (5.13);
  - 3 Using linear algebra, find  $v_1, \dots, v_{\#\mathcal{F}+1}$  such that  $\mathcal{O} = \sum_{i=1}^{\#\mathcal{F}+1} [v_i a_i]P \oplus [v_i b_i]Q$ ;
  - 4  $A = \sum_{i=1}^{\#\mathcal{F}+1} v_i a_i$ ;  $B = \sum_{i=1}^{\#\mathcal{F}+1} v_i b_i$ ;
  - 5 **if**  $B$  is invertible modulo  $m$  **then return**  $-\frac{A}{B} \pmod{m}$ ;
  - 6 **else** go back to step 2;
- 

### 5.5.2 Complexity analysis

To compute the factor base, we proceed as follows. We recall that we assume the curve given in Weierstrass representation but similar reasoning apply for others representations. For each value  $x \in \mathbb{F}_q$ , we substitute it into the equation defining the curve and then we have to compute a square root in  $\mathbb{F}_{q^n}$  or factorize a polynomial of degree 2 if the field is of characteristic two. This can be done in a polynomial time in  $\log(q)$  and  $n$ , see for instance [VZGG03]. Hence, the computation of  $\mathcal{F}$  can be done in  $\tilde{O}(q)$  operations where the notation  $\tilde{O}$  means that we neglect logarithmic factors in  $q$  and polynomial factors in  $n$ .

Let  $c$  be the cost of finding a relation of the form (5.13) *i.e.* solving equation (5.14). Let  $\mathcal{P}$  be the probability of finding a relation *i.e.*  $\frac{1}{n!}$ . The cost of the relation search step is then in  $O\left(\frac{qc}{\mathcal{P}}\right)$  operations.

Finally, since each relation involves at most  $n$  points, the matrix contains at most  $O(nq)$  non zero entries. Hence, using for instance Wiedemann algorithm [Wie86] (or [CF05, page 501]) taking advantage of the sparsity of the matrix, the linear algebra step can be done in

$O(nq^2 \log(q)^2)$  operations in  $\mathbb{Z}/m\mathbb{Z}$  where we recall that  $m$  is the order of  $P$ . Hence, at worst  $m = O(q^n)$ . By consequence, the total complexity of Algorithm 19 can be expressed as

$$O\left(\frac{qc}{\mathcal{P}} + nq^2 \log(q)^2\right) \quad (5.16)$$

arithmetic operations.

In [Gau09], it is considered abelian variety of fixed dimension *i.e.*  $n$  is fixed. In that case the cost of computing a relation of the form (5.13) is polynomial in  $\log(q)$ . Indeed, we have to solve a polynomial system with coefficients in  $\mathbb{F}_q$  whose degree and number of variables are fixed. Consequently, the complexity of Algorithm 19 is dominated by the linear algebra step and is in  $\tilde{O}(q^2)$ . Since, the linear algebra step is more time consuming than the relation search, one can balance the complexity of these two steps using the *double large prime variation* of Gaudry *et al* [GTTD07].

### 5.5.3 Balancing relation search and linear algebra using the *double large prime variation*

The idea is to decrease the size of the factor base to speed up the linear algebra step and increasing the complexity of the relation search. Indeed, since the factor base is smaller, the probability of finding a relation between  $P$  and  $Q$  is smaller too. It is a generalization of the *large prime variation* of Thériault [Thé03].

The principle is as follows. First, one divides the factor base  $\mathcal{F}$  into two sets. The first set  $G$  containing *genuine* elements has size  $(\#\mathcal{F})^r$  where  $0 < r \leq 1$ . The second set  $L$  contains elements called *large prime*. One looks for relations matching one of the three following forms:

$$[a_i]P \oplus [b_i]Q = P_1 \oplus \cdots \oplus P_n \quad (5.17)$$

$$[a_i]P \oplus [b_i]Q = P_1 \oplus \cdots \oplus P_{n-1} \oplus L_1 \quad (5.18)$$

$$[a_i]P \oplus [b_i]Q = P_1 \oplus \cdots \oplus P_{n-2} \oplus L_1 \oplus L_2 \quad (5.19)$$

where the  $P_i$ 's are genuine elements in  $G$  and the  $L_i$ 's are large prime in  $L$ . Then, by combining relations of the form (5.18) and (5.19) one can build relations involving only genuine elements (possibly more than  $n$ ). For this purpose, the authors of [GTTD07] suggest considering a *graph of large prime relations*. It is an undirected acyclic graph whose vertices correspond to large prime *i.e.* elements in  $L$  plus the special vertex  $\star$ . All edges are labelled with a relation. More precisely, if the relation contains two large primes then it labels the edge between the two vertices corresponding to these large primes. If the relation contains one large prime then it labels the edge between the vertex corresponding to this large prime and the special vertex  $\star$ . The graph is constructed as one goes along the relation collection until a cycle is detected. Indeed, if a new relation creates a cycle the corresponding edge is not added to the graph but the cycle allows to compute a new relation involving only genuine elements. The algorithm proceeds as above until  $\#G + 1$  relations involving only genuine elements are computed. Then, as usual the linear algebra allows to recover the discrete logarithm. Note that the size of the matrix is decreased to  $\#G$ .

From [GTTD07], the relation collection has now a complexity in

$$O\left(\left(1 + \frac{r(n-1)}{n}\right)(n-2)!q^{1-(n-2)(r-1)}\log(q)c\right)$$

arithmetic operations where  $c$  is the cost of finding one relation involving at most two large primes. Moreover, in [GTTD07] it is shown that the number of genuine elements in the recombined relation is in  $O(\log(q))$ . Hence, the complexity of the linear algebra becomes in

$$O(q^{2r} \log(q)^3)$$

arithmetic operations in  $\mathbb{Z}/m\mathbb{Z}$  where  $m = O(q^n)$  is the order of  $P$ .

Finally, one looks for  $r$  such that the cost of the linear algebra step and the relation collection are equal. In the context of the analysis of Gaudry in [Gau09] *i.e.* when the dimension of the variety  $n$  (or the degree of the extension of the field  $\mathbb{F}_{q^n}$ ) is fixed then  $c = O(1)$ . Hence, we look for  $r$  satisfying  $q^{1-(n-2)(r-1)} = q^{2r}$  if we omit logarithmic factors. Consequently, we obtain that by choosing  $r = 1 - \frac{1}{n}$  the complexity of Gaudry index calculus attack using the *double large prime variation* is in  $\tilde{O}\left(q^{2-\frac{2}{n}}\right)$ .

Although it is omitted in Gaudry's analysis, the cost of solving a polynomial system in order to find a relation is exponential in  $n$ . Hence, from  $n = 5$  the bottleneck of the index calculus attack is the resolution of such a system. In order to pass over this barrier Joux and Vitse propose in [JV13] a new variant of the index calculus attack that they call *variant "n - 1"* which is a trade-off between probability of decomposing a point and the difficulty of finding a relation.

#### 5.5.4 Variant "n - 1"

This approach can be seen as an hybrid approach where one mixes an exhaustive search and an algebraic resolution (*e.g.* see [BFP09] for application of such a strategy in another context). If one looks for a decomposition of a given point  $R$ , instead of searching for  $n$  points of the factor base whose sum is equal to  $R$ , one can search for only  $n - 1$  points of the factor base whose sum is equal to  $R$ . Using this technique simplifies the resolution of the polynomial systems, since we manipulate the summation polynomial of degree  $n$  instead of  $n + 1$  so that the degree and the number of variables are reduced. Furthermore, the systems become overdetermined and if they have a solution, then in general it is unique. Indeed, the number of variables is reduced to  $n - 1$  (the abscissa of the  $n - 1$  points in the decomposition) but the Weil restriction still leads to  $n$  equations. Hence the DRL Gröbner basis is also the LEX Gröbner basis and we do not need the FGLM step in the general solving strategy. On the other hand, it decreases the probability of finding a decomposition by a factor  $q/n$ . Consequently, from equation (5.16) the approach in [JV13] has an arithmetic complexity in

$$O\left(\frac{q^2 \mathcal{C}(n-1)}{(n-1)!} + (n-1)q^2 \log(q)^2\right)$$

where  $\mathcal{C}(n-1)$  is the cost of solving one system corresponding to the decomposition of a point of the curve in  $n - 1$  points of the factor base. Let  $\mathcal{C}(n)$  be the cost of solving one system corresponding to the decomposition of a point of the curve in  $n$  points of the factor base. This complexity has to be compared with

$$O\left(\frac{q \mathcal{C}(n)}{n!} + nq^2 \log(q)^2\right)$$

for the original version of Gaudry keeping in mind that  $\mathcal{C}(n)$  is more expensive than  $\mathcal{C}(n-1)$ . We will make more explicit the value of  $\mathcal{C}(n)$  and  $\mathcal{C}(n-1)$  in Section 5.6.1. Anyway, until

now the only viable approach for handling the case where  $n = 5$  is the *variant* “ $(n - 1)$ ” by Joux and Vitse [JV13].

### 5.5.5 Diem’s variant of the index calculus attack

As previously mentioned, at the same time Diem (independently) proposes an index calculus attack for solving the elliptic curve discrete logarithm for curves defined over non prime finite field. The main difference between the two approaches is the definition of the factor base. Diem’s algorithm uses a larger factor base. More precisely, let  $A$  be the Weil restriction of  $E(\mathbb{F}_{q^n})$  from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$ . Let  $m = \lceil \frac{n}{k} \rceil$  for some integer  $k \leq n$  and  $I \subset \{0, \dots, n-1\}$  such that  $\#I = k$ , the factor base  $\mathcal{F}$  is defined as follows

$$\begin{aligned} \mathcal{F} &= \left\{ P \in A \mid P \in \bigcap_{j \notin I} H_j \right\} \\ &= \left\{ (x, y) \in E(\mathbb{F}_{q^n}) \mid x = \sum_{i \in I} x_i \alpha^i, x_i \in \mathbb{F}_q, y \in \mathbb{F}_{q^n} \right\} \end{aligned}$$

where  $\{1, \alpha, \dots, \alpha^{n-1}\}$  is a basis of  $\mathbb{F}_{q^n}$  seen as a  $\mathbb{F}_q$ -vector space of dimension  $n$  and  $H_j$  is the hyperplane defined by the equation  $x_j = 0$ . Then, the algorithm is the same as that of Gaudry except that we look for decompositions of the form

$$R = [a_i]P \oplus [b_i]Q = P_1 \oplus \dots \oplus P_m$$

where  $P_1, \dots, P_m \in \mathcal{F}$ . The decompositions are still computed by solving polynomial systems obtained from applying the Weil restriction from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$  on the  $(m + 1)$ th summation polynomial evaluated in the abscissa of  $R$ .

Moreover, Diem’s analysis allows to remove heuristics present in Gaudry’s analysis as for instance the size of the factor base and the probability of finding a relation. The parameter  $k$  depends on the parameters  $q$  and  $n$  in input and is chosen in order to obtain the best complexity. In particular, Diem highlights some families of curves on which the discrete logarithm problem can be solved in subexponential time. Before that, the only way to obtain subexponential algorithm for some particular families of curves was to use transfer method, see for instance [MVO91, MOV93, Sem98]. The idea of such a method is to find an homomorphism from  $E(\mathbb{F}_q)$  to  $\mathbb{F}_{q^e}^\times$  then solving the problem in  $\mathbb{F}_{q^e}^\times$  for which there exists subexponential algorithm. Obviously, the complexity of such method depends on the existence of a small  $e$ . Note that for anomalous curves *i.e.* elliptic curves defined over a prime field  $\mathbb{F}_p$  whose the number of rational points is  $p$ , there even exists polynomial time algorithm, see [Sem98, Sma99].

## 5.6 Using symmetries to improve the ECDLP solving

Even if in the original analysis in [Gau09] the dimension and the degree of the equations defining the variety are fixed, the author looks for an efficient way of solving polynomial systems obtained by applying a Weil restriction from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$  on summation polynomials.

### 5.6.1 Solving the point decomposition problem

As it is shown in Section 5.4, the summation polynomials are symmetric and it is natural [Gau09] to use this to decrease the cost of the Gröbner basis computation in the polynomial systems solving process (see Chapter 3). Let  $S_m \in \mathbb{K}[x_1, \dots, x_m]$  be the  $m$ th summation polynomial of an elliptic curve given in Weierstrass representation. Since,  $S_m$  is symmetric we have  $S_m \in \mathbb{K}[x_1, \dots, x_m]^{\mathfrak{S}_m}$ . As mentioned in Chapter 3 Section 3.2.3 it is well known that the invariant ring of  $\mathfrak{S}_m$  is a polynomial algebra with basis  $\{e_1, \dots, e_m\}$  where  $e_i$  is the  $i$ th elementary symmetric polynomial in terms of  $x_1, \dots, x_m$ .

**Remark 5.17.** *Since,  $S_m$  is symmetric one can note that  $S_m$  is also invariant under the action of  $\mathfrak{S}_{m-1}$ . That is to say the polynomial  $S_m(x_1, \dots, x_{m-1}, x_R) \in \mathbb{K}[x_1, \dots, x_{m-1}]$  is in the invariant ring of  $\mathfrak{S}_{m-1}$ .*

To solve the PDP, as explained in Section 5.5.1 we consider the  $(n+1)$ th summation polynomial evaluated in the abscissa of the point  $R$  we want to decompose. That is to say, we apply the Weil restriction from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$  on the polynomial  $S_{n+1}(x_1, \dots, x_n, x_R)$ . According to Remark 5.17,  $S_{n+1}(x_1, \dots, x_n, x_R)$  is in  $\mathbb{F}_{q^n}[x_1, \dots, x_n]^{\mathfrak{S}_n}$ . Hence, there exists a unique polynomial  $S'_n \in \mathbb{F}_{q^n}[e_1, \dots, e_n]$  such that  $S'_n$  is the expression of  $S_{n+1}(x_1, \dots, x_n, x_R)$  in terms of the  $e_i$ . In Section 5.4, we have seen that  $S_{n+1}$  is of degree  $2^{n-1}$  in each variable thus  $S_{n+1}(x_1, \dots, x_n, x_R)$  too. Consequently, by construction  $S'_n$  is of total degree  $2^{n-1}$ .

To solve the PDP we now apply the Weil restriction on  $S'_n$  instead of  $S_{n+1}(x_1, \dots, x_n, x_R)$ . Hence, after the Weil restriction on  $S'_n \in \mathbb{F}_{q^n}[e_1, \dots, e_n]$  we obtain a new system  $\mathcal{S}_{\mathfrak{S}_n}^1 \subset \mathbb{F}_q[e_1, \dots, e_n]$  with  $n$  polynomials of total degree  $2^{n-1}$ .

Consequently, the Bezout's bound allows to bound the degree of the ideal generated by  $\mathcal{S}_{\mathfrak{S}_n}$  by  $2^{n(n-1)}$ . In practice, we observe in this context that this bound is reached. Without taking into account the symmetric group, the bound would have been  $n!$  times larger, therefore, the complexity of FGLM is reduced by  $(n!)^3$  (or  $(n!)^\omega$  if we use change of ordering for *Shape Position* ideals of Chapter 4). Moreover, we observe in practice that the system  $\mathcal{S}_{\mathfrak{S}_n}$  is regular (in the sense of Definition 2.79). Hence, we follow Hypothesis 5.18.

**Hypothesis 5.18.** *Let  $R$  be a fixed point of  $E(\mathbb{F}_{q^n})$  and  $S'_n(e_1, \dots, e_n)$  be the expression of  $S_{n+1}(x_1, \dots, x_n, x_R)$  in terms of the elementary symmetric polynomials. Polynomial systems arising from a Weil descent from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$  on  $S'_n(e_1, \dots, e_n)$  are regular.*

By consequence, under Hypothesis 5.18 from Theorem 2.83 and Theorem 2.85 and by using Macaulay and Bézout bounds (Corollary 2.76) the complexity of solving the system  $\mathcal{S}_{\mathfrak{S}_n}$  is bounded by

$$O\left(n \binom{n2^{n-1} + 1}{n}^\omega + n2^{3n(n-1)}\right) = O\left(n2^{3n(n-1)}\right)$$

arithmetic operations in  $\mathbb{F}_q$  when using FGLM algorithm. Furthermore, we observe that the systems  $\mathcal{S}_{\mathfrak{S}_n}$  are actually in *Shape Position*. Hence, using algorithm for *Shape Position* ideals presented in Chapter 4 the complexity of solving  $\mathcal{S}_{\mathfrak{S}_n}$  is decreased to

$$O\left(n \binom{n2^{n-1} + 1}{n}^\omega + n2^{\omega n(n-1)}\right) = O\left(ne^{\omega n} 2^{\omega n(n-1)}\right)$$

---

<sup>1</sup>The notation  $\mathcal{S}_{\mathbb{G}}$  means that the system is expressed w.r.t. the change of variables associated to  $\mathbb{G}$  i.e. the change of variables formed by the primary and secondary invariants of  $\mathbb{F}_q[x_1, \dots, x_n]^{\mathbb{G}}$ .

arithmetic operations in  $\mathbb{F}_q$ . Moreover, let  $\mathcal{S}$  be the system obtain from the Weil descent applied on  $S_{n+1}(x_1, \dots, x_n, x_R)$ . As presented in Section 3.2.3 computing the solutions of  $\mathcal{S}$  from that of  $\mathcal{S}_{\mathfrak{S}_n}$  can be done very efficiently and is negligible in comparison of solving  $\mathcal{S}_{\mathfrak{S}_n}$ .

**Theorem 5.19** ([Gau09]). *The complexity of solving the elliptic curve discrete logarithm problem for curves defined over  $\mathbb{F}_{q^n}$  with  $n > 1$  can heuristically be bounded by*

$$O\left(qn e^{\omega n} 2^{\omega n(n-1)} n! + nq^2 \log(q)^2\right)$$

arithmetic operations in  $\mathbb{F}_q$ .

Obviously, this strategy can be applied to the variant “ $n - 1$ ”. Using this strategy, the authors of [JV13] obtain the following total complexity.

**Theorem 5.20** ([JV13]). *The complexity of solving the elliptic curve discrete logarithm in curves defined over  $\mathbb{F}_{q^n}$  with  $n > 1$  using the variant “ $n - 1$ ” can heuristically be bounded by*

$$O\left((n-1)! 2^{\omega(n-1)(n-2)} e^{\omega n} n^{-\frac{\omega}{2}} q^2\right)$$

arithmetic operations in  $\mathbb{F}_q$ .

Polynomial systems involved in the variant “ $n - 1$ ” are overdetermined. Indeed, we have  $n$  equations and  $n - 1$  variables. Hence, the arguments justifying the complexity of Theorem 5.20 are slightly different of those presented in Chapter 2. We now give some intuitions behind such a complexity.

Since the systems are overdetermined then in general they do not have any solution. In the particular case when the system admits a solution then in general it is unique. Consequently, most of the time is spent to test if the polynomial 1 is in the considered ideal. If it is the case then the system admits no solution and the point of the curve cannot be decomposed into  $n - 1$  points of the factor base.

**Definition 5.21** (Homogenized ideal). *We define the map  $\text{Hom}$  as*

$$\begin{aligned} \text{Hom} : \mathbb{K}[x_1, \dots, x_n] &\rightarrow \mathbb{K}[x_1, \dots, x_n, h] \\ f &\mapsto h^{\deg(f)} f\left(\frac{x_1}{h}, \dots, \frac{x_n}{h}\right). \end{aligned}$$

Let  $\mathcal{I} = \langle f_1, \dots, f_s \rangle \subset \mathbb{K}[x_1, \dots, x_n]$  be an affine ideal. We call the homogenized ideal of  $\mathcal{I}$  the homogeneous ideal  $\mathcal{J} \subset \mathbb{K}[x_1, \dots, x_n, h]$  defined by

$$\mathcal{J} = \langle \text{Hom}(f_1), \dots, \text{Hom}(f_s) \rangle.$$

In the literature, the term homogenized ideal denotes the ideal  $\{\text{Hom}(f) \mid f \in \mathcal{I}\}$ . However, we do not need this definition here.

Let  $\mathcal{S} = \{f_1, \dots, f_n\} \subset \mathbb{K}[x_1, \dots, x_{n-1}]$  be the system to solve. Let  $\mathcal{J}$  be the homogenized ideal of  $\langle \mathcal{S} \rangle$ . If the sequence  $(\text{Hom}(f_1), \dots, \text{Hom}(f_n))$  is regular then  $\mathcal{J}$  is of dimension zero since it has as many variables as equations. In that case, as shown in [BFSS13], testing if  $\mathcal{S}$  has a solution can be done by testing the consistency of a linear system involving the Macaulay matrix in a well chosen degree  $d$  containing all polynomials of the form  $tf_i$  with  $\deg(tf_i) \leq d$  and  $tf_i \in \mathbb{F}_q[x_1, \dots, x_{n-1}]$ . If the linear system is consistent then  $\mathcal{S}$  has no solution. In [BFSS13], it is shown that  $d$  is given by the degree of regularity of the homogenized ideal of

$\langle \mathcal{S} \rangle$ . Consequently, the Macaulay bound yields  $d \leq n2^{n-2} - n + 1$ . Therefore, testing if a point is decomposable into  $n - 1$  points of the factor base can be done in  $O(n e^{\omega n} 2^{\omega(n-1)(n-2)})$  arithmetic operations in  $\mathbb{F}_q$ .

When the system has one solution then the homogenized system is not of dimension zero. In that case, since it has as many variables as equations it cannot be a regular system. However, we can find the linear relations describing the unique solution in a similar way that we test if the system admits a solution. Nevertheless, in that case, there is no precise bound on the degree to consider. In practice, we observe that this degree is almost the same as in the case where the system has no solution. Hence, by assuming this hypothesis we obtain a similar complexity of that of Theorem 5.20. The difference resides in the asymptotic simplification of the size of the Macaulay matrix.

One of the bottleneck of the index calculus attack to solve the elliptic curve discrete logarithm is the computation of the summation polynomials. Indeed, from  $m > 5$  the  $m$ th summation polynomial of Weierstrass curves has never been computed. Hence, having an efficient way of computing them becomes a challenge to improve this kind of attack.

In this context, the author of [JV13] suggest to compute them directly expressed in terms of the elementary symmetric polynomials.

### 5.6.2 Computation of summation polynomials

The classical strategy to compute summation polynomials in terms of the elementary symmetric polynomials is first to compute them in their classical form. Then, to express them in terms of the elementary symmetric polynomials one uses for instance Gröbner bases computations or the well-known algorithm dedicated to the symmetric group (see for instance [Stu08]). However, expressing the summation polynomials in terms of the elementary symmetric polynomials allows to decrease their degree and their density. Moreover, we have seen in Section 5.4 that summation polynomials are computed recursively. Consequently, it would be appropriate to use the action of the symmetric group to speed up the computation of the summation polynomials. That is to say, we express the summation polynomials in terms of the elementary symmetric polynomials throughout their computation instead only at the end.

Following [JV13] spreading the symmetrization throughout the computation of the  $m$ th summation polynomial can be done by noting that

$$\begin{cases} e_1 &= e'_1 + x_n \\ e_2 &= e'_2 + x_n e'_1 \\ e_3 &= e'_3 + x_n e'_2 \\ &\vdots \\ e_{n-1} &= e'_{n-1} + x_n e'_{n-2} \\ e_n &= e'_{n-1} x_n \end{cases} \quad (5.20)$$

where  $e'_i$  is the  $i$ th elementary symmetric polynomial in terms of  $x_1, \dots, x_{n-1}$ . In the context of solving the PDP, one is interested in computing  $S'_{n+1}(e_1, \dots, e_n, x_{n+1})$ , the expression of the  $S_{n+1}(x_1, \dots, x_n, x_{n+1})$  in terms of the elementary symmetric polynomials. From Theorem 5.10 we have

$$S_{n+1}(x_1, \dots, x_n, x_{n+1}) = \text{Res}_X(S_n(x_1, \dots, x_{n-1}, X), S_3(x_n, x_{n+1}, X)).$$

Thus, assume that we know  $S'_n(e'_1, \dots, e'_{n-1}, x_n)$  then

$$F = \text{Res}_X(S'_n(e'_1, \dots, e'_{n-1}, X), S_3(x_{n-1}, x_n, X))$$

is the expression of  $S_{n+1}(x_1, \dots, x_{n+1})$  in terms of  $e'_1, \dots, e'_{n-1}, x_n, x_{n+1}$ . Hence, computing  $S'_{n+1}(e_1, \dots, e_n, x_{n+1})$  is reduced to apply the change of variables given in equation (5.20) to  $F$ .

The next two chapters are devoted to highlight some families of elliptic curves which have adding symmetries and use them to speed up the computation of summation polynomials or the PDP solving.

## CHAPTER 6

# Point decomposition problem in high characteristic

---

### Contents

|            |                                                                         |            |
|------------|-------------------------------------------------------------------------|------------|
| <b>6.1</b> | <b>Impact of the elliptic curve representation on the PDP solving .</b> | <b>132</b> |
| <b>6.2</b> | <b>Impact of a 2-torsion subgroup on the PDP solving . . . . .</b>      | <b>133</b> |
| 6.2.1      | Action of the 2-torsion on the solutions of the PDP . . . . .           | 133        |
| 6.2.2      | Action of the 2-torsion on the polynomial systems modelling the PDP     | 136        |
| <b>6.3</b> | <b>Action of the 4-torsion on the PDP . . . . .</b>                     | <b>139</b> |
| 6.3.1      | Twisted Edwards curve . . . . .                                         | 139        |
| 6.3.2      | Universal Edwards model of elliptic curves . . . . .                    | 139        |
| 6.3.3      | Twisted Jacobi intersection curve . . . . .                             | 139        |
| <b>6.4</b> | <b>Experimental results and security estimates . . . . .</b>            | <b>141</b> |
| 6.4.1      | Experiments with $n = 4$ . . . . .                                      | 141        |
| 6.4.2      | Experiments for $n = 5$ and $n = 6$ . . . . .                           | 143        |
| 6.4.3      | Security level estimates . . . . .                                      | 145        |

---

*The results presented in this chapter are from a joint work with J.-C. Faugère, P. Gaudry and G. Renault*

Using the double large prime variation and for a fixed degree extension  $n$ , the complexity of the index calculus attack of Gaudry presented in Chapter 5 is  $\tilde{O}(q^{2-\frac{2}{n}})$  where the notation  $\tilde{O}$  means that we omit the logarithmic factors in  $q$ . It is thus faster than Pollard rho method in  $\tilde{O}(q^{\frac{n}{2}})$  for  $n \geq 3$  and sufficiently large  $q$ . However, this complexity hides an exponential dependence in  $n$  due to the resolution of the PDP problem, which is the main topic of this chapter. Let us recall the PDP.

**Point Decomposition Problem (PDP).** *Given a point  $R$  in an elliptic curve  $E(\mathbb{F}_{q^n})$  and a factor base  $\mathcal{F} \subset E(\mathbb{F}_{q^n})$ , find, if they exist,  $P_1, \dots, P_n$  in  $\mathcal{F}$ , such that*

$$R = P_1 \oplus \dots \oplus P_n.$$

To solve the PDP, as shown in Chapter 5 one can choose  $\mathcal{F}$  with an algebraic structure and the summation polynomials introduced by Semaev [Sem04]. The resolution of the PDP is then equivalent to solve a polynomial system. Following Algorithm 11, this can be done by first computing a Gröbner basis of the system for a degree ordering with  $F_4$  [Fau99] or  $F_5$  [Fau02], see Chapter 2. Then computing the lexicographical Gröbner basis by using a change of ordering algorithm [FGLM93, FM11, FM13, FGHR12a, FGHR13a], see Chapter 2 and Chapter 4.

We note that Nagao [Nag10] introduced a variant of the index calculus algorithm, well-suited to hyperelliptic curves, in which the PDP step is replaced by another approach that creates relations from Riemann-Roch spaces. It also relies, in the end, on polynomial system solving. If the curve is elliptic, the Nagao variant needs to solve polynomial systems with a number of variables quadratic in  $n$  instead of  $n$  variables with the summation polynomials of Semaev. Therefore, in the elliptic case, it seems to be always better to use Semaev's polynomials, so we stick to that case in our study.

## Contributions

In the case of the Pollard rho and sibling methods, it is well-known that if there is a small rational subgroup in  $\mathbb{G}$ , the Pohlig-Hellman reduction allows to speed-up the computation by a factor of roughly the square root of the order of this subgroup. It is also the case if there is an explicit automorphism of small order. For index calculus in general, it is far less easy to make use of such an additional structure. For instance, in the multiplicative group of a prime finite field, the number field sieve algorithm must work in the full group, even if one is interested only in the discrete logarithm in a subgroup. A key element is the action of the rational subgroup that must be somewhat compatible with the factor base. See for instance the article by Couveignes and Lercier [CL08], where a factor base is chosen especially to fit this need, again in the context of multiplicative groups of finite fields.

The aim of this chapter is to emphasize some elliptic curves models where one can indeed make use of the presence of a small rational subgroup to speed-up the index calculus algorithm, and especially the PDP step. In particular, for curve representations having an important interest from a cryptographic point of view, we decrease the bound on the complexity by a factor of  $2^{\omega(n-1)}$ . More precisely, under the hypothesis that the systems are regular *i.e.* Hypothesis 5.18, we have the following result.

**Theorem 6.1.** *Let  $E$  be an elliptic curve defined over a non binary field  $\mathbb{F}_{q^n}$  where  $n > 1$ . If  $E$  can be put in universal Edwards model or twisted Edwards or twisted Jacobi intersection representation then the complexity of solving the PDP is*

- (proven complexity)  $\tilde{O}\left(n \cdot 2^{3(n-1)^2}\right)$
- (heuristic complexity)  $\tilde{O}\left(ne^{\omega n} \cdot 2^{\omega(n-1)^2}\right)$

where the notation  $\tilde{O}$  means that we omit logarithmic factors in  $q$ .

This result can be compared to the complexity of solving the PDP in the general case (*i.e.*  $E$  cannot be put in the representation mentioned in the above theorem). From Section 5.6.1 Theorem 5.19, this complexity is in  $\tilde{O}(n2^{3n(n-1)})$  (or  $\tilde{O}(ne^{\omega n}2^{\omega n(n-1)})$  in the heuristic case).

The proven complexity of Theorem 6.1 is obtained by using the classical complexity of change of ordering algorithm, FGLM in  $O(nD^3)$  [FGLM93] where  $D$  is the number of solutions counted with multiplicities in the algebraic closure of the coefficient field. The heuristic complexity is obtained by using the change of ordering algorithm for *Shape Position* ideals proposed in Chapter 4.

The main ingredient of the proof of Theorem 6.1 is to use the symmetries of the curves corresponding to a group action: they allow to reduce the number of solutions in  $\overline{\mathbb{F}_q}$  of the polynomial systems to be solved and to speed up intermediate Gröbner bases computations.

As presented in Chapter 5, the first symmetries to be used are inherent in the very definition of the PDP: the ordering of the  $P_i$ 's does not change their sum, so that the full symmetric group acts naturally on the polynomial system corresponding to the PDP. It is a classical way to reduce the number of solutions by a factor  $n!$ , and speed up accordingly the resolution.

Twisted Edwards, twisted Jacobi intersection curves and universal Edwards model of elliptic curves have more symmetries than ordinary elliptic curves, due to the presence of a rational 2-torsion point with an interesting action. It is remarkable that, for the natural choice of the factor base, this action translates into the polynomial systems constructed using summation polynomials in a very simple manner: any sign change on an even number of variables is allowed. This action combined with the full symmetric group gives the so-called dihedral Coxeter group, see Chapter 3 or for instance [Kan01]. Using invariant theory techniques [Stu08], we can thus express the system in terms of adapted coordinates, and therefore the number of solutions is reduced by a factor  $2^{n-1} \cdot n!$  (the cardinality of the dihedral Coxeter group). This yields a speed-up by a factor  $2^{3(n-1)}$  (or  $2^{\omega(n-1)}$  for the heuristic case) in the change of ordering step, compared to the general case.

Let denote by  $\mathcal{S}$  the system obtained from summation polynomial by applying the Weil restriction from  $\mathbb{F}_{q^n}$  to  $\mathbb{F}_q$  (see Section 5.5.1). We denote by  $\mathcal{S}_{\mathfrak{S}_n}$  (respectively  $\mathcal{S}_{D_n}$ ) the expression of  $\mathcal{S}$  in terms of the elementary symmetric polynomials (respectively the primary invariants of the dihedral Coxeter group). Since the symmetric group is a subgroup of the dihedral Coxeter group, from Lemma 3.24 there exists a change of variables  $\rho_1, \dots, \rho_n$  to express  $\mathcal{S}_{\mathfrak{S}_n}$  in terms of the primary invariants of  $D_n$ . Consequently, under Hypothesis 5.18, results of Chapter 3 (particularly Corollary 3.25) allows to conclude that using the 2-torsion of twisted Edwards or Jacobi intersection curves or universal Edwards model of elliptic curves, the bound on the complexity of computing a Gröbner basis for a degree monomial ordering is divided by  $2^{\omega(n-1)}$ .

We present also several practical experiments which confirm the exponential decrease of the complexity. All experiments were carried out using the computer algebra system MAGMA [BCP97] and the FGb library [Fau10].

## Consequences and limitations

Our experiments show that for some parameters, the new version of the algorithm is significantly faster than generic algorithms. For instance for a twisted Edwards or twisted Jacobi intersection curve defined over  $\mathbb{F}_{q^5}$  where  $\log_2(q) = 64$ , solving the ECDLP with generic algorithms requires approximately  $2^{160}$  operations in  $E(\mathbb{F}_{q^5})$  and only  $2^{130}$  basic arithmetic operations (multiplications of two 32-bits words) with our approach.

We do not change the very nature of the attack; therefore it applies only to curves defined over small extension fields. This work has no implication on the ECDLP instances recommended by the NIST [Nat09], since they are defined over prime finite fields of high characteristic or binary fields of prime degree extension.

## Related work

The purpose of our work as that of the variant “ $n - 1$ ” of Joux and Vitse [JV13] presented in Chapter 5 is to decrease the running time to solve polynomial systems modelling the PDP problem. The difference between the two approaches is that in our case we do not decrease the probability of decomposing a point. Hence, while the variant “ $n - 1$ ” is interesting for medium

$q$  our work does not limit the size of  $q$ . Nevertheless, it limits to curves having particular torsion subgroup. However, these two approaches are compatible and we will show that by combining these two methods one can for the first time tackle instances of the PDP with  $n = 6$ .

Throughout this chapter the field  $\mathbb{F}_q$  is assumed to be of characteristic greater than 3.

## 6.1 Impact of the elliptic curve representation on the PDP solving

In this section, we compare for different representations, the solving of the PDP problem using summation polynomials as suggested in [Gau09]. That is to say we take into account the action of the symmetric group as presented in Section 5.6.1. We consider the practical solving (using MAGMA) of the polynomial systems  $\mathcal{S}_{\mathfrak{S}_n} \subset \mathbb{F}_q[e_1, \dots, e_n]$  for four families of elliptic curves: Weierstrass curve, universal Edwards model of elliptic curves, twisted Edwards curves and twisted Jacobi intersection curves.

We are able to solve these systems for  $n = 2, 3, 4$ . For  $n = 2$  or 3 the resolution is instantaneous for all curve representations. In the following, we present some practical results for  $n = 4$  obtained by using the computer algebra system MAGMA (v2.19-7) on one core of a 2.00GHz Intel<sup>®</sup> E7540 CPU.

| $\log_2(q)$ |                     | $F_4$ (s) | Change-Order (s) | Total time (s) |
|-------------|---------------------|-----------|------------------|----------------|
| 16          | Weierstrass [Gau09] | 5         | 496              | 501            |
|             | Edwards             | < 1       | 212              | 213            |
|             | Jacobi              | < 1       | 272              | 273            |
|             | Universal Edwards   | < 1       | 190              | 191            |
| 64          | Weierstrass [Gau09] | 342       | 6317             | 6659           |
|             | Edwards             | 6         | 1458             | 1464           |
|             | Jacobi              | 8         | 1675             | 1683           |
|             | Universal Edwards   | 5         | 1426             | 1431           |

We note that for twisted Edwards or Jacobi intersection curves or universal Edwards model of elliptic curves the running time of the system resolution is equivalent and significantly smaller than for Weierstrass representation. This can be explained by the particular shapes of the lexicographical Gröbner basis :

Lexicographical Gröbner basis of  
 $\langle \mathcal{S}_{\mathfrak{S}_n} \rangle$  for Weierstrass representation  
 :

$$\begin{cases} e_1 + h_1(e_n) \\ e_2 + h_2(e_n) \\ \vdots \\ e_{n-2} + h_{n-2}(e_n) \\ e_{n-1} + h_{n-1}(e_n) \\ h_n(e_n) \end{cases}$$

Lexicographical Gröbner basis of  
 $\langle \mathcal{S}_{\mathfrak{S}_n} \rangle$  for twisted Edwards / Jacobi  
 intersection representations and  
 universal Edwards model of elliptic  
 curves:

$$\begin{cases} e_1 + \mathfrak{p}_1(e_{n-1}, e_n) \\ e_2 + \mathfrak{p}_2(e_{n-1}, e_n) \\ \vdots \\ e_{n-2} + \mathfrak{p}_{n-2}(e_{n-1}, e_n) \\ \mathfrak{p}_{n-1}(e_{n-1}, e_n) \\ \mathfrak{p}_n(e_n) \end{cases}$$

where  $\deg(h_n) = 2^{n(n-1)}$ ,  $\deg(\mathbf{p}_n) = 2^{(n-1)^2}$ ,  $\deg_{e_{n-1}}(\mathbf{p}_{n-1}) = 2^{n-1}$  and for all curve representations  $\mathbf{V}_{\mathbb{F}_q}(\mathcal{S}_{\mathfrak{S}_n}) = 2^{n(n-1)}$ .

**Remark 6.2.** *The form of the lexicographical Gröbner basis is given here in order to explain some intuition of our approach.*

The gain of efficiency observed in the case of twisted Edwards, twisted Jacobi intersection curves and universal Edwards model of elliptic curves is due to the smaller degree appearing in the computation of Gröbner basis of  $\mathcal{S}_{\mathfrak{S}_n}$  in comparison with the Weierstrass case. Note that the lexicographical Gröbner bases for Weierstrass representation is in *Shape Position*. That is to say, to find the solutions of the system from the lexicographical Gröbner basis, we need to factor only one univariate polynomial in the smallest variable. The value of the others variables is obtained when the value of the smallest variable is fixed. In this case, the smallest variable, here  $e_n$ , is said to be separating (see for instance [CCS11]). This means that any element in the variety of the ideal generated by  $\mathcal{S}_{\mathfrak{S}_n}$  is distinguishable by  $e_n$ . Contrary to Weierstrass representation, the lexicographical Gröbner bases for twisted Edwards, twisted Jacobi intersection curves and universal Edwards model of elliptic curves are not in *Shape Position*. The variable  $e_n$  is not separating for these three representations. In fact, for each solution of the system, there are  $2^{n-1} - 1$  others solutions with same value in  $e_n$ . By consequence, one would like to find a larger group than  $\mathfrak{S}_n$  acting on the system (and thus on the variety of solutions) such that each orbit gathers all such solutions with the same value in  $e_n$ . In the next section, we show how to use such a larger group related to 2-torsion points in order to increase the efficiency of the computation.

## 6.2 Impact of a 2-torsion subgroup on the PDP solving

In this section, we show how a 2-torsion subgroup can act on the *point decomposition problem*. First, we discuss about the action of the 2-torsion on the solutions of the PDP. Then, we will show how this action is translated to polynomial systems modelling the PDP. In particular, we show that the choice of the elliptic curve representation is crucial. More precisely, in order to take advantage of the action of the 2-torsion subgroup, its action must be simple enough.

As mentioned in Chapter 5, depending on the curve representation, the coordinate chosen for the projection can be  $x$ ,  $y$  or  $z$ . For more generality, here we note the chosen coordinate  $c$  and the  $(n+1)^{\text{th}}$  summation polynomial evaluated in one variable in the  $c$ -coordinate of a point  $R$  of the curve is denoted  $S_{n+1}^R$ . The notation  $c(P)$  denotes the  $c$ -coordinate of the point  $P$ . Let  $\mathcal{F}_i = \left\{ P \in E(\mathbb{F}_{q^n}) \mid \frac{c(P)}{\alpha^i} \in \mathbb{F}_q \right\}$  for any  $i = 0, \dots, n-1$  where  $\alpha$  is a generator of  $\mathbb{F}_{q^n}$ . For Weierstrass, twisted Edwards representations or universal Edwards model of elliptic curves, we take as factor base  $\mathcal{F} = \mathcal{F}_0$ . For Jacobi intersection curves, if  $\mathbb{F}_q$  is a prime field then  $\mathcal{F}_0$  contains only the 2-torsion of the curves; hence it does not contain enough points to be used as factor base. Therefore, for this representation we take as factor base  $\mathcal{F} = \mathcal{F}_1$ .

### 6.2.1 Action of the 2-torsion on the solutions of the PDP

Suppose that we have a solution  $(P_1, P_2, \dots, P_n)$  to the PDP, and denote by  $T_2$  a 2-torsion point. Thus for all  $k = 1, \dots, \left\lfloor \frac{n}{2} \right\rfloor$  we have  $P_1 \oplus \dots \oplus P_n \oplus [2k]T_2 = R$ . Therefore, from one

decomposition of  $R$  (modulo the order) we have in fact  $\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2k} = 2^{n-1}$  decompositions of  $R$  obtained by adding an even number of times a 2-torsion point :

$$\begin{aligned}
R &= P_1 \oplus \cdots \oplus P_n \\
&= (P_1 \oplus T_2) \oplus (P_2 \oplus T_2) \oplus P_3 \oplus \cdots \oplus P_n \\
&= (P_1 \oplus T_2) \oplus P_2 \oplus (P_3 \oplus T_2) \oplus P_4 \oplus \cdots \oplus P_n \\
&\vdots \\
&= P_1 \oplus \cdots \oplus P_{n-2} \oplus (P_{n-1} \oplus T_2) \oplus (P_n \oplus T_2) \\
&= (P_1 \oplus T_2) \oplus (P_2 \oplus T_2) \oplus (P_3 \oplus T_2) \oplus (P_4 \oplus T_2) \oplus P_5 \oplus \cdots \oplus P_n \\
&\vdots
\end{aligned}$$

In general, these decompositions do not correspond to solutions of the PDP, since  $(P_i + T_2)$  is not always in the factor base  $\mathcal{F}$ . If the action of the 2-torsion point leaves invariant the factor base  $\mathcal{F}$  *i.e.*  $P \in \mathcal{F}$  implies that  $P \oplus T_2 \in \mathcal{F}$  then the 2-torsion point can be used to reduce the size of the factor base (see Remark 6.4). By consequence, if we know a decomposition of  $R$  w.r.t. the factor base  $\mathcal{F}$  (respectively a solution of the polynomial system to solve for solving the PDP) we can construct  $2^{n-1}$  decompositions of  $R$  w.r.t.  $\mathcal{F}$  (respectively  $2^{n-1}$  solutions of the polynomial system).

Let  $c$  and  $c_2$  be respectively the  $c$ -coordinate of  $P$  and  $P \oplus T_2$ . The action of the 2-torsion point leaves the factor base invariant if

$$\begin{cases} c_2 = \frac{p_1(c)}{p_2(c)} \text{ with } p_1, p_2 \in \mathbb{F}_q[c] & \text{if } \mathcal{F} = \mathcal{F}_0 \\ c_2 = \beta c + \gamma \text{ with } \beta \in \mathbb{F}_q \text{ and } \frac{\gamma}{\alpha^i} \in \mathbb{F}_q & \text{if } \mathcal{F} = \mathcal{F}_i, 1 \leq i < n \end{cases} \quad (6.1)$$

where  $\alpha$  is a generator of  $\mathbb{F}_{q^n}$ . The difference between the two cases is due to when  $\mathcal{F} = \mathcal{F}_0$  the  $c$ -coordinates of the points in the factor base are in a field whereas when  $\mathcal{F} = \mathcal{F}_i$  with  $i > 0$  the  $c$ -coordinates of the points in the factor base are in a vector space.

By consequence, if condition (6.1) is satisfied then the size of the factor base can be reduced. Moreover, we can *a priori* use the action of the 2-torsion to speed up the polynomial systems solving step in the PDP solving. Nevertheless, in order to use the action of the 2-torsion point in the polynomial system solving process, we need that  $c_2$  depends only on  $c$  and that the action of  $T_2$  on the coordinates is not too much complicated. The simplest being a linear action.

### Weierstrass curves

For Weierstrass representation, the 2-torsion points of  $E(\mathbb{F}_{q^n})$  are  $T_2 = (X, 0)$  where  $X$  is a root of  $X^3 + a_4X + a_6 = 0$  and we have

$$P \oplus T_2 = \left( \frac{x^3 + a_4x + a_6}{(X - x)^2} - x - X, \frac{(2x + X)y}{(x - X)} - \frac{y^3}{(x - X)^3} - y \right).$$

In this representation, we project the PDP on  $x$ -coordinate. As the  $x$ -coordinate of the point  $P \oplus T_2$  does not verify any of the equalities in (6.1), the 2-torsion points cannot be used to decrease the factor base. Moreover, the action of the 2-torsion points is not easy to handle in the polynomial systems solving process.

### Twisted Edwards curves

In the case of twisted Edwards representation, the 2-torsion point of a twisted Edwards curve is  $T_2 = (0, -1)$  and  $P \oplus T_2 = (-x, -y)$ . Thus the action of the 2-torsion point leaves invariant the factor base and the  $2^{n-1}$  decompositions of the point  $R$  translate into as many solutions of the PDP. Furthermore, the action of the 2-torsion point being very simple (*i.e.* linear) we can use it to decrease the number of solutions in the polynomial systems solving process.

### Universal Edwards model of elliptic curves

Universal Edwards model of elliptic curves have three 2-torsion point  $T_2 = (2t, 1), (-\frac{1}{2t}, -1)$  and  $(-2t, -1)$ . Moreover, if  $P = (x, y)$  we have  $P \oplus T_2 = (\frac{1}{x}, \frac{1}{y}), (-x, -y), (-\frac{1}{x}, -\frac{1}{y})$ . For this representation of elliptic curves, the factor base is  $\mathcal{F} = \mathcal{F}_0$ . Hence, these three 2-torsion points leave the factor base invariant and can thus be used to decrease the size of the factor base (Remark 6.4). Nevertheless, the action of the points  $(2t, 1)$  and  $(-2t, -1)$  is not linear and by consequence do not fit into the theory presented in Chapter 3 Section 3.2. However, as for twisted Edwards curves the linear action of the point  $(-\frac{1}{2t}, -1)$  fits into this theory and can thus be used to decrease the number of solutions in the polynomial systems solving process.

### Twisted Jacobi intersection curves

Finally for twisted Jacobi intersection representation, the three 2-torsion points of a twisted Jacobi intersection curve are  $T_2 = (0, 1, -1), (0, -1, 1), (0, -1, -1)$ . Thus we have  $P \oplus T_2 = (-x, y, -z), (-x, -y, z), (x, -y, -z)$  and similarly to the twisted Edwards curves, the decompositions mentionned above should correspond to solutions of the system associated to the decomposition of the point  $R$ .

Obviously, as Jacobi intersection curves have three 2-torsion points, the factor base can be further decreased and from one decomposition of  $R$  one can construct more than  $2^{n-1}$  decompositions of  $R$ . However, since after projection on the  $c$ -coordinate ( $y$  or  $z$ ) for any 2-torsion points,  $c_2 = \pm c$  these decompositions will match with only  $2^{n-1}$  solutions of the system we want to solve.

As a consequence, for twisted Edwards, Jacobi intersection curve and universal Edwards model of elliptic curves from one solution of the polynomial system  $(c_1, \dots, c_n)$  corresponding to the decomposition  $R = P_1 \oplus \dots \oplus P_n$ , we can construct  $2^{n-1}$  solutions of the system by applying an even number of sign changes. Obviously, each of these solutions can be the projection of many decompositions. Hence, from one solution  $(c_1, \dots, c_n)$  of  $S_{n+1}^R$ , we have not only  $n!$  solutions coming from  $\mathfrak{S}_n$  (see Section 5.6.1) but  $n! \cdot 2^{n-1}$  : all  $n$ -tuples formed by  $(c_1, \dots, c_n)$  to which we apply an even number of sign changes and a permutation of  $\mathfrak{S}_n$ , that is the orbit of  $(c_1, \dots, c_n)$  under the action of the Coxeter group  $D_n$  introduced in Section 3.2.1.

**Remark 6.3.** *Same reasoning works mutatis mutandis on the solutions of the  $m$ th summation polynomial (*i.e.* without evaluating one variable). By consequence, the solutions of the  $m$ th summation polynomial  $S_m$  are invariant under the action of the dihedral Coxeter group  $D_m$ .*

**Remark 6.4.** *In addition to speeding up the resolution of the polynomial systems, the use of the 2-torsion points of twisted Edwards, Jacobi intersection curves and universal Edwards model of elliptic curves allows to further decrease the size of the factor base by keeping the*

same probability of decomposition. More precisely, we follow the idea in [Gau09] presented in Section 5.5.1 to divide the size of the factor base by 2 without decreasing the probability of decomposing a point.

Let us write  $\mathcal{F} = S_1 \cup S_2$  such that for all  $P \in \mathcal{F}$ ,  $S_1$  contains a representative of the orbit of  $P$  under the action of  $\iota$  and the two torsion of the curve and  $S_2$  contains all the others points in the orbit of  $P$ . Finally, we take as factor base  $S_1$  of size  $\sim \frac{q}{4}$  for twisted Edwards curves since they have a unique 2-torsion point which leaves the factor base invariant and  $\sim \frac{q}{8}$  for twisted Jacobi intersection curves and universal Edwards model of elliptic curves since they have three 2-torsion points which leave the factor base invariant.

## 6.2.2 Action of the 2-torsion on the polynomial systems modelling the PDP

If a linear group acts on the variety of a polynomial system, there is no guarantee that the system is in the invariant ring of the linear group. In our case, the system obtained from  $S_{n+1}^R$  by a Weil restriction is invariant under the action of  $D_n$  and we have the following result.

**Proposition 6.5.** *For universal Edwards model of elliptic curves, twisted Jacobi intersection curves and twisted Edwards curves defined over a field  $\mathbb{K}$  we have for any  $m \geq 3$*

$$S_m(c_1, \dots, c_m) \in \mathbb{K}[c_1, \dots, c_m]^{D_m}.$$

As a consequence,  $S_{n+1}^R(c_1, \dots, c_n) \in \mathbb{K}[c_1, \dots, c_n]^{D_n}$ .

The idea of the proof is to use the relations between generators of the dihedral Coxeter group to show that these generators leave  $S_m$  invariant. First we use the action of the linear group  $D_m$  on the solutions of  $S_m$  to underline that for any  $g$  in  $D_m$ , the action of  $g$  on  $S_m$  leaves it invariant, up to a multiplicative factor  $h_g \in \mathbb{K}$ . Then we use that  $D_m$  is generated by elements of order 2, relations between generators of  $D_m$  and that  $D_m$  contains  $\mathfrak{S}_m$  to show that  $h_g = \pm 1$  and  $h_g = h_{g'}$  for all elements  $g$  and  $g'$  in  $D_m$ . Finally we use the recursive construction of summation polynomials to show that one generator of  $D_m$  leaves  $S_m$  invariant and consequently that  $D_m$  leaves  $S_m$  invariant.

*Proof.* The summation polynomials are irreducible hence  $\langle S_m \rangle = \sqrt{\langle S_m \rangle}$ . The solutions of  $S_m$  are invariant by the action of  $D_m$  thus for all  $g \in D_m$ ,  $g \cdot S_m$  vanishes in all solutions of  $S_m$ . Consequently for all  $g \in D_m$ ,  $g \cdot S_m \in \langle S_m \rangle$  and so  $g \cdot S_m = h_g \cdot S_m$  where  $h_g \in \mathbb{K}[c_1, \dots, c_m]$ . The group  $D_m$  is a linear group hence for all  $g \in D_m$ ,  $\deg(g \cdot S_m) = \deg(S_m)$  thus  $h_g \in \mathbb{K}^\times$ .

Let  $\phi : D_m \rightarrow \mathbb{K}^\times$  be the application which maps  $g$  to  $h_g$  as defined above. Clearly, this application is a group morphism and thus  $\phi(g)^o = h_g^o = 1$  where  $o$  is the order of  $g$ .

We note  $\tau_{i,j}$  the transposition which swaps the elements in position  $i$  and  $j$ . Let  $\mathcal{B} = \{\tau_{i,i+1} \mid i = 1, \dots, m-1\}$  be a basis of  $\mathfrak{S}_m$ . A transposition is of order two and all the transpositions are conjugated, hence  $\phi(\tau_{i,j}) = \phi(\tau_{k,\ell}) \in \{-1, 1\}$  for all  $i, j, k, \ell \in \{1, \dots, n\}$ .

We now show, by induction, that  $S_m$  is invariant under the permutation  $\tau_{1,2}$ . Clearly (see Section 5.4),  $S_3$  is invariant under  $\tau_{1,2}$ . Let  $k > 2$ , assume that  $S_k$  is invariant under  $\tau_{1,2}$ . We have

$$\begin{aligned} S_{k+1} &= \text{Res}_X \left( S_k(c_1, \dots, c_{k-1}, X), S_3(c_k, c_{k+1}, X) \right) \\ &= \text{Det} \left( \text{Syl}_X \left( S_k(c_1, \dots, c_{k-1}, X), S_3(c_k, c_{k+1}, X) \right) \right) \end{aligned}$$

where  $\text{Syl}_X(p_1, p_2)$  is the Sylvester matrix of  $p_1$  and  $p_2$  w.r.t. the variable  $X$ . The Sylvester matrix of  $S_k(c_1, \dots, c_{k-1}, X)$  and  $S_3(c_k, c_{k+1}, X)$  w.r.t.  $X$  is stable by permutation of  $c_1$  and  $c_2$  (induction hypothesis). Hence its determinant too and  $S_{k+1}$  also. Consequently,  $S_m$  is invariant under  $\tau_{1,2}$  for all  $m \geq 3$ . Thus  $h_\tau = 1$  for all  $\tau \in \mathcal{B}$ . This confirms that the summation polynomials are symmetric.

A basis of  $D_m$  is given by  $\mathcal{A} = \mathcal{B} \cup (-1, -2)$  where  $(-1, -2)$  denotes the sign changes of the first two elements. The element  $(-1, -2)$  is of order 2 hence  $h_{(-1, -2)} = \pm 1$ . Let  $g = (-1, -2) \cdot \tau_{2,3} \cdot \tau_{1,2}$ ,  $g$  is of order 3 thus  $h_g^3 = 1 = (h_{\tau_{1,2}} \cdot h_{\tau_{2,3}} \cdot h_{(-1, -2)})^3 = h_{(-1, -2)}^3$ . Consequently for all elements  $g$  in  $\mathcal{A}$ ,  $h_g = 1$  and so  $S_m$  is invariant under  $D_m$ .  $\square$

As previously announced in Section 3.2.1, by assuming that  $q \neq 2^k$  then  $\mathbb{F}_{q^n}[c_1, \dots, c_n]^{D_n}$  is a polynomial algebra of basis  $\{s_1, \dots, s_{n-1}, e_n\}$  (or  $\{p_2, \dots, p_{2(n-1)}, p_n\}$ ). Hence, there exists a unique polynomial  $g_n^R \in \mathbb{F}_{q^n}[s_1, \dots, s_{n-1}, e_n]$  (respectively  $\mathbb{F}_{q^n}[p_2, \dots, p_{2(n-1)}, p_n]$ ) such that  $g_n^R$  is the expression of  $S_{n+1}^R$  in terms of the primary invariants  $\{s_1, \dots, s_{n-1}, e_n\}$  (respectively  $\{p_2, \dots, p_{2(n-1)}, p_n\}$ ). By applying a Weil restriction on  $g_n^R$  we obtain a new system  $\mathcal{S}_{D_n} \subset \mathbb{F}_q[s_1, \dots, s_{n-1}, e_n]$  (respectively  $\mathbb{F}_q[p_2, \dots, p_{2(n-1)}, p_n]$ ) with  $n$  variables and  $n$  equations. The degree of  $\langle \mathcal{S}_{D_n} \rangle$  can be bounded by

$$\frac{\deg(\langle \mathcal{S} \rangle)}{\#D_n} = \frac{\deg(\langle \mathcal{S} \rangle)}{n! \cdot 2^{n-1}} = \frac{\deg(\langle \mathcal{S}_{\mathfrak{S}_n} \rangle)}{2^{n-1}} = \frac{2^{n(n-1)}}{2^{n-1}} = 2^{(n-1)^2}.$$

To estimate an explicit complexity bound on the resolution of the *Point Decomposition Problem* we need to assume that the system  $\mathcal{S}_{\mathfrak{S}_n}$  is regular. This property for  $\mathcal{S}_{\mathfrak{S}_n}$  has been verified on all experiments we did (see Table 6.1). Moreover, a similar hypothesis was already done for the same kind of systems in [JV13] (see Section 5.6.1). Hence, it is reasonable to assume it and we still follow Hypothesis 5.18. We can note that Hypothesis 5.18 implies Hypothesis 5.15 about the dimension of the ideal. We have therefore obtained our main theorem.

**Theorem 6.6.** *In twisted Edwards (respectively twisted Jacobi intersection or universal Edwards model) representation under the Hypothesis 5.18, the Point Decomposition Problem can be solved in*

- (proven complexity)  $\tilde{O}\left(n \cdot 2^{3(n-1)^2}\right)$
- (heuristic complexity)  $\tilde{O}\left(ne^{\omega n} \cdot 2^{\omega(n-1)^2}\right)$

arithmetic operations in  $\mathbb{F}_q$ .

*Proof.* Since  $\mathfrak{S}_n \subset D_n$  from Lemma 3.24 there exists a change of variables  $\rho_1, \dots, \rho_n$  to express  $\mathcal{S}_{\mathfrak{S}_n}$  in terms of the primary invariants of  $D_n$ . That is to say,  $\rho_1, \dots, \rho_n$  is the change of variables to pass from  $\mathcal{S}_{\mathfrak{S}_n}$  to  $\mathcal{S}_{D_n}$ . By considering  $e_1, \dots, e_n$  (respectively  $s_1, \dots, s_{n-1}, e_n$ ) for the primary invariants of  $\mathfrak{S}_n$  (respectively  $D_n$ ) one can easily deduce that

$$\begin{cases} \rho_i = e_i^2 + 2 \sum_{j=1}^{i-1} (-1)^j e_{i-j} e_{i+j} + 2(-1)^i e_{2i} & \text{if } i \leq \lfloor n/2 \rfloor \\ \rho_i = e_i^2 + 2 \sum_{j=1}^{n-i} (-1)^j e_{i-j} e_{i+j} & \text{if } \lfloor n/2 \rfloor < i < n \\ \rho_n = e_n \end{cases}.$$

Moreover,  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are algebraically independent. Indeed, let us consider DRL ordering on  $\mathbb{K}[e_1, \dots, e_n]$  with  $e_1 >_{\text{drl}} \dots >_{\text{drl}} e_n$ . We have that  $\text{LT}_{>_{\text{drl}}}(\rho_i^{(h)}) = e_i^2$  and  $\text{LT}_{>_{\text{drl}}}(\rho_n^{(h)}) =$

$e_n$ . Hence, let  $\mathcal{I} = \langle \rho_1^{(h)}, \dots, \rho_n^{(h)} \rangle$  for all  $i = 1, \dots, n$  there exists an integer  $n_i > 0$  such that  $e_i^{n_i} \in \text{in}_{>\text{drl}}(\mathcal{I})$ . Thus,  $\mathcal{I}$  is of dimension zero. Consequently, since  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are homogeneous polynomials Theorem 2.75 implies that  $(\rho_1^{(h)}, \dots, \rho_n^{(h)})$  is a regular sequence. Hence, as mentioned in Chapter 3 from [Smi95, Theorem 6.2.1]  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are algebraically independent.

Consequently, under Hypothesis 5.18 and from Proposition 3.10 the arithmetic complexity of computing a WDRL Gröbner basis with weights system  $(2, \dots, 2, 1)$  of  $\mathcal{S}_{D_n}$  can be bounded by  $O\left(ne^{\omega n} \left(\frac{2^{n(n-1)}}{2^{n-1}}\right)^\omega\right) = O\left(ne^{\omega n} 2^{\omega(n-1)^2}\right)$ .

Given this Gröbner basis, from Proposition 3.12 computing the LEX Gröbner basis can be done in  $\tilde{O}\left(n \cdot 2^{3(n-1)^2}\right)$ . The heuristic complexity is obtained by using change of ordering algorithm for *Shape Position* ideals presented in Chapter 4 of complexity  $\tilde{O}\left(n^2 \cdot 2^{\omega(n-1)^2}\right)$ . Indeed, we observe that the randomization strategy is not needed to ensure the efficient computation of the multiplication matrix  $T_n$ . Thus, we do not break the quasi-homogeneous structure and the results of Chapter 3 and Chapter 4 can be combined. Note that we cannot use the complexity of the deterministic change of ordering since the degrees of the input equations depend on the number of variables  $n$ .

Moreover, as shown in Section 3.2.3 computing the solutions of the PDP given that of  $\mathcal{S}_{D_n}$  is negligible in comparison of computing the LEX Gröbner basis. As a consequence, it is straightforward that the change of ordering step dominates which concludes the proof.  $\square$

Considering the action of the dihedral Coxeter group reduces the lexicographical Gröbner basis – for twisted Edwards, Jacobi intersection curves and universal Edwards model of elliptic curves – which is now in *Shape Position*.

Lexicographical Gröbner basis of  
 $\langle \mathcal{S}_{\mathfrak{S}_n} \rangle :$

$$\begin{cases} e_1 + \mathfrak{p}_1(e_{n-1}, e_n) \\ e_2 + \mathfrak{p}_2(e_{n-1}, e_n) \\ \vdots \\ e_{n-2} + \mathfrak{p}_{n-2}(e_{n-1}, e_n) \\ \mathfrak{p}_{n-1}(e_{n-1}, e_n) \\ \mathfrak{p}_n(e_n) \end{cases}$$

Lexicographical Gröbner basis of  
 $\langle \mathcal{S}_{D_n} \rangle :$

$$\begin{cases} s_1 + h_1(e_n) \\ s_2 + h_2(e_n) \\ \vdots \\ s_{n-2} + h_{n-2}(e_n) \\ s_{n-1} + h_{n-1}(e_n) \\ h_n(e_n) \end{cases}$$

where

- $\deg(\langle \mathcal{S}_{\mathfrak{S}_n} \rangle) = 2^{n(n-1)}$  and  $\deg(\langle \mathcal{S}_{D_n} \rangle) = 2^{(n-1)^2}$
- $\deg_{e_{n-1}}(\mathfrak{p}_{n-1}) = 2^{n-1}$ ,  $\deg(\mathfrak{p}_n) = 2^{(n-1)^2}$  and  $\deg(h_n) = 2^{(n-1)^2}$ .

As expected the degree of the ideal is divided by the cardinality of  $D_n$ ,  $2^{n-1} \cdot n!$  instead of  $n!$  when taking into account only the symmetric group.

In Section 6.4 we will show some experimental results which confirm that considering the action of the 2-torsion points significantly simplifies the resolution of the PDP.

### 6.3 Action of the 4-torsion on the PDP

As we saw in Chapter 5 universal Edwards model of elliptic curves, twisted Edwards and Jacobi intersection curves can also have rational 4-torsion points. The natural question follows, whether 4-torsion points are as useful as 2-torsion points for PDP resolution?

#### 6.3.1 Twisted Edwards curve

The two 4-torsion points of a twisted Edwards curve are  $T_4 = (\pm a^{-\frac{1}{2}}, 0)$ . Thus, if  $P = (x, y) \in E_{a,d}(\mathbb{F}_{q^n})$  then we have

$$P \oplus T_4 = (\pm a^{-\frac{1}{2}} \cdot y, \pm a^{\frac{1}{2}} \cdot x)$$

The sum of  $P$  with a 4-torsion point swaps – up to multiplication by  $\pm a^{\frac{1}{2}}$  or  $\pm a^{-\frac{1}{2}}$  – the coordinates of the point  $P$ . Hence, the action of  $T_4$  does not leave invariant the factor base. Moreover, in this representation the  $x$ -coordinate cannot be expressed in terms of the  $y$ -coordinate only so we cannot use this action to decrease the number of solutions of polynomial systems to solve.

#### 6.3.2 Universal Edwards model of elliptic curves

Assuming  $-1$  is a square in  $\mathbb{F}_{q^n}$  then the four 4-torsion points of a curve in universal Edwards model are  $T_{4,1} = (0, \sqrt{-1})$ ,  $T_{4,2} = (0, -\sqrt{-1})$ ,  $T_{4,3} = (\sqrt{-1}, 0)$  and  $T_{4,4} = (-\sqrt{-1}, 0)$ . Moreover, if  $P = (x, y)$  is a point of the curve then

- $P \oplus T_{4,1} = \left( \sqrt{-1} \frac{2tx-y}{xy-2t}, \sqrt{-1} \frac{1-2txy}{2tx-y} \right);$
- $P \oplus T_{4,2} = \left( \sqrt{-1} \frac{y-2tx}{xy-2t}, \sqrt{-1} \frac{2txy-1}{2tx-y} \right);$
- $P \oplus T_{4,3} = \left( \sqrt{-1} \frac{y-2tx}{2txy-1}, \sqrt{-1} \frac{2txy-1}{x-2ty} \right);$
- $P \oplus T_{4,4} = \left( \sqrt{-1} \frac{2tx-y}{2txy-1}, \sqrt{-1} \frac{1-2txy}{x-2ty} \right).$

Consequently, as for twisted Edwards curves the 4-torsion points do not leave the factor base invariant. Hence, they cannot be used to improve the PDP solving or to decrease the size of the factor base in a similar way that we use the 2-torsion.

#### 6.3.3 Twisted Jacobi intersection curve

In this section, we present a similar method, as for 2-torsion, to use the 4-torsion of twisted Jacobi intersection curves. Although we will see in Section 6.4 that this method does not allow to simplify the polynomial system solving step in the PDP solving, we present it for completeness and in order to report the experiments we did. Moreover, we will see that this approach is not useless, since it allows to further decrease the size of the factor base and consequently to speed up the complete solving of the ECDLP by index calculus attack.

We concentrate first on the case of the following 4-torsion point:

$$T_4 = \left( \pm \frac{1}{\sqrt{a}}, 0, \pm \sqrt{\frac{a-b}{a}} \right).$$

After a few simplifications, adding  $T_4$  to a generic point  $P = (x, y, z)$  of  $E_{a,b}(\mathbb{F}_{q^n})$  gives the formula

$$P \oplus T_4 = \left( \pm \frac{1}{\sqrt{a}} \cdot \frac{y}{z}, \pm \sqrt{a-b} \cdot \frac{x}{z}, \pm \sqrt{\frac{a-b}{a}} \cdot \frac{1}{z} \right).$$

As seen in Section 5.4.2, for twisted Jacobi intersection curves, it is possible to use either  $y$  or  $z$  for projecting the PDP and obtain interesting summation polynomials. To take advantage of the action of  $T_4$ , we project on  $z$  and work with the summation polynomial  $S_{m,z}$ .

One can notice that the  $z$ -coordinate of  $P \oplus T_4$  depends only on the  $z$ -coordinate of  $P$ . However, due to the factor  $\pm \sqrt{\frac{a-b}{a}}$  and also that for this representation the factor base cannot be  $\mathcal{F}_0$  the action of  $T_4$  does not leave the factor base invariant.

By consequence, in order to normalize a bit more the action of  $T_4$  and to use the action of the 4-torsion, we assume that  $\frac{a-b}{a}$  is a fourth power and do the change of coordinate

$$Z = \sqrt[4]{\frac{a}{a-b}} z,$$

so that adding  $T_4$  changes the  $Z$ -coordinate to  $\pm 1/Z$ . Moreover, in this case the factor base  $\mathcal{F} = \mathcal{F}_0$  seems to be large enough. Hence, the action of  $T_4$  leaves the factor base invariant and can be used to further decrease the size of the factor base  $\sim \frac{q}{16}$ . This change of coordinate preserves the property that adding  $T_2$  changes the sign of the  $Z$ -coordinate, so that we still have the action of  $D_m$  on  $S_{m,z}$ . This explicit action of  $T_4$  transforms a decomposition into another one, but unfortunately, this action is not linear and therefore does not fit easily in the framework that we have developed. As a consequence, we will not be able to reduce the degree of the ideal as much as we could hope for. Still, by adding a well-chosen variable to make the symmetry more visible, we constrain the LEX Gröbner basis to be in non shape position that had shown to be useful for  $T_2$ , before reducing the degree of the ideal.

We explain this strategy in the case of  $n = 4$ . Adding  $T_4$  to the 4 points of a decomposition gives another decomposition, where all the  $Z_i$  have been inverted. We defined a new coordinate  $v_4$  that is invariant by this involution:

$$v_4 = Z_1 Z_2 Z_3 Z_4 + \frac{1}{Z_1 Z_2 Z_3 Z_4} = e_4(Z_1, Z_2, Z_3, Z_4) + \frac{1}{e_4(Z_1, Z_2, Z_3, Z_4)}.$$

Therefore, we add the equation  $e_4 v_4 - e_4^2 - 1 = 0$  to the system obtained by applying a Weil restriction on  $g_4$  (the expression of  $S_{5,Z}^R$  in terms of  $s_1, s_2, s_3, e_4$ ). The corresponding LEX Gröbner basis has the following form:

$$\begin{cases} s_1 + \ell_1(e_4, v_4) \\ s_2 + \ell_2(e_4, v_4) \\ s_3 + \ell_3(e_4, v_4) \\ e_4 v_4 - e_4^2 - 1 \\ \ell_4(v_4) \end{cases}$$

where  $\deg(\ell_i) = 2^{n(n-2)}$  for all  $i = 1, \dots, 4$  and the degree of the ideal remains  $2^{(n-1)^2}$  as when using only  $T_2$ .

**Remark 6.7.** For  $n > 4$ , the variable  $v_4$  must be replaced by a variable that is invariant by any change of a multiple of four number of variables by their inverses.

We can note that adding two times  $T_4$  (i.e. adding a 2-torsion point) does not change the  $Z$ -coordinate. By consequence, we can change only an even number of variables by their inverse. Instead of  $v_4 = e_4 + \frac{1}{e_4}$  we could use  $v'_4 = \frac{s_2+1+e_4^2}{e_4}$  to further decrease the degree of the univariate polynomial in the lexicographical Gröbner basis.

The construction that we have just shown works *mutatis mutandis* with the other 4-torsion point of the form

$$T_4 = \left( \pm \frac{1}{\sqrt{b}}, \pm \sqrt{\frac{b-a}{b}}, 0 \right),$$

but in that case, we have to work with the  $y$ -coordinate instead of the  $z$ -coordinate.

From the parameters of the system, it is not clear that adding a variable to reduce the degree of the polynomials in the resulting Gröbner basis is worthwhile. Nevertheless, whether we add the variable  $v_4$  or not, the action of this 4-torsion point allows to further decrease the size of the factor base by a factor 2. Indeed, we mention in the beginning of Section 6.2 that for twisted Jacobi intersection curves we cannot use the factor base  $\mathcal{F}_0$  since it does not contain enough points. Hence, in this case the 4-torsion does not leave invariant the factor base and then cannot be used to decrease to size of the factor base. However, by changing the representation of the curve to normalize the action of the 4-torsion, the corresponding factor base  $\mathcal{F}_0$  seems to contain the expected number of points and then can be chosen for index calculus attack. Moreover, in this case the action of the 4-torsion leaves invariant the factor base and in consequence can be used to further decrease the size of the factor base by a factor 2.

## 6.4 Experimental results and security estimates

All experiments or comparisons in this section assume that the elliptic curve is a twisted Edwards or twisted Jacobi intersection curve or an universal Edwards model of elliptic curve. We recall that only curves with a particular torsion structure can be put into these forms and are subject to our improved attack.

The PDP problem for  $n = 2$  is not interesting, since it does not yield an attack that is faster than the generic ones. For  $n = 3$ , the PDP problem can be solved very quickly, so that our improvements using symmetries are difficult to measure. Therefore, we will concentrate on the  $n = 4$  and higher cases. Most of our experiments are done with MAGMA, which provides an easy-to-reproduce environment (the MAGMA codes to solve the PDP are available at <http://www-polsys.lip6.fr/~huot/CodesPDP>). For the largest computations, we used the FGb library which is more efficient for systems of the type encountered in the context of this chapter. The FGb library also provides a precise count of the number of basic operations (a multiplication of two 32-bit integers is taken as unit) that are required in a system resolution. We will use this information to interpolate security levels for large inputs.

### 6.4.1 Experiments with $n = 4$

In the case of  $n = 4$ , as mentioned in [JV13] the resolution is still fast enough so that the “ $n-1$ ” approach by Joux and Vitse does not pay. So we compare the three following approaches: the

classical index-calculus of [Gau09] based on Weierstrass representation (denoted W. [Gau09], in the following) and our approaches using the 2-torsion point (denoted  $T_2$ ) and using additionally the 4-torsion point (denoted  $T_{2,4}$ ). For  $T_2$  and  $T_{2,4}$ , we have implemented the two choices for the basis of the invariant ring for the dihedral Coxeter group given in Section 3.2.1, that we denote by  $s_i$  and  $p_i$ . As previously announced, we observe that  $\mathcal{S}_{\mathfrak{S}_n} \in \mathbb{K}[e_1, \dots, e_n]$  is a regular sequence. This is not the case for  $\mathcal{S}_{\mathfrak{S}_n} \in \mathbb{K}[p_1, \dots, p_n]$ . Hence, following results in Chapter 3 and Section 6.2.2, we equip the ring  $\mathbb{K}[s_1, \dots, s_{n-1}, e_n]$  with the weighted degree with weights  $(2, \dots, 2, 1)$ , while the ring  $\mathbb{K}[p_2, \dots, p_{2(n-1)}, p_n]$  is equipped with the usual degree. The results are given in Table 6.1, where one finds for various sizes of the base field the runtimes and the maximal (weighted) degree reached by polynomials during the computation of a (W)DRL Gröbner basis with  $F_4$ . In column  $d_{max}/d_{theo}$  one can find the maximal (weighted) degree reached by the polynomials and when the system is regular the bound on this maximal degree given by Corollary 2.76. The two last columns of Table 6.1 give the number of multiplications of two 32-bits words required to solve the corresponding polynomial system. The penultimate column gives an interpolated number of multiplications of two 32-bits words required by the MAGMA software. Since we observe that the most consuming step is the change of ordering we interpolate this number thanks to the complexity of the FGLM algorithm in  $O(nD^3)$  arithmetic operations. The last column gives the exact number of multiplications of two 32-bits words required by the FGb implementation. Since, FGb library uses the recent sparse change of ordering algorithm in [FM11, FM13, Mou13] (see Chapter 2) its practical arithmetic complexity is closer to be quadratic in the number of solutions than cubic.

| $\log_2(q)$ | weights    | $F_4$                      |                      | $d_{max}/d_{theo}$         |                      | Change Order               |                      | Total                      |                      | #ops     | # ops FGb                  |                      |
|-------------|------------|----------------------------|----------------------|----------------------------|----------------------|----------------------------|----------------------|----------------------------|----------------------|----------|----------------------------|----------------------|
|             |            | $s_i$<br>(2, ..., 2, 1, 1) | $p_i$<br>(1, ..., 1) | $s_i$<br>(2, ..., 2, 1, 1) | $p_i$<br>(1, ..., 1) | $s_i$<br>(2, ..., 2, 1, 1) | $p_i$<br>(1, ..., 1) | $s_i$<br>(2, ..., 2, 1, 1) | $p_i$<br>(1, ..., 1) | MAGMA    | $s_i$<br>(2, ..., 2, 1, 1) | $p_i$<br>(1, ..., 1) |
| 16          | W. [Gau09] | 5s                         |                      | 29/29                      |                      | 423s                       |                      | 428s                       |                      | $2^{36}$ | $2^{29}$                   |                      |
|             | $T_2$      | < 1s                       | < 1s                 | 26/27                      | 14                   | 1s                         | 3s                   | < 2s                       | < 4s                 | $2^{27}$ | $2^{24}$                   | $2^{26}$             |
|             | $T_{2,4}$  | < 1s                       | 1s                   | 21                         | 15                   | 2s                         | 3s                   | < 3s                       | 4s                   |          | $2^{24}$                   | $2^{27}$             |
| 64          | W. [Gau09] | 331s                       |                      | 29/29                      |                      | 5994s                      |                      | 6325s                      |                      | $2^{40}$ | $2^{33}$                   |                      |
|             | $T_2$      | 2s                         | 32s                  | 26/27                      | 14                   | 13s                        | 24s                  | 15s                        | 56s                  | $2^{31}$ | $2^{28}$                   | $2^{30}$             |
|             | $T_{2,4}$  | 8s                         | 61s                  | 21                         | 15                   | 12s                        | 25s                  | 20s                        | 86s                  |          | $2^{28}$                   | $2^{31}$             |
| 128         | W. [Gau09] | 480s                       |                      | 29/29                      |                      | 7179s                      |                      | 7559s                      |                      | $2^{42}$ | $2^{35}$                   |                      |
|             | $T_2$      | 2s                         | 40s                  | 26/27                      | 14                   | 14s                        | 32s                  | 16s                        | 72s                  | $2^{33}$ | $2^{30}$                   | $2^{32}$             |
|             | $T_{2,4}$  | 9s                         | 80s                  | 21                         | 15                   | 16s                        | 32s                  | 25s                        | 112s                 |          | $2^{30}$                   | $2^{33}$             |

Table 6.1: Computing time of Gröbner basis with MAGMA (V2-19.1) on one core of a 2.00 GHz Intel® E7540 CPU for  $n = 4$ . The last column (number of operations) is based on FGb.

We can observe that taking into account the symmetries dramatically decreases the computing time of the PDP resolution by a factor of about 400. This is consistent with the theoretical expected gain, as shown by the interpolated number of multiplications of two 32-bits words required by MAGMA which is divided by  $2^9 = 2^{3(n-1)}$ ; and also shown by the exact number of multiplications of two 32-bits words required by FGb which is divided by  $2^5$  of the order of  $2^{2(n-1)}$  corresponding to a quadratic complexity for the change of ordering and  $F_5$

algorithm (whose implementation in FGb also uses sparse linear algebra).

These experiments also show that the choice of the invariant ring basis  $s_i$  or  $p_i$  for the dihedral Coxeter group is not computationally equivalent. Indeed, the degrees of the polynomials depend on it: it is 8 for the  $s_i$  basis and 12 with the  $p_i$ . Moreover, one of the sequence is regular while the other is not. As a consequence, the DRL part of the computation is more costly for the  $p_i$  than for the  $s_i$ . One can notice that for the systems expressed in terms of the primary invariants of  $\mathfrak{S}_n$  ( $e_1, \dots, e_n$ ) and the systems expressed in terms of the primary invariants of  $D_n$  ( $s_1, \dots, s_{n-1}, e_n$ ) the maximal (weighted) degree reached by the polynomials during the computation of a degree monomial ordering Gröbner basis is tightly bounded by the bound of Corollary 2.76. We observe that the system  $\mathcal{S}_{\mathfrak{S}_n}$  (resp.  $\mathcal{S}_{D_n}$ ) is regular when we consider the usual degree (resp. the weighted degree with weights  $(2, \dots, 2, 1)$ ).

Moreover, we notice that the change of ordering step is the most time consuming step which is consistent with the complexity analysis of Theorem 6.6. This shows that it is important to have precise complexity bound for the change of ordering. Moreover, the complexity of change of ordering depends on the number of solutions of the system so this emphasizes the impact of the action of a pseudo reflective group.

One can notice that adding a variable to decrease the degree of polynomials in the computation of Gröbner basis (to use the 4-torsion) does not speed up the computation in this case. Indeed, adding the variable  $v_4$  breaks the quasi-homogeneous structure since we do not find an appropriate weight for this variable. Hence, in the following the 4-torsion point is used only to further decrease the size of the factor base. That is to say, we change the representation as presented in the previous section but we do not add the variable  $v_4$ . In this context the 4-torsion can be used for any  $n$ .

It can be observed that the two steps of the resolution are faster with the  $s_i$  basis. This is a general practical fact observed during our experiments. Thus, in the sequel, we consider only the  $s_i$  basis.

### 6.4.2 Experiments for $n = 5$ and $n = 6$

One of the main improvement brought by this work, is that we are now able to solve the polynomial systems coming from the summation polynomials for  $n = 5$  when the symmetries are used. Still, these computations are not feasible with MAGMA and we use the FGb library. Actually, the graded reverse lexicographical Gröbner basis can be computed with MAGMA but the change of ordering cannot. The timings are given in table 6.2.

| $\log_2(q)$ |                     | $F_5$            | $d_{max}/d_{theo}$ | Change-Order | Total | # ops    |
|-------------|---------------------|------------------|--------------------|--------------|-------|----------|
| 16          | W. [Gau09]<br>$T_2$ | > 2 days<br>567s | ??/76<br>72/73     | 2165s        | 2732s | $2^{44}$ |

Table 6.2: Computing time of Gröbner basis with FGb on a 3.47 GHz Intel® X5677 CPU for  $n = 5$ .

For  $n = 5$  Corollary 2.76 gives also a precise bound on the maximal degree reached by the polynomials. The regular hypothesis has been checked also on these systems.

Our improved algorithm using symmetries can be combined with the “ $n - 1$ ” approach of Joux and Vitse. This allows us to compare the running times with the approach taken in [JV13] in the case of  $n = 5$ , and to handle, for the first time, the case of  $n = 6$ . The results

are summarized in tables 6.3 and 6.4. For  $n = 6$ , MAGMA was not able to solve the system, so we used again FGb. Because of the low success probability, this technique is interesting only for medium  $q$ . Hence, we limit the size of  $q$  to 32 bits, and even to 16 bits for  $n = 6$ .

| $\log_2(q)$ |           | $F_4$         | # ops    |
|-------------|-----------|---------------|----------|
| 16          | W. [JV13] | 13.400s       | $2^{32}$ |
|             | $T_2$     | <b>0.090s</b> | $2^{22}$ |
|             | $T_{2,4}$ | 0.130s        | $2^{24}$ |
| 32          | W. [JV13] | 1278s         | $2^{34}$ |
|             | $T_2$     | <b>1.100s</b> | $2^{24}$ |
|             | $T_{2,4}$ | 1.760s        | $2^{26}$ |

Table 6.3: Computing time of Gröbner basis with MAGMA (V2-19.1) on one core of a 2.00 GHz Intel® E7540 CPU for  $n = 5$  and decomposition in  $n - 1$  points. Operation counts are obtained using FGb.

| $\log_2(q)$ |           | $F_5$    | # ops    |
|-------------|-----------|----------|----------|
|             |           | $s_i$    | $s_i$    |
| 16          | W. [JV13] | > 2 days |          |
|             | $T_2$     | 2448s    | $2^{39}$ |

Table 6.4: Computing time of DRL Gröbner basis with FGb on a 3.47 GHz Intel® X5677 CPU for  $n = 6$  and decomposition in  $n - 1$  points.

Using symmetries decreases the running time also for decompositions in  $n - 1$  points. For  $n = 5$ , the speed-up is by a factor about 150 for a 16-bit base field and by 1000 for a 32-bit base field. For  $n = 6$ , without using the symmetries of twisted Edwards or twisted Jacobi intersection curves or universal Edwards model of elliptic curves, we cannot compute decompositions in  $n - 1$  points while this work allows to compute them in approximately 40 minutes.

In Table 6.3, we can observe that considering the action of 4-torsion points of Jacobi intersection curves is more time consuming. Indeed, if the system admits a solution then it also admits all the solutions associated to the action of the 4-torsion points. By consequence, the overdetermined systems have not the same DRL and LEX Gröbner bases and their computation are slower. By consequence, for the “ $n - 1$ ” variant, the trade-off between the size of the factor base and the difficulty of decomposing a point is better when using only the 2-torsion.

Indeed, when we consider only the action of  $T_2$ , we use the factor base  $\mathcal{F} = \mathcal{F}_1$  ( $\mathcal{F}_0$  is too small). Hence, the action of  $T_4$  does not leave the factor base invariant. Moreover, the decompositions related to the action of the 4-torsion do not necessarily correspond to solutions of the system obtained after the Weil restriction on summation polynomials. In fact, we observe that the corresponding system has the expected number of solutions that is 0 or 1.

**Remark 6.8.** For  $n \geq 6$ , the first difficulty to solve the PDP is the construction of the summation polynomials. Actually, the seventh summation polynomial or the seventh summation polynomial evaluated in the  $c$ -coordinate of a point  $R$  have never been computed. We will show

in Chapter 7 that for fields of characteristic two we can now compute summation polynomials until  $n = 8$ .

### 6.4.3 Security level estimates

To conclude these experimental results, we use our operation counts for the PDP to estimate the cost of a complete resolution of the ECDLP for twisted Edwards or twisted Jacobi intersection curves. In this section, we count only arithmetic operations and we neglect communications and memory occupation. Hence, this does not give an approximation of the computation time but this gives a first approximation of the cost to solve some instances of the ECDLP.

We compare the result with all previously known attacks, including the generic algorithms, whose complexity is about  $q^{\frac{n}{2}}$  operations in  $E(\mathbb{F}_{q^n})$ . The cost of an elliptic curve operation can be approximated by  $\log_2(q^n)^2$ . Since our cost unit for boolean operations is a 32-bit integer multiplication, we roughly approximate the cost of an elliptic curve operation by  $n^2 \log_{2^{32}}(q)^2$  and the total boolean cost of a generic attack by

$$n^2 q^{\frac{n}{2}} \log_{2^{32}}(q)^2.$$

According to Remark 6.4 and the end of Section 6.3.3, for index calculus using the point decomposition in  $n$  points we look for  $N$  relations where  $N$  is:

- $\frac{q}{2}$  for Weierstrass representation,
- $\frac{q}{4}$  for twisted Edwards curves,
- $\frac{q}{8}$  for universal Edwards model of elliptic curves and twisted Jacobi intersection curves when using only the 2-torsion,
- $\frac{q}{16}$  for twisted Jacobi intersection curves and by using the 2-torsion and the 4-torsion.

The probability to decompose a point is  $\frac{1}{n!}$ . Let  $c(n, q, m)$  be the number of boolean operations needed to solve one polynomial system obtained from a Weil restriction of the  $(m+1)^{\text{th}}$  summation polynomial defined over  $\mathbb{F}_{q^n}$ , evaluated in one variable. This number of operations is obtained by experiments with FGb as demonstrated in the previous subsections. From the function  $c(n, q, m)$  from Section 5.5.2 one can deduce the total number of operations needed to solve the ECDLP over  $\mathbb{F}_{q^n}$ :

$$N \cdot n! \cdot c(n, q, n) + n^3 \log_{2^{32}}(q)^2 N^2.$$

If we use the point decomposition in  $n-1$  points, due to exhaustive search, the probability to find a decomposition is now  $\frac{1}{q \cdot (n-1)!}$ . Hence, according to Section 5.5.4 the total number of operations is, in this case, given by

$$q(n-1)! \cdot N \cdot c(n, q, n-1) + n^2(n-1) \log_{2^{32}}(q)^2 \cdot N^2.$$

When the linear algebra step is more time consuming than the relation search, by using the double large prime variation [GTTD07] we can rebalance the costs of these two steps (see [Th  03, GTTD07]). The total number of operations needed to solve the ECDLP over  $\mathbb{F}_{q^n}$  by using the double large prime variation is given by (see Section 5.5.3):

$$\log_2(q) \left( 1 + r \frac{n-1}{n} \right) (n-2)! q^{1+(n-2)(1-r)} c(n, q, n) + n^3 \log_{2^{32}}(q)^2 N^{2r}$$

where we look for  $r$  such that the two parts of this complexity are equal.

The results are summarized in Table 6.5. The notations  $T_2$  and  $T_{2,4}$  still denote the use of the 2-torsion points of twisted Edwards, twisted Jacobi intersection curves and universal Edwards model of elliptic curves and the use of the 2-torsion and 4-torsion points of twisted Jacobi intersection curves respectively. Twisted Jacobi intersection representation is denoted Jac or Jacobi for short, twisted Edwards representation is denoted Edwards for short and universal Edwards model of elliptic curves is denoted Uni-Edw for short.

We observe that the smallest number of operations obtained for each parameter is given by index calculus using symmetries induced by the 2-torsion points (and 4-torsion point when decomposing in  $n$  points is possible) or generic algorithms. We note that for  $n \leq 5$  our version of the index calculus attack is better than generic algorithms. For example, if  $\log_2(q) = 64$  and  $n = 4$  generic algorithms need  $2^{134}$  operations to attack the ECDLP and we obtain  $2^{116}$  by using the 2-torsion points and 4-torsion point. In this case, our approach is more efficient than the basic index calculus, solving this instance of ECDLP in  $2^{121}$  operations. For  $n = 5$ , the resolution of the PDP was intractable but with our method, we can now solve these instances of the PDP and we attack the corresponding instances of the ECDLP with a gain of  $2^{39}$  over generic algorithms and a gain of  $2^{40}$  over Joux and Vitse approach.

We remark that for parameters for which it is possible to choose between the decomposition in  $n$  or  $n - 1$  points, the best solution is the first. For  $n = 6$  we are not able to decompose a point in  $n$  points of the factor base. Consequently it is necessary to use the decomposition in  $n - 1$  points. For  $n = 6$  generic algorithms have a complexity in  $O(q^3)$ , while the index calculus attack using the decomposition in  $n - 1$  points has a complexity in  $O(C(n) \cdot q^2)$  where  $C(n)$  is exponential in  $n$ . Hence to be better than generic algorithms, we have to consider high values of  $q$  and consequently high security levels. For instance if  $\log_2(q) = 64$ , the index calculus attack using symmetries of twisted Edwards or twisted Jacobi intersection curves or universal Edwards model of elliptic curves and decomposition in  $n - 1$  points needs less operations ( $2^{176}$ ) than the generic algorithms, ( $2^{200}$ ). In our point of view the only hope to have a better gain in general (for lower security level) compared to generic algorithms, would be to remove the bad dependence in  $q$  in the complexity that seems intrinsic to the “ $n - 1$ ” approach.

In cryptology, one looks for parameters giving some user-prescribed security level. Thereafter we give the domain parameters for different security levels expressed in number of boolean operations.

In Table 6.6, we compare for a fixed security level the size of  $q$  that we have to choose for  $n = 4, 5, 6$  by considering the attack based on generic algorithms with the attack based on the best version of index calculus. For the index calculus attack, except for  $n = 6$ , the size of  $q$  is obtained by considering decomposition in  $n$  points using the symmetries (2-torsion and 4-torsion) of twisted Jacobi intersection curves. This table confirms the previous observations. For  $n = 4, 5$ , the size of  $q$  is increased because of the new version of index calculus proposed in this work. For  $n = 6$  this is true only for very high security level.

| Curve parameters |             | Curve representation and torsion used                                 | Generic algorithm | Linear algebra                                   | Relations search decomposition in                        |                                                         | Double large prime variation                     | Total DLP                                                                                 |
|------------------|-------------|-----------------------------------------------------------------------|-------------------|--------------------------------------------------|----------------------------------------------------------|---------------------------------------------------------|--------------------------------------------------|-------------------------------------------------------------------------------------------|
| $n$              | $\log_2(q)$ |                                                                       |                   |                                                  | $n$ points                                               | $n - 1$ points                                          |                                                  |                                                                                           |
| 4                | 32          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{68}$          | $2^{68}$<br>$2^{66}$<br>$2^{64}$<br>$2^{62}$     | $2^{67}$ [Gau09]<br>$2^{61}$<br>$2^{60}$<br>$2^{59}$     |                                                         | $2^{66}$<br>$2^{64}$                             | $2^{68}$<br>$2^{66}$<br>$2^{64}$<br><b><math>2^{62}</math></b>                            |
|                  | 64          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{134}$         | $2^{134}$<br>$2^{132}$<br>$2^{130}$<br>$2^{128}$ | $2^{101}$ [Gau09]<br>$2^{95}$<br>$2^{94}$<br>$2^{93}$    |                                                         | $2^{121}$<br>$2^{118}$<br>$2^{117}$<br>$2^{116}$ | $2^{121}$<br>$2^{118}$<br>$2^{117}$<br><b><math>2^{116}</math></b>                        |
|                  | 128         | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{264}$         | $2^{264}$<br>$2^{262}$<br>$2^{260}$<br>$2^{258}$ | $2^{167}$ [Gau09]<br>$2^{161}$<br>$2^{160}$<br>$2^{159}$ |                                                         | $2^{220}$<br>$2^{216}$<br>$2^{215}$<br>$2^{215}$ | $2^{220}$<br>$2^{216}$<br><b><math>2^{215}</math></b><br><b><math>2^{215}</math></b>      |
| 5                | 32          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{85}$          | $2^{69}$<br>$2^{67}$<br>$2^{65}$<br>$2^{63}$     | $\infty$<br>$2^{83}$<br>$2^{82}$<br>$2^{81}$             | $2^{102}$ [JV13]<br>$2^{91}$<br>$2^{90}$<br>$2^{92}$    |                                                  | $2^{85}$<br>$2^{83}$<br>$2^{82}$<br><b><math>2^{81}</math></b>                            |
|                  | 64          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{167}$         | $2^{135}$<br>$2^{133}$<br>$2^{131}$<br>$2^{129}$ | $\infty$<br>$2^{117}$<br>$2^{116}$<br>$2^{115}$          | $2^{168}$ [JV13]<br>$2^{157}$<br>$2^{156}$<br>$2^{158}$ | $2^{130}$<br>$2^{129}$<br>$2^{128}$              | $2^{167}$<br>$2^{130}$<br>$2^{129}$<br><b><math>2^{128}</math></b>                        |
|                  | 128         | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw<br>$T_{2,4}$ Jacobi | $2^{329}$         | $2^{265}$<br>$2^{263}$<br>$2^{261}$<br>$2^{259}$ | $\infty$<br>$2^{183}$<br>$2^{182}$<br>$2^{181}$          | $2^{298}$ [JV13]<br>$2^{287}$<br>$2^{286}$<br>$2^{288}$ | $2^{235}$<br>$2^{234}$<br>$2^{233}$              | $2^{298}$<br>$2^{235}$<br>$2^{234}$<br><b><math>2^{233}</math></b>                        |
| 6                | 32          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw                     | $2^{102}$         | $2^{70}$<br>$2^{68}$<br>$2^{66}$                 | $\infty$<br>$\infty$<br>$\infty$                         | $\infty$<br>$2^{110}$<br>$2^{109}$                      |                                                  | <b><math>2^{102}</math></b><br><b><math>2^{102}</math></b><br><b><math>2^{102}</math></b> |
|                  | 64          | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw                     | $2^{200}$         | $2^{136}$<br>$2^{134}$<br>$2^{132}$              | $\infty$<br>$\infty$<br>$\infty$                         | $\infty$<br>$2^{176}$<br>$2^{175}$                      |                                                  | $2^{200}$<br>$2^{176}$<br><b><math>2^{175}</math></b>                                     |
|                  | 128         | Weierstrass<br>$T_2$ Edwards<br>$T_2$ Jac/Uni-Edw                     | $2^{394}$         | $2^{266}$<br>$2^{264}$<br>$2^{262}$              | $\infty$<br>$\infty$<br>$\infty$                         | $\infty$<br>$2^{306}$<br>$2^{305}$                      |                                                  | $2^{394}$<br>$2^{306}$<br><b><math>2^{305}</math></b>                                     |

Table 6.5: Number of operations needed to solve the ECDLP defined over  $\mathbb{F}_{q^n}$  for  $n = 4, 5, 6$  and  $32 \leq \log_2(q) \leq 128$ .

| Security level                      |             | 2 <sup>80</sup>  |           |           | 2 <sup>112</sup> |            |           |
|-------------------------------------|-------------|------------------|-----------|-----------|------------------|------------|-----------|
| $n$                                 |             | 4                | 5         | 6         | 4                | 5          | 6         |
| Generic Algorithm<br>Index Calculus | $\log_2(q)$ | 38               | 31        | <b>26</b> | 54               | 43         | <b>36</b> |
|                                     |             | <b>42</b>        | <b>32</b> | 19        | <b>62</b>        | <b>56</b>  | 34        |
| Security level                      |             | 2 <sup>128</sup> |           |           | 2 <sup>192</sup> |            |           |
| $n$                                 |             | 4                | 5         | 6         | 4                | 5          | 6         |
| Generic Algorithm<br>Index Calculus | $\log_2(q)$ | 62               | 49        | 41        | 93               | 74         | 62        |
|                                     |             | <b>72</b>        | <b>64</b> | <b>42</b> | <b>113</b>       | <b>103</b> | <b>73</b> |

Table 6.6: Domain parameters according to the security level given in number of boolean operations needed to solve the ECDLP.

## CHAPTER 7

# Summation polynomials in characteristic 2

---

### Contents

---

|                                                                                                       |            |
|-------------------------------------------------------------------------------------------------------|------------|
| <b>7.1 Compact representation of summation polynomials in characteristic two</b>                      | <b>153</b> |
| 7.1.1 Symmetries                                                                                      | 153        |
| 7.1.2 Density                                                                                         | 158        |
| <b>7.2 Compact summation polynomials by resultant and Gröbner bases</b>                               | <b>159</b> |
| <b>7.3 Outline of sparse multivariate polynomial interpolation algorithm</b>                          | <b>161</b> |
| 7.3.1 Description of Zippel's sparse multivariate polynomial interpolation algorithm                  | 161        |
| 7.3.2 Complexity and probability of success of Zippel's algorithm                                     | 164        |
| <b>7.4 Summation polynomials by implicit sparse multivariate interpolation</b>                        | <b>165</b> |
| 7.4.1 Evaluation of summation polynomials using factorization and resultant of univariate polynomials | 166        |
| 7.4.2 Sparing factorizations                                                                          | 169        |
| 7.4.3 Degree of summation polynomials                                                                 | 171        |
| 7.4.4 Computation of the eighth summation polynomial                                                  | 172        |
| 7.4.5 Discussion about the computation of the ninth summation polynomial                              | 173        |
| <b>7.5 Application to the Discrete Logarithm Problem</b>                                              | <b>175</b> |
| 7.5.1 Using symmetries to speed up the PDP solving in characteristic two                              | 175        |
| 7.5.2 Benchmarks on the PDP solving                                                                   | 176        |

---

*The results presented in this chapter are from a joint work in progress with J.-C. Faugère, A. Joux, G. Renault and V. Vitse.*

In this chapter we investigate the computation of summation polynomials for binary curves. The drawback of the method presented in Chapter 5 Section 5.6.2 to compute them is that it involves polynomials that are much bigger than the output. Using this method, for binary curves we cannot compute summation polynomials for  $n > 6$ .

In order to overcome this issue, we use interpolation method. Note that using such a method has been suggested from a complexity point of view by Diem in [Die11b]. In order to take full advantage of interpolation methods we highlight a compact representation of summation polynomials.

Such a compact representation is obtained by studying the symmetries of binary curves. Indeed, as the summation polynomials inherit the symmetries of binary curves we can use

those to highlight a polynomial change of variables. Applying this change of variables then decreases the degree of these polynomials and their number of monomials.

To take advantage of this compact representation we want to compute the summation polynomials by interpolation directly expressed in terms of this change of variables. We recall that the summation polynomial of index  $n$  is defined as the resultant of two summation polynomials of smaller index. However, the compact representation of these two polynomials cannot be expressed in terms of the same set of variables. Recovering their corresponding evaluation points then requires to invert the change of variables *i.e.* to solve a polynomial system. We show that this system has a particular structure and can be solved very efficiently by factorizing univariate polynomials.

All in all, this enables the computation of the summation polynomials up to  $n = 8$ .

**Compact representation of summation polynomials.** In Section 5.2.4, we have seen that universal Edwards model of binary elliptic curve (*i.e.* defined over any field of characteristic two) has a rational two-torsion point. Hence, we would like to use it to highlight some symmetries on the corresponding summation polynomials. Let  $T_2$  be the two-torsion point of a binary curve in universal Edwards model. If  $(x, y)$  is any point of the curve then  $(x, y) \oplus T_2 = \left(\frac{1}{x}, \frac{1}{y}\right)$ . Contrary to the action of the two-torsion point in high characteristic, the action of the two-torsion point of binary curves is no longer linear. By consequence, a first difficulty in using the action of the two-torsion point in characteristic two is then to find a convenient representation of the curve making this action “*simple*” enough.

Note that the action of a two-torsion point on summation polynomials is necessarily of order two. The only linear action whose the order divides two in characteristic two is the identity. Hence, the action of the two-torsion point of a binary curve on summation polynomials cannot be linear. As a consequence, the most simple action that we can hope is the affine action given by addition with the constant 1.

We show that by applying a well-chosen change of coordinates on the binary curve in universal Edwards model, this affine action is exactly the action of the two-torsion point on summation polynomial. In that case, the two-torsion point implies particular symmetries, providing a compact representation of the summation polynomials. By introducing a new theory about summation polynomials, we are able to generalize the result of Proposition 6.5 to affine action of the dihedral Coxeter group in characteristic two. More precisely, we obtain the following result.

**Proposition 7.1.** *Let  $E$  be a binary elliptic curve defined over  $\mathbb{K}$ . Assume  $E$  has a two-torsion point  $T_2$  such that the  $x$ -coordinate of  $(x, y) \oplus T_2$  is  $x + 1$  and  $P$  and  $\ominus P$  share the same abscissa. The  $n$ th summation polynomial  $S_n \in \mathbb{K}[x_1, \dots, x_n]$  of  $E$  is invariant under the dihedral Coxeter group  $D_n = (\mathbb{Z}/2\mathbb{Z})^{n-1} \rtimes \mathfrak{S}_n$  where  $(\mathbb{Z}/2\mathbb{Z})^{n-1}$  acts by adding 1 to an even number of variables. For  $n \geq 3$ ,  $S_n$  can thus be expressed in terms of  $e_1(\mathbf{x}), e_2(\mathbf{X}), \dots, e_n(\mathbf{X})$  where  $e_i$  is the  $i$ th elementary symmetric polynomial,  $\mathbf{x} = (x_1, \dots, x_n)$  and  $\mathbf{X} = (x_1^2 + x_1, \dots, x_n^2 + x_n)$ .*

In Section 5.2.4 we have seen that the universal Edwards model of elliptic curves has good reduction modulo two. That is to say, the equation defining the curve and the group law of the curve are independent from the characteristic of the field the curve is defined over. Thus, as for Weierstrass curves [Die11b], summation polynomials of universal Edwards model of binary elliptic curves can be constructed as the reduction modulo two of summation polynomials of universal Edwards model of rational elliptic curves. As a consequence, binary summation

polynomials inherit the symmetries of summation polynomials of curves defined over the rational numbers. We show that for a well chosen change of coordinates the symmetries inherited from the action of the two-torsion in characteristic zero are consistent with the symmetries induced by the action of the two-torsion of the binary curve. More precisely, we obtain the following result.

**Proposition 7.2.** *Let  $E$  be an elliptic curve in universal Edwards model defined over a binary field  $\mathbb{K} = \mathbb{F}_{2^k}$ . Let  $\phi_\gamma : E \rightarrow E_\gamma$  be the change of coordinates defined by  $(x, y) \mapsto \left(\frac{1}{x+1} + \gamma, y\right)$  with  $\gamma \in \mathbb{K}$ . If  $\gamma \in \{0, 1\}$  then for any  $n \geq 3$  the  $n$ th summation polynomials of  $E_\gamma$  can be written in terms of*

$$\begin{cases} e_1(x_1^2, \dots, x_n^2) \\ e_j(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) & \text{for } j = 2, \dots, n-1 \\ e_n(x_1^2 + x_1, \dots, x_n^2 + x_n) \end{cases}$$

where  $e_i$  is the  $i$ th elementary symmetric polynomial.

A direct consequence of this compact representation is that we are now able to compute until the 7th summation polynomial by using the usual method involving multivariate resultant and Gröbner bases computations.

In Chapter 5 Remark 5.11 we already note that summation polynomials in characteristic two are particularly sparse. The change of coordinates of Proposition 7.2 allows to further decrease their density. For instance, the sixth summation polynomial expressed in terms of the elementary symmetric polynomials as suggested in [Gau09] contains 638 terms while using this compact representation it contains only 50 terms. By using this compact representation the seventh summation polynomial has 2247 terms for a total degree of 16 and the degrees in each variable are 16, 5, 8, 4, 8, 5, 16. In comparison with a dense polynomial of same degree containing 169,581 terms, the density of the seventh summation polynomial is then about 1.32%.

**Summation polynomials by sparse interpolation.** Let  $S'_n$  be the  $n$ th summation polynomial expressed in terms of the change of coordinates given in Proposition 7.2. To take advantage of this compact representation, we compute  $S'_n$  by sparse interpolation method. We thus need an efficient way to evaluate it. The difficulty in evaluating this polynomial is that we do not have a parametrization of the evaluation but only an implicit evaluation given by polynomial equations. Let us make it clearer.

The  $n$ th summation polynomial is constructed recursively as follows:

$$S_n(x_1, \dots, x_n) = \text{Res}_X (S_{n-k+1}(x_1, \dots, x_{n-k}, X), S_{k+1}(x_{n-k+1}, \dots, x_n, X))$$

for any  $k \in \{2, \dots, n-2\}$ . To evaluate  $S'_n$  at the evaluation point  $(\tilde{y}_1, \dots, \tilde{y}_n) \in \mathbb{K}^n$  we thus need to invert the change of coordinates *i.e.* to solve the following polynomial system:

$$\mathcal{S} = \begin{Bmatrix} e_1(x_1^2, \dots, x_n^2) - \tilde{y}_1 \\ e_2(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) - \tilde{y}_2 \\ \vdots \\ e_{n-1}(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) - \tilde{y}_{n-1} \\ e_n(x_1^2 + x_1, \dots, x_n^2 + x_n) - \tilde{y}_n \end{Bmatrix}.$$

According to the Bézout bound, the system  $\mathcal{S}$  has more than one solution but the evaluation of  $S'_n$  at  $(\tilde{y}_1, \dots, \tilde{y}_n)$  is unique. Actually, we can choose any solution of  $\mathcal{S}$ . The change of coordinates of Proposition 7.2 corresponds to the action of a group  $\mathbb{G}$  on the summation polynomials. Inverting this change of coordinates is then equivalent to find all the elements in the orbit represented by  $(\tilde{y}_1, \dots, \tilde{y}_n)$ . Since the polynomial  $S_n$  is invariant under the action of  $\mathbb{G}$ , its evaluations at all the elements in this orbit are equal.

One can notice that all the equations in  $\mathcal{S}$  are given by composition of elementary symmetric polynomials and univariate polynomials. Assume we want to solve the system

$$\mathcal{S}' = \{e_1(f(x_1), \dots, f(x_n)) + \alpha_1, \dots, e_n(f(x_1), \dots, f(x_n)) + \alpha_n\} \subset \mathbb{F}_{2^k}[x_1, \dots, x_n]$$

where  $\alpha_1, \dots, \alpha_n$  are in  $\mathbb{F}_{2^k}$ . By noting that

$$\prod_{i=1}^n (X + f(x_i)) = X^n + \sum_{i=1}^n e_i(f(x_1), \dots, f(x_n)) X^{n-i}$$

solving  $\mathcal{S}'$  is reduced to solve univariate polynomials. We show that  $\mathcal{S}$  can be solved similarly.

Finally, to evaluate  $S'_n$  we have to evaluate the two polynomials  $S_{n-k+1}$  and  $S_{k+1}$  at the evaluation point given by the chosen solution of  $\mathcal{S}$ . By taking care that the leading terms in  $X$  of these two polynomials are not cancelled then the evaluation of  $S'_n$  is obtained by computing a univariate resultant.

Thus, we avoid multivariate resultants and Gröbner bases computations and obtain an efficient way to evaluate summation polynomials in the compact representation.

Since, we do not know in advance a sharp bound on the number of terms of the  $n$ th summation polynomial we prefer to use Zippel's sparse multivariate interpolation [Zip79, Zip90]. Its complexity and the size of the involved polynomials depend only on the size of the output *i.e.* the real number of terms in the expected polynomial. This way, we are now able to compute the eighth summation polynomial. For a given curve defined over a binary field of 32 bits size, the eighth summation polynomial contains 470,369 terms and can be computed in approximately six hours using eight CPU cores.

**Impact on the PDP solving.** The symmetries on binary summation polynomials imply a theoretical and practical speed up of the *Point Decomposition Problem* solving. The polynomials defining the change of coordinates in Proposition 7.2 are symmetric and can thus be expressed as polynomials in the elementary symmetric polynomials. That is to say, there exist  $n$  multivariate polynomials  $\rho_1, \dots, \rho_n$  such that

$$\begin{cases} \rho_1(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) = e_1(x_1^2, \dots, x_n^2) \\ \rho_i(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) = e_i(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) \quad \text{for } i = 2, \dots, n-1 \\ \rho_n(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) = e_n(x_1^2 + x_1, \dots, x_n^2 + x_n) \end{cases}$$

where  $\mathbf{x} = (x_1, \dots, x_n)$ . In particular,  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are algebraically independent. Therefore, we can use results of Chapter 3 to estimate the complexity of solving the *Point Decomposition Problem* on binary curves and also to estimate the gain in comparison with the previous approach of Gaudry [Gau09] presented in Chapter 5. Since  $\prod_{i=1}^n \deg(\rho_i) = 2^{2(n-1)}$ , the complexity of solving the PDP for binary curves defined over a field  $\mathbb{F}_{q^n}$  with  $q = 2^k$  is divided by an exponential factor  $2^{2\omega(n-1)}$  in comparison to the original algorithm of Gaudry. More precisely, we obtain the following result.

**Theorem 7.3.** *Let  $E$  be a binary elliptic curve defined over  $\mathbb{K} = \mathbb{F}_{2^{nk}}$  by*

$$E : y^2 + xy = x^3 + \alpha \quad (7.1)$$

*where  $\alpha \in \mathbb{K}$ . Under Hypothesis 5.18, the arithmetic complexity of solving the Point Decomposition Problem is bounded by*

- (proven complexity)  $\tilde{O}(n \cdot 2^{3(n-1)(n-2)})$ ;
- (heuristic complexity)  $\tilde{O}(ne^{\omega n} 2^{\omega(n-1)(n-2)})$ ;

*where the notation  $\tilde{O}$  means that we omit polynomial factors in  $k$ .*

The proven complexity of Proposition 7.3 is obtained by using the complexity of FGLM algorithm. The heuristic complexity is obtained by observing that the ideal generated by the system coming from the summation polynomials expressed w.r.t. the change of coordinates of Proposition 7.2 is in *Shape Position*. Thus, we can heuristically use fast change of ordering, presented in Chapter 4, with better complexity.

An overall consequence of this work is one can now solve the PDP until  $n = 5$  by using the computer algebra system MAGMA [BCP97]. For instance, if  $k = 16$  the PDP is solved in less than six minutes while this instance of the PDP was intractable before.

## 7.1 Compact representation of summation polynomials in characteristic two

In this section, we first investigate the symmetries of summation polynomials. In particular we obtain Proposition 7.2. Then, we study the impact of such a representation on their density.

Throughout this chapter, summation polynomials are defined as the projection on the  $x$ -coordinate of the modeling of the PDP as a multivariate polynomial system.

### 7.1.1 Symmetries

To begin with, we study how the symmetries of rational curves are handed down to binary summation polynomials.

#### Inherited symmetries from rational curves

Summation polynomials are defined up to multiplication by a non zero constant. Hence, we define their canonical form as follows.

**Definition 7.4** (Canonical form of summation polynomials). *Let  $t$  be the parameter of an elliptic curve in universal Edwards model defined over  $\mathbb{K}$ . The canonical form  $f$  of its  $n$ th summation polynomial satisfies  $f \in R[x_1, \dots, x_n]$  and there not exist  $1 \neq c \in R \setminus \{0\}$  such that  $f = cg$  with  $g \in R[x_1, \dots, x_n]$  where  $R = \mathbb{Z}[t]$  if  $\mathbb{K} = \mathbb{Q}$  and  $R = \mathbb{F}_q[t]$  if  $\mathbb{K} = \mathbb{F}_q$ .*

Since, the  $n$ th summation polynomial of universal Edward model of elliptic curve defined over  $\mathbb{K}$  is in  $\mathbb{K}(t)[x_1, \dots, x_n]$  the existence of its canonical form is straightforward. Note that the uniqueness of this canonical form is also straightforward from its definition.

In Chapter 6 we have shown that in characteristic zero the action of the 2-torsion point  $(-\frac{1}{2t}, -1)$  of elliptic curves in universal Edwards model implies particular symmetries on their summation polynomials.

**Notation 7.5.** Let  $S_n \in \mathbb{Z}[t][x_1, \dots, x_n]$  be the canonical form of the  $n$ th summation polynomial of a curve in universal Edwards model defined over  $\mathbb{Q}$ .

More precisely, for any  $n \geq 3$  we have shown that  $S_n$  is invariant under the dihedral Coxeter group  $D_n$ , and thus can be expressed in terms of the primary invariants of  $D_n$ , for instance  $s_1(\mathbf{x}), \dots, s_{n-1}(\mathbf{x}), e_n(\mathbf{x})$  with  $\mathbf{x} = (x_1, \dots, x_n)$ , see Section 3.2.1.

**Notation 7.6.** Let  $SD_n$  be the expression of  $S_n$  in terms of  $s_1(\mathbf{x}), \dots, s_{n-1}(\mathbf{x}), e_n(\mathbf{x})$  i.e.  $SD_n(s_1(\mathbf{x}), \dots, s_{n-1}(\mathbf{x}), e_n(\mathbf{x})) = S_n(\mathbf{x})$ .

The equation defining a curve in universal Edwards model and its group law are independent from the characteristic of the field the curve is defined over. Moreover, the summation polynomials are constructed from this equation and the group law. Hence, to compute the  $n$ th summation polynomial in  $\mathbb{F}_p(t)[x_1, \dots, x_n]$  of a curve defined over  $\mathbb{F}_{p^k}$  one can perform the reduction modulo  $p$  throughout the computation or only at the end. That is to say, the summation polynomials of binary universal Edwards model can be computed as the reduction modulo 2 of the canonical form of the summation polynomials of universal Edwards model of elliptic curves defined over  $\mathbb{Q}$ .

**Notation 7.7.** Let  $Sb_n \in \mathbb{F}_2[t][x_1, \dots, x_n]$  be the canonical form of the  $n$ th summation polynomial of a curve in universal Edwards model defined over  $\mathbb{F}_{2^k}$ .

Note that, for any prime  $p$  the definition of the canonical form of summation polynomials of universal Edwards model of elliptic curve defined over  $\mathbb{Q}$  implies that  $S_n \bmod p \neq 0$ .

**Proposition 7.8.** For any  $n \geq 3$ , there exists a unique polynomial  $SbD_n \in \mathbb{F}_2[t][y_1, \dots, y_n]$  such that

$$Sb_n(\mathbf{x}) = SbD_n(s_1(\mathbf{x}), \dots, s_{n-1}(\mathbf{x}), e_n(\mathbf{x}))$$

where  $\mathbf{x} = (x_1, \dots, x_n)$  and  $s_1, \dots, s_{n-1} \in \mathbb{F}_2[x_1, \dots, x_n]$  are the elementary symmetric polynomial in terms of  $x_1^2, \dots, x_n^2$  and  $e_n$  is the  $n$ th elementary symmetric polynomial.

*Proof.* From Proposition 6.5,  $S_n$  is invariant under the action of  $D_n$ . Since  $\mathbb{Q}$  is of characteristic zero, there exists a unique polynomial  $SD_n \in \mathbb{Z}[t][y_1, \dots, y_n]$  such that  $SD_n$  is the expression of  $S_n$  in terms of  $s_1, \dots, s_{n-1}, e_n$ . One can note that there is no  $c \in \mathbb{Z}[t] \setminus \{0\}$  such that  $c \neq 1$  and  $SD_n = cg$  with  $g \in \mathbb{Z}[t][y_1, \dots, y_n]$ . Otherwise this contradicts the fact that  $S_n$  is the canonical form of the  $n$ th summation polynomial. By consequence, let  $F$  be the reduction of  $SD_n$  modulo 2, we then have  $F \neq 0$  is the unique (up to multiplication by a non zero constant) polynomial in  $\mathbb{F}_2[t][y_1, \dots, y_n]$  such that  $F$  is the expression of  $Sb_n$  in terms of  $s_1, \dots, s_{n-1}, e_n$ . If there is no  $c \in \mathbb{F}_2[t] \setminus \{0\}$  such that  $c \neq 1$  and  $F = cg$  with  $g \in \mathbb{F}_2[t][y_1, \dots, y_n]$  then  $F = SbD_n$  otherwise in order to get  $SbD_n$  we have to normalize  $F$ . The following diagram summarizes this construction of summation polynomials in characteristic two where  $c \in \mathbb{F}_2[t] \setminus \{0\}$ .

$$\begin{array}{ccc} S_n & \xrightarrow{\text{mod } 2} & c \cdot Sb_n \\ \downarrow D_n & & \uparrow \begin{array}{l} s_1 = x_1^2 + \dots + x_n^2 \\ \vdots \\ e_n = x_1 x_2 \cdots x_n \end{array} \\ SD_n & \xrightarrow{\text{mod } 2} & c \cdot SbD_n \end{array}$$

□

**Example 7.9.** For  $n = 3$  we have

$$Sb_3 = t(x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2) + x_1x_2x_3 + t$$

and

$$S_3 = 4t^3x_1^2x_2^2x_3^2 - t(x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2) + (1 - 16t^4)x_1x_2x_3 + 4t^3(x_1^2 + x_2^2 + x_3^2) - t$$

which implies that

$$SD_3 = 4t^3y_3^2 - ty_2 + (1 - 16t^4)y_3 - 4t^3y_1 + t.$$

Thus,

$$F = SD_3 \pmod{2} = ty_2 + y_3 + t.$$

Since  $F$  is normalized then  $SbD_3 = F$  and we have

$$SbD_3(s_1, s_2, e_3) = t(x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2) + x_1x_2x_3 + t = Sb_3$$

with  $s_1 = x_1^2 + x_2^2 + x_3^2$ ,  $s_2 = x_1^2x_2^2 + x_1^2x_3^2 + x_2^2x_3^2$  and  $e_3 = x_1x_2x_3$ .

Obviously, it is more efficient to perform the reduction modulo two throughout the computation of the summation polynomial instead only at the end. Hence, we do not use this construction in practice.

**Remark 7.10.** Note that in characteristic two, these symmetries do not correspond to several solutions of the systems to solve (or equivalently to the PDP) but they correspond to multiplicity of the solutions. This can be explained geometrically (the point behind these symmetries is projected onto the neutral element) and algebraically ( $s_i = e_i^2$  and it is obvious that an equation of the form  $x^2 + \alpha = 0$  has a unique solution of multiplicity two in characteristic two).

We now handle the action of the rational two-torsion point of binary curves in universal Edwards model.

### Combining inherited symmetries with the action of the two-torsion

Let  $T_2 = (0, 1)$  be the rational two-torsion point of a binary elliptic curve in universal Edwards model. This point acts on the points of the curve by  $(x, y) \oplus T_2 = \left(\frac{1}{x}, \frac{1}{y}\right)$ . In order to get an affine action of the two-torsion point, we consider the following change of coordinates:

$$\begin{aligned} \phi_\gamma : E(\mathbb{F}_{2^k}) &\rightarrow E_\gamma(\mathbb{F}_{2^k}) & \phi_\gamma^{-1} : E_\gamma(\mathbb{F}_{2^k}) &\rightarrow E(\mathbb{F}_{2^k}) \\ (x, y) &\mapsto \left(\frac{1}{x+1} + \gamma, y\right) & (X, Y) &\mapsto \left(\frac{1}{X+\gamma} + 1, Y\right) \end{aligned} \quad (7.2)$$

with  $\gamma \in \mathbb{K}$ . The two-torsion point of  $E_\gamma$  is then  $T'_2 = \phi_\gamma(T_2) = (\gamma + 1, 1)$ . Let  $P' = (X, Y)$  be a point of  $E_\gamma$  we have

$$P' \oplus T'_2 = \phi_\gamma(\phi_\gamma^{-1}(P') \oplus T_2) = \left(X + 1, \frac{1}{Y}\right).$$

From Proposition 7.1, the action of the two-torsion point of  $E_\gamma$  implies particular symmetries on its summation polynomials. In the following, our aim is to show that for a “good” choice of  $\gamma$ , these symmetries are consistent with those inherited from rational curves.

Note that  $\phi_\gamma$  changes only the  $x$ -coordinate of a point, so we defined the map  $\varphi_\gamma : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$  accordingly.

**Definition 7.11.** The map  $\varphi_\gamma$  is defined as

$$\begin{aligned} \varphi_\gamma: \mathbb{F}_{2^k} &\rightarrow \mathbb{F}_{2^k} & \varphi_\gamma^{-1}: \mathbb{F}_{2^k} &\rightarrow \mathbb{F}_{2^k} \\ x &\mapsto \frac{1}{x+1} + \gamma & X &\mapsto \frac{1}{X+\gamma} + 1 \end{aligned}$$

Let denote by  $Sb_{n,\gamma}$  the  $n$ th summation polynomial of  $E_\gamma$ . Since  $E_\gamma$  is obtained by applying the change of coordinates  $\phi$  on  $E$ , the polynomial  $Sb_{n,\gamma}$  can be computed by applying the change of coordinates  $\varphi_\gamma^{-1}$  on  $Sb_n$ .

Note that  $\varphi_\gamma^{-1}$  is not an affine map and  $Sb_n(\varphi_\gamma^{-1}(X_1), \dots, \varphi_\gamma^{-1}(X_n))$  is no longer a polynomial. This is not restrictive since summation polynomials are defined by their solutions. Let  $\frac{N(X_1, \dots, X_n)}{D(X_1, \dots, X_n)}$  be the irreducible form of  $Sb_n(\varphi_\gamma^{-1}(X_1), \dots, \varphi_\gamma^{-1}(X_n))$  we then have  $Sb_{n,\gamma} = N$ .

From Proposition 7.8  $Sb_n$  can be expressed in terms of  $s_1, \dots, s_{n-1}, e_n$ . In particular, this implies that  $Sb_n$  can be expressed in terms of  $x_1^2, \dots, x_n^2, x_1 \cdots x_n$ . Let  $F_n \in \mathbb{F}_{2^k}[y_1, \dots, y_{n+1}]$  be such an expression of  $Sb_n$ . That is to say  $F_n(x_1^2, \dots, x_n^2, x_1 \cdots x_n) = Sb_n$ . We thus have

$$Sb_{n,\gamma}(X_1, \dots, X_n) = \text{Numerator} \left( F_n(\varphi_\gamma^{-1}(X_1)^2, \dots, \varphi_\gamma^{-1}(X_n)^2, \varphi_\gamma^{-1}(X_1) \cdots \varphi_\gamma^{-1}(X_n)) \right). \quad (7.3)$$

**Lemma 7.12.** If  $\gamma \in \{0, 1\}$  then  $\varphi_\gamma^{-1}(X_1) \cdots \varphi_\gamma^{-1}(X_n)$  is a rational fraction in terms of  $e_n^2(X_1, \dots, X_n) = X_1^2 \cdots X_n^2$  and  $e_n(X_1^2 + X_1, \dots, X_n^2 + X_n)$ .

*Proof.* For any  $i = 1, \dots, n$  we have  $\varphi_\gamma^{-1}(X_i) = \frac{1}{X_i + \gamma} + 1 = \frac{1+X_i+\gamma}{X_i+\gamma}$ . Hence,

$$\varphi_\gamma^{-1}(X_i) = \begin{cases} \frac{1+X_i}{X_i} = \frac{X_i+X_i^2}{X_i^2} & \text{if } \gamma = 0 \\ \frac{X_i}{X_i+1} = \frac{X_i^2}{X_i^2+X_i} & \text{if } \gamma = 1 \end{cases}.$$

By consequence,

$$\varphi_\gamma^{-1}(X_1) \cdots \varphi_\gamma^{-1}(X_n) = \begin{cases} \prod_{i=1}^n \frac{X_i+X_i^2}{X_i^2} = \frac{e_n(X_1^2+X_1, \dots, X_n^2+X_n)}{e_n^2(X_1, \dots, X_n)} & \text{if } \gamma = 0 \\ \prod_{i=1}^n \frac{X_i^2}{X_i^2+X_i} = \frac{e_n^2(X_1, \dots, X_n)}{e_n(X_1^2+X_1, \dots, X_n^2+X_n)} & \text{if } \gamma = 1 \end{cases}.$$

□

Since we work in binary fields for any  $i = 1, \dots, n$  we have  $\varphi_\gamma^{-1}(X_i)^2 = \frac{1}{X_i^2 + \gamma^2} + 1$ . By consequence the equation (7.3) and Lemma 7.12 imply that  $Sb_{n,\gamma}$  can be expressed in terms of  $X_1^2, \dots, X_n^2$  and  $e_n(X_1^2 + X_1, \dots, X_n^2 + X_n)$ .

**Proposition 7.13.** For any  $n \geq 3$ , the  $n$ th summation polynomial of  $E_\gamma$  with  $\gamma \in \{0, 1\}$  can be expressed in terms of  $e_1(X_1^2, \dots, X_n^2)$ ,  $e_n(X_1^2 + X_1, \dots, X_n^2 + X_n)$  and  $e_i(X_1^4 + X_1^2, \dots, X_n^4 + X_n^2)$  for  $i = 2, \dots, n-1$ .

*Proof.* From Proposition 7.1  $Sb_{n,\gamma}$  is invariant by any even number of transformations of the form  $\tau: X_j \mapsto X_j + 1$  and can thus be expressed in terms of  $e_1(X_1, \dots, X_n)$  and  $E_i(X_1, \dots, X_n) = e_i(X_1^2 + X_1, \dots, X_n^2 + X_n)$  for  $i = 2, \dots, n$ . Hence,

$$Sb_{n,\gamma}(X_1, \dots, X_n) = \sum_{i=0}^d f_i(e_1, E_2, \dots, E_{n-1}) E_n^i$$

where  $f_i(e_1, E_2, \dots, E_n) = \xi_i(X_1, \dots, X_n)$  is invariant by any even number of transformations  $\tau$ . Moreover, we have  $\xi_i(X_1, \dots, X_n) = \psi_i(X_1^2, \dots, X_n^2)$  and in characteristic two  $(\tau(X_j))^2 = X_j^2 + 1 = \tau(X_j^2)$ . By consequence,

$$\begin{aligned} \xi_i(X_1, \dots, X_n) &= \xi_i(\tau^{\epsilon_1}(X_1), \dots, \tau^{\epsilon_n}(X_n)) \\ &= \psi_i(\tau^{\epsilon_1}(X_1^2), \dots, \tau^{\epsilon_n}(X_n^2)) \\ &= \psi_i(X_1^2, \dots, X_n^2) \end{aligned}$$

for any  $(\epsilon_1, \dots, \epsilon_n) \in (\mathbb{Z}/2\mathbb{Z})^n$  such that  $\sum_{i=1}^n \epsilon_i = 0 \pmod{2}$ . Therefore, for  $i = 0, \dots, d$  the polynomial  $\psi_i(y_1, \dots, y_n)$  is invariant by any even number of transformations of the form  $y_j \mapsto y_j + 1$  and can thus be expressed in terms of  $e_1(y_1, \dots, y_n)$  and  $e_j(y_1^2 + y_1, \dots, y_n^2 + y_n)$  for  $j = 2, \dots, n$ . That is to say there exists  $g_i$  such that  $\psi_i(y_1, \dots, y_n) = g_i(e_1(y_1, \dots, y_n), e_2(y_1^2 + y_1, \dots, y_n^2 + y_n), \dots, e_n(y_1^2 + y_1, \dots, y_n^2 + y_n))$ . Finally, we have

$$\begin{aligned} Sb_{n,\gamma}(X_1, \dots, X_n) &= \sum_{i=0}^d \psi_i(X_1^2, \dots, X_n^2) e_n(X_1^2 + X_1, \dots, X_n^2 + X_n)^i \\ &= \sum_{i=0}^d g_i(s_1, S_2, \dots, S_n) e_n(X_1^2 + X_1, \dots, X_n^2 + X_n)^i \end{aligned}$$

where  $s_1 = e_1(X_1^2, \dots, X_n^2)$  and  $S_j = e_j(X_1^4 + X_1^2, \dots, X_n^4 + X_n^2)$  for  $j = 2, \dots, n$ .  $\square$

**Example 7.14.** Assume  $\gamma = 0$ , for  $n = 3$  we have

$$Sb_3 = t(x_1^2 x_2^2 + x_1^2 x_3^2 + x_2^2 x_3^2) + x_1 x_2 x_3 + t$$

then

$$\begin{aligned} Sb_3(\varphi_0^{-1}(X_1), \varphi_0^{-1}(X_2), \varphi_0^{-1}(X_3)) &= \frac{t(1 + X_1^2)(1 + X_2^2)}{X_1^2 X_2^2} + \frac{t(1 + X_1^2)(1 + X_3^2)}{X_1^2 X_3^2} + t + \\ &\quad \frac{t(1 + X_2^2)(1 + X_3^2)}{X_2^2 X_3^2} + \frac{(1 + X_1)(1 + X_2)(1 + X_3)}{X_1 X_2 X_3} \end{aligned}$$

which implies that

$$\begin{aligned} Sb_{3,0} &= t(X_3^2(1 + X_1^2)(1 + X_2^2) + X_2^2(1 + X_1^2)(1 + X_3^2) + X_1^2(1 + X_2^2)(1 + X_3^2)) + \\ &\quad X_1 X_2 X_3 (1 + X_1)(1 + X_2)(1 + X_3) + t X_1^2 X_2^2 X_3^2 \\ &= e_3(X_1^2 + X_1, X_2^2 + X_2, X_3^2 + X_3) + t(X_3^2 + X_2^2 + X_1^2) \\ &= e_3(X_1^2 + X_1, X_2^2 + X_2, X_3^2 + X_3) + t e_1(X_1^2, X_2^2, X_3^2). \end{aligned}$$

From now on, we consider only summation polynomials of binary elliptic curves in universal Edwards model after the change of coordinates  $\phi_\gamma$  defined in equation (7.2) with  $\gamma = 0$ . Hence, we no longer use the notation  $Sb_n$  or  $Sb_{n,\gamma}$  but simply  $S_n$  since there is no ambiguity. By consequence, for any  $n \geq 3$ ,  $S_n$  admits the change of variables  $\Omega_{n,k}$  for any  $2 \leq k \leq n$  defined as follows.

**Definition 7.15.** *The change of variables  $\Omega_{n,k}$  with  $n \geq 3$  and  $2 \leq k \leq n$  from  $\mathbb{K}[x_1, \dots, x_n]$  to  $\mathbb{K}[s_1, S_2, \dots, S_{k-1}, E_k, x_{k+1}, \dots, x_n]$  is defined by*

$$\begin{aligned} \Omega_{n,k}^{-1} : \quad \mathbb{K}[s_1, S_2, \dots, S_{k-1}, E_k, x_{k+1}, \dots, x_n] &\rightarrow \mathbb{K}[x_1, \dots, x_n] \\ s_1 &\mapsto e_1(x_1^2, \dots, x_k^2) \\ S_2 &\mapsto e_2(x_1^4 + x_1^2, \dots, x_k^4 + x_k^2) \\ &\vdots \\ S_{k-1} &\mapsto e_{k-1}(x_1^4 + x_1^2, \dots, x_k^4 + x_k^2) \\ E_k &\mapsto e_n(x_1^2 + x_1, \dots, x_k^2 + x_k) \\ x_{k+1}, \dots, x_n &\mapsto x_{k+1}, \dots, x_n \end{aligned} \quad (7.4)$$

We call totally symmetrized the  $n$ th summation polynomial expressed in terms of the change of coordinates  $\Omega_{n,n}$  and we denote it  $S_n^t = \Omega_{n,n}(S_n) \in \mathbb{K}[s_1, S_2, \dots, S_{n-1}, E_n]$ .

We call partially symmetrized the  $n$ th summation polynomial expressed in terms of the change of coordinates  $\Omega_{n,n-1}$  and we denote it  $S_n^p = \Omega_{n,n-1}(S_n) \in \mathbb{K}[s_{1,n-1}, S_{2,n-1}, \dots, S_{n-2,n-1}, E_{n-1,n-1}, x_n]$ .

In the next section, we investigate the impact of such a compact representation on the density of summation polynomials.

### 7.1.2 Density

In Table 7.1 we give for  $n = 3, \dots, 7$  the degree in each variables of  $S_n^t$ . We give also the total degree of the summation polynomial, the number of terms it contains and finally its density. All the degrees (in each variables or total) are given experimentally. That is to say the values given in Table 7.1 are the exact degrees of summation polynomials and not bounds.

To compute the density of the  $n$ th summation polynomial we have compared its number of monomials with the number of monomials of a dense polynomial of same degree (total and in each variable). Let  $d_i = \deg_i S_n^t$  be the degree of  $S_n^t$  in the  $i$ th variable. Let  $\mathcal{I} = \langle x_1^{d_1+1}, \dots, x_n^{d_n+1} \rangle \subset R$ . Following notations of Definition 2.72, the number  $N$  of monomials  $m$  satisfying  $\deg(m) \leq d$  and  $\deg_i(m) \leq d_i$  for  $i = 1, \dots, n$  is given by  $N = \sum_{k=0}^d \dim_{\mathbb{K}}(R_k/\mathcal{I}_k)$  which can be read on the Hilbert series of  $R/\mathcal{I}$ , see Definition 2.72 and Theorem 2.75.

| $n$ | Degree in each variable | Total degree | Number of monomials | Density |
|-----|-------------------------|--------------|---------------------|---------|
| 3   | (1,0,1)                 | 1            | 2                   | 66.67%  |
| 4   | (2,0,1,1)               | 2            | 3                   | 37.50%  |
| 5   | (4,1,2,1,4)             | 4            | 9                   | 11.39%  |
| 6   | (8,2,4,2,4,5)           | 8            | 50                  | 2.61%   |
| 7   | (16,5,8,4,8,5,16)       | 16           | 2247                | 1.32%   |

Table 7.1: Density of summation polynomials of binary elliptic curves expressed in terms of the polynomial change of variables  $\Omega_{n,n}$ .

From Table 7.1 we can observe that except for very small  $n$ , due to their compact representation, summation polynomials are really sparse.

In the next section we show that the usual method presented in Chapter 5 to compute summation polynomials breaks this compact representation in the sense that it involves much bigger polynomials than the output.

## 7.2 Compact summation polynomials by resultant and Gröbner bases

In Section 5.6.2 it is described how to compute the  $n$ th summation polynomials by applying the change of coordinates given by the symmetric group throughout the computation. This allows to reduce the size of the resultant. Moreover, the Gröbner basis computations, required to express the summation polynomial in terms of this change of coordinates, will be less difficult.

To compute  $S_n^t \in \mathbb{F}_{2^k}[s_1, S_2, \dots, S_{n-1}, E_n]$ , one can proceed similarly. Let denote  $s_{1,k}, S_{2,k}, \dots, S_{k-1,k}, E_{k,k}$  (resp.  $s_{1,n-k}, S_{2,n-k}, \dots, S_{n-k-1,n-k}, E_{n-k,n-k}$ ) the new variables induced by the change of variables  $\Omega_{k,k}$  on  $\{x_1, \dots, x_k\}$  (resp.  $\Omega_{n-k,n-k}$  on  $\{x_{k+1}, \dots, x_n\}$ ). The following resultant computation gives an expression of the  $n$ th summation polynomial in terms of  $s_{1,k}, S_{2,k}, \dots, S_{k-1,k}, E_{k,k}$  and  $s_{1,n-k}, S_{2,n-k}, \dots, S_{n-k-1,n-k}, E_{n-k,n-k}$ :

$$\text{Res}_X \left( \begin{matrix} S_{n-k+1}^p(s_{1,n-k}, S_{2,n-k}, \dots, S_{n-k-1,n-k}, E_{n-k,n-k}, X), \\ S_{k+1}^p(s_{1,k}, S_{2,k}, \dots, S_{k-1,k}, E_{k,k}, X) \end{matrix} \right). \quad (7.5)$$

In order to compute  $S_n^t$  we just have to find the corresponding change of variables to apply to this polynomial to expressed it in terms of  $s_1, S_2, \dots, S_{n-1}, E_n$ . Let us denote

$$\begin{aligned} \mathbf{X}_1 &= (x_1^2, \dots, x_{n-k}^2) & \mathbf{Z}_2 &= (x_{n-k+1}^2 + x_{n-k+1}, \dots, x_n^2 + x_n) \\ \mathbf{Y}_1 &= (x_1^4 + x_1^2, \dots, x_{n-k}^4 + x_{n-k}^2) & \mathbf{X}_3 &= (x_1^2, \dots, x_n^2) \\ \mathbf{Z}_1 &= (x_1^2 + x_1, \dots, x_{n-k}^2 + x_{n-k}) & \mathbf{Y}_3 &= (x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) \\ \mathbf{X}_2 &= (x_{n-k+1}^2, \dots, x_n^2) & \mathbf{Z}_3 &= (x_1^2 + x_1, \dots, x_n^2 + x_n) \\ \mathbf{Y}_2 &= (x_{n-k+1}^4 + x_{n-k+1}^2, \dots, x_n^4 + x_n^2) \end{aligned}.$$

Since the field is of characteristic two, spreading the symmetrization throughout the computation can be done by noting that:

$$\begin{aligned} e_1(\mathbf{X}_3) &= e_1(\mathbf{X}_1) + e_1(\mathbf{X}_2) \\ e_2(\mathbf{Y}_3) &= e_2(\mathbf{Y}_1) + e_2(\mathbf{Y}_2) + \alpha_1 \alpha_2 \\ e_3(\mathbf{Y}_3) &= e_3(\mathbf{Y}_1) + e_3(\mathbf{Y}_2) + \alpha_1 e_2(\mathbf{Y}_2) + \alpha_2 e_2(\mathbf{Y}_1) \\ &\vdots \\ e_{n-2}(\mathbf{Y}_3) &= e_{n-k}(\mathbf{Z}_1)^2 e_{k-2}(\mathbf{Y}_2) + e_{n-k-1}(\mathbf{Y}_1) e_{k-1}(\mathbf{Y}_1) + e_{n-k-2}(\mathbf{Y}_1) e_k(\mathbf{Z}_1)^2 \\ e_{n-1}(\mathbf{Y}_3) &= e_{n-k}(\mathbf{Z}_1)^2 e_{k-1}(\mathbf{Y}_2) + e_{n-k-1}(\mathbf{Y}_1) e_k(\mathbf{Z}_1)^2 \\ e_n(\mathbf{Z}_3) &= e_{n-k}(\mathbf{Z}_1) e_k(\mathbf{Z}_2) \end{aligned} \quad (7.6)$$

where  $\alpha_1 = e_1(\mathbf{X}_1)^2 + e_1(\mathbf{X}_1)$  and  $\alpha_2 = e_1(\mathbf{X}_2)^2 + e_1(\mathbf{X}_2)$ .

According to Chapter 2 Section 2.1.4 applying this corresponding change of variables can be done in two ways by using either elimination ideals (Algorithm 2) or normal forms (Algorithm 3).

### Benchmarks

Using the algorithm presented above, if the parameter  $t$  of the curve is not instantiated *i.e.*  $S_n^t \in \mathbb{F}_2[s_1, S_2, \dots, S_{n-1}, E_n, t]$ ; by using MAGMA one can compute to the sixth summation polynomial.

In Table 7.2 (respectively 7.3) we present timings to compute  $S_n^t$  for  $n \leq 6$ . We give the time to compute the resultant in equation (7.5) (column “Res”) and its size (column “#Res”).

In column “Gröbner basis” we give the time to compute the Gröbner basis  $\mathcal{G}_{>n+1}$  involved in Algorithm 2 or Algorithm 3 (note that these two Gröbner bases are not the same). The column “ $\#S_n^t$ ” (respectively “ $\#S_{k+1}^p$ ” and “ $\#S_{n-k+1}^p$ ”) contains the number of monomials in  $S_n^t$  (respectively  $S_{k+1}^p$  and  $S_{n-k+1}^p$ ). The column  $S_n^t$  contains the total time to compute this polynomial when it is in  $\mathbb{F}_2[s_1, S_2, \dots, S_{n-1}, E_n, t]$ . In Table 7.3 we also give the time to compute the normal form of the resultant w.r.t. the Gröbner basis  $\mathcal{G}_{>n+1}$  (column “NF”). For each  $n$  we assume that  $S_{k+1}^p$  and  $S_{n-k+1}^p$  is known for any  $k \in \{2, \dots, n-2\}$ . By consequence, the time to compute them is not take into account.

| $n$ | $k$ | $\#S_{k+1}^p$ | $\#S_{n-k+1}^p$ | Res    | #Res | Gröbner basis | $S_n^t$ | $\#S_n^t$ |
|-----|-----|---------------|-----------------|--------|------|---------------|---------|-----------|
| 4   | 2   | 4             | 4               | 0.000s | 8    | 0.010s        | 0.010s  | 3         |
| 5   | 2   | 4             | 9               | 0.000s | 37   | 0.010s        | 0.020s  | 9         |
| 6   | 2   | 4             | 47              | 0.390s | 619  | 3.380s        | 3.780s  | 51        |
| 6   | 3   | 9             | 9               | 0.020s | 686  | 7.860s        | 7.900s  | 51        |

Table 7.2: CPU time to compute the  $n$ th summation polynomial with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU by using resultant and elimination ideals.

| $n$ | $k$ | $\#S_{k+1}^p$ | $\#S_{n-k+1}^p$ | Res    | #Res | Gröbner basis | NF     | $S_n^t$ | $\#S_n^t$ |
|-----|-----|---------------|-----------------|--------|------|---------------|--------|---------|-----------|
| 4   | 2   | 4             | 4               | 0.000s | 8    | 0.000s        | 0.000s | 0.000s  | 3         |
| 5   | 2   | 4             | 9               | 0.000s | 37   | 0.000s        | 0.000s | 0.000s  | 9         |
| 6   | 2   | 4             | 47              | 0.300s | 619  | 0.020s        | 0.690s | 1.030s  | 51        |
| 6   | 3   | 9             | 9               | 0.010s | 686  | 0.030s        | 4.390s | 4.440s  | 51        |

Table 7.3: CPU time to compute the  $n$ th summation polynomial with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU by using resultant and normal forms.

From Tables 7.2 and 7.3 it seems that the most efficient strategy is to use normal forms. One can notice that this method to compute summation polynomials involves polynomials much larger than the output polynomial. Indeed, for instance for  $n = 6$ , the resultant between  $S_{k+1}^p$  and  $S_{n-k+1}^p$  contains at least 619 terms in comparison to 51 terms for  $S_6^t$ .

For  $n = 7$ , we can still use MAGMA to compute  $S_7^t$ . However, by computing directly the normal form or the elimination ideal by using functions of MAGMA we cannot expressed the resultant in terms of  $s_1, S_2, \dots, S_6, E_7$ . In order to perform the corresponding change of variables we perform by hand some well-chosen normal forms and finally we use elimination ideal to end the symmetrization.

Finally, for  $n = 7$  the resultant between  $S_3^p$  and  $S_6^p$  is computed in 51 seconds and contains 63448 terms. The change of coordinates to express this resultant in terms of  $s_1, S_2, S_3, S_4, S_5, S_6, E_7$  is done in 332 seconds and  $S_7^t$  is computed in 383 seconds and contains 2581 terms.

Nevertheless, the computation of the eighth summation polynomial still seems intractable using this method. To compute it we need to remove Gröbner bases and multivariate resultant computations which become too difficult since they involve bigger polynomials. We will show that by using evaluation-interpolation method we can avoid this kind of computations. In the next section we recall the principle of sparse multivariate interpolation.

### 7.3 Outline of sparse multivariate polynomial interpolation algorithm

The first algorithm for sparse multivariate polynomial interpolation was due to Zippel [Zip79]. This algorithm is probabilistic and has a polynomial time complexity in  $O(ndt^3)$  arithmetic operations and requires  $O(ndt)$  evaluations where  $n$  is the number of variables,  $d$  a bound on the degree in each variables of the polynomial to interpolate and  $t$  is the number of its terms. The first deterministic algorithm for sparse multivariate polynomial interpolation was due to Ben-Or and Tiwari [BOT88] and has an arithmetic complexity in  $O(\tau^2(\log^2 \tau + \log nd))$  where  $\tau$  is an upper bound on the number of terms in the polynomial. The algorithm of Ben-Or and Tiwari requires  $2\tau$  evaluations. Using some ideas of Ben-Or and Tiwari, Zippel [Zip90] then proposed a new version of its probabilistic algorithm whose complexity is in  $O(ndt^2)$  and always requires  $O(ndt)$  evaluations. In [Zip90], the author also proposes deterministic solutions of the zero avoidance problem that he uses to adapt his probabilistic algorithm for sparse multivariate polynomial interpolation. He gets a deterministic algorithm whose complexity is in  $O(ndt^2\tau)$  and which requires  $O(ndt\tau)$  evaluations. Finally, in [KL89], Kaltofen and Lakshman proposed new efficient algorithms to find the rank and solve a special Toeplitz system arising in Ben-Or and Tiwari algorithm and solve a transposed Vandermonde system arising in Zippel's algorithm. These algorithms allow to decrease the complexity of Ben-Or and Tiwari's algorithm to  $O(dn\tau M(\tau) \log(\tau) \log(n))$  arithmetic operations and Zippel's algorithm to  $O(dnM(t) \log(t))$  arithmetic operations where  $M(\ell)$  denotes the complexity of multiplying two univariate polynomials of degree  $\ell$  which is quasi-linear in  $\ell$ , see [CK91, VZGG03].

We cannot predict the sparsity of summation polynomials *i.e.* we do not know a sharp bound  $\tau$  on the number of terms. By consequence, in order to take advantage of the sparsity of these polynomials we use as multivariate polynomial interpolation algorithm the probabilistic algorithm of Zippel whose complexity and number of evaluations does not depend on  $\tau$ .

#### 7.3.1 Description of Zippel's sparse multivariate polynomial interpolation algorithm

The principle of Zippel's algorithm [Zip79, Zip90] is to interpolate the multivariate polynomial one variable at a time by using dense univariate interpolation. From now on  $\tilde{x}$  denotes the evaluation of the variable  $x$  at some element of the field  $\mathbb{K}$ .

Let  $f(x_1, \dots, x_n) \in \mathbb{K}[x_1, \dots, x_n]$  be the polynomial to interpolate and  $d_i$  be a bound on the degree of  $f$  in  $x_i$ . First we choose an initial evaluation point  $(\tilde{x}_{1,0}, \dots, \tilde{x}_{n,0})$ . At step  $i$  we want to recover the polynomial  $f(x_1, \dots, x_i, \tilde{x}_{i+1,0}, \dots, \tilde{x}_{n,0})$  denoted  $f_i$  assuming we know the polynomial  $f_{i-1} = f(x_1, \dots, x_{i-1}, \tilde{x}_{i,0}, \dots, \tilde{x}_{n,0})$ .

We can write  $f_i$  as a polynomial in  $x_1, \dots, x_{i-1}$  and coefficients in  $\mathbb{K}(x_i)$  as follows  $f_i = \sum_{j=1}^{t_{i-1}} c_{\alpha_j} x^{\alpha_j}$  where  $c_{\alpha_j}$  is a univariate polynomial in  $x_i$ , the monomial  $x^{\alpha_j}$  satisfies  $\alpha_i = \dots = \alpha_n = 0$  for  $j = 1, \dots, t_{i-1}$  and  $t_{i-1} \leq t$  is the number of terms in  $f_{i-1}$ . To recover  $f_i$  we have to interpolate each coefficients  $c_{\alpha_j}$  for  $j = 1, \dots, t_{i-1}$  as a univariate polynomial in  $x_i$ . In order to interpolate these coefficients, we need to evaluate them at  $d_i + 1$  evaluation points  $\tilde{x}_{i,k}$  with  $k = 0, \dots, d_i$ . That is to say we need to compute all the polynomials  $f_{i-1,k} = f(x_1, \dots, x_{i-1}, \tilde{x}_{i,k}, \tilde{x}_{i+1,0}, \dots, \tilde{x}_{n,0}) = \sum_{j=1}^{t_{i-1}} c_{\alpha_{j,k}} x^{\alpha_j}$  where  $c_{\alpha_{j,k}} = c_{\alpha_j}(\tilde{x}_{i,k})$ . Once the polynomials  $f_{i-1,k}$  for  $k = 0, \dots, d_i$  are computed we can recover  $f_i$  by interpolating  $t_{i-1}$  dense univariate polynomials (each coefficient  $c_{\alpha_j}$  for  $j = 1, \dots, t_{i-1}$ ).

We now investigate how to efficiently compute all the polynomials  $f_{i-1,k}$  for  $k = 0, \dots, d_i$ . First we note that  $f_{i-1} = f_{i-1,0}$  and we have to compute only  $d_i$  polynomials. To compute  $f_{i-1,k}$  for some  $k \in \{1, \dots, d_i\}$  we randomly choose an evaluation point for all  $f_{i-1,k}$  that is to say we randomly choose  $\tilde{x}_{1,i}, \dots, \tilde{x}_{i-1,i}$  in  $\mathbb{K}$  and we denote by  $\tilde{x}^\alpha$  the evaluation of the monomial  $x^\alpha$  at  $\tilde{x}_{1,i}, \dots, \tilde{x}_{i-1,i}$ . Then we compute  $v_{i,k,\ell} = f_{i-1,k}(\tilde{x}_{1,i}^\ell, \dots, \tilde{x}_{i-1,i}^\ell) = \sum_{j=1}^{t_{i-1}} c_{\alpha_{j,k}} \tilde{x}^{\ell \alpha_j} = f(\tilde{x}_{1,i}^\ell, \dots, \tilde{x}_{i-1,i}^\ell, \tilde{x}_{i,k}, \tilde{x}_{i+1,0}, \dots, \tilde{x}_{n,0})$  for  $\ell = 0, \dots, t_{i-1} - 1$  and we obtain the following transposed Vandermonde system:

$$\left\{ \begin{array}{cccccc} c_{\alpha_{1,k}} & + & c_{\alpha_{2,k}} & + & \dots & + & c_{\alpha_{t_{i-1},k}} & = & v_{i,k,0} \\ c_{\alpha_{1,k}} \tilde{x}^{\alpha_1} & + & c_{\alpha_{2,k}} \tilde{x}^{\alpha_2} & + & \dots & + & c_{\alpha_{t_{i-1},k}} \tilde{x}^{\alpha_{t_{i-1}}} & = & v_{i,k,1} \\ c_{\alpha_{1,k}} \tilde{x}^{2\alpha_1} & + & c_{\alpha_{2,k}} \tilde{x}^{2\alpha_2} & + & \dots & + & c_{\alpha_{t_{i-1},k}} \tilde{x}^{2\alpha_{t_{i-1}}} & = & v_{i,k,2} \\ & & & & & & \vdots & & \\ c_{\alpha_{1,k}} \tilde{x}^{t_{i-1}\alpha_1} & + & c_{\alpha_{2,k}} \tilde{x}^{t_{i-1}\alpha_2} & + & \dots & + & c_{\alpha_{t_{i-1},k}} \tilde{x}^{t_{i-1}\alpha_{t_{i-1}}} & = & v_{i,k,t_{i-1}} \end{array} \right. \quad (7.7)$$

Let denote  $\tilde{x}^{\alpha_j}$  by  $\pi_j$ , the linear system in equation (7.7) can be represented in matrix form as follows:

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ \pi_1 & \pi_2 & \dots & \pi_{t_{i-1}} \\ \pi_1^2 & \pi_2^2 & \dots & \pi_{t_{i-1}}^2 \\ \vdots & \vdots & & \vdots \\ \pi_1^{t_{i-1}} & \pi_2^{t_{i-1}} & \dots & \pi_{t_{i-1}}^{t_{i-1}} \end{pmatrix} \cdot \begin{pmatrix} c_{\alpha_{1,k}} \\ c_{\alpha_{2,k}} \\ c_{\alpha_{3,k}} \\ \vdots \\ c_{\alpha_{t_{i-1},k}} \end{pmatrix} = \begin{pmatrix} v_{i,k,0} \\ v_{i,k,1} \\ v_{i,k,2} \\ \vdots \\ v_{i,k,t_{i-1}} \end{pmatrix}. \quad (7.8)$$

It is well-known that a Vandermonde matrix is non-singular if  $\pi_i \neq \pi_j$  for all  $i \neq j$  and  $1 \leq i, j \leq t_{i-1}$ . Thus, in order to get an invertible matrix we choose  $\tilde{x}_{1,i}, \dots, \tilde{x}_{i-1,i}$  such that  $\tilde{x}^{\alpha_j}$  for  $j = 1, \dots, t_{i-1}$  are all distinct. Note that whatever the value of  $k$ , the Vandermonde matrix in equation (7.8) does not change.

Finally, by solving the system in equation (7.8) for all  $k = 1, \dots, d_i$  we can compute all the polynomials  $f_{i-1,k}$  and perform the univariate polynomial interpolation on each coefficient of  $f_{i-1}$  to recover  $f_i$ . Zippel's algorithm is summarized in Algorithm 20.

**Example 7.16.** Assume one wants to interpolate  $f = 8x_1^2 + 7x_1x_2 + 12x_2^2 + 6$ , a polynomial in  $\mathbb{F}_{13}[x_1, x_2]$ . Assume we know a bound  $d_i = 2$  on  $\deg_{x_i}(f)$  for  $i = 1, 2$  and we have a black box to evaluate  $f$  in any points.

First we randomly choose  $\tilde{x}_{1,0} = 2$  and  $\tilde{x}_{2,0} = 9$  and we compute  $f_0 = f(\tilde{x}_{1,0}, \tilde{x}_{2,0}) = 5$ .

Step 1: we want to compute  $f_1 = f(x_1, \tilde{x}_{2,0})$ . We proceed by univariate polynomial interpolation. That is to say we randomly choose  $\tilde{x}_{1,1} = 11$  and  $\tilde{x}_{1,2} = 10$  and we compute  $f_{0,1} = f(\tilde{x}_{1,1}, \tilde{x}_{2,0}) = 0$  and  $f_{0,2} = f(\tilde{x}_{1,2}, \tilde{x}_{2,0}) = 3$ . Since  $f_1$  is a univariate polynomial of degree at most 2 and we know its evaluation in 3 distinct points we compute it using the Lagrange interpolation see for instance [Fid72, MB72, BM74] and we find  $f_1 = 8x_1^2 + 11x_1 + 3$ .

Step 2: we want to compute  $f_2 = f(x_1, x_2)$  and we know  $f_1 = f(x_1, \tilde{x}_{2,0}) = 8x_1^2 + 11x_1 + 3$ . We randomly choose  $\tilde{x}_{1,2} = 4$ , we check that  $\pi_1 = \tilde{x}_{1,2}^2 = 3 \neq \pi_2 = \tilde{x}_{1,2} = 4 \neq \pi_3 = \tilde{x}_{1,2}^0 = 1$  and we construct the transpose Vandermonde matrix

$$V_T = \begin{pmatrix} 1 & 1 & 1 \\ 3 & 4 & 1 \\ 9 & 3 & 1 \end{pmatrix}.$$

**Algorithm 20:** Sparse multivariate polynomial interpolation: Zippel's algorithm.

---

**Input** :  $\text{Eval}_f$  a function evaluating the polynomial  $f \in \mathbb{K}[x_1, \dots, x_n]$ ,  
 $d_i$  a bound on  $\deg_i(f)$  for  $i = 1, \dots, n$  and  
 $\text{Interpolation}([a_1, \dots, a_d], [v_1, \dots, v_d], x_i)$  a function which returns the unique univariate polynomial  $p$  in  $x_i$  of degree at most  $d - 1$  such that  $p(a_j) = v_j$  for  $j = 1, \dots, d$ .

**Output:** The polynomial  $f \in \mathbb{K}[x_1, \dots, x_n]$  or *fail*.

```

1 Randomly choose  $\tilde{x}_{1,0}, \dots, \tilde{x}_{n,0}$  and  $\tilde{x}_{1,1}, \dots, \tilde{x}_{1,d_1}$  in  $\mathbb{K}$  s.t.  $\tilde{x}_{1,j}$  are distinct;
2 for  $k := 0$  to  $d_1$  do  $v_k := \text{Eval}_f(\tilde{x}_{1,k}, \tilde{x}_{2,0}, \dots, \tilde{x}_{n,0})$ ;
3  $F := \text{Interpolation}([\tilde{x}_{1,0}, \dots, \tilde{x}_{1,d_1}], [v_0, \dots, v_{d_1}], x_1)$ ;
4 for  $i := 2$  to  $n$  do
5    $t :=$  Number of monomials in  $F$ ;
6    $m_0, \dots, m_{t-1} :=$  monomials of  $F$ ;
7    $\alpha_0, \dots, \alpha_{t-1} :=$  coefficients of  $m_0, \dots, m_{t-1}$  in  $F$ ;
8   Randomly choose  $\tilde{x}_1, \dots, \tilde{x}_{i-1}$  in  $\mathbb{K}$ ;
9   for  $\ell := 0$  to  $t - 1$  do  $\pi_\ell :=$  Evaluation of  $m_\ell$  in  $(\tilde{x}_1, \dots, \tilde{x}_{i-1})$ ;
10  if  $\pi_\ell \neq \pi_j$  for all  $j \neq \ell$  then
11     $V :=$  Vandermonde matrix associated to  $\pi_0, \dots, \pi_{t-1}$ ;
12    Randomly choose  $\tilde{x}_{i,1}, \dots, \tilde{x}_{i,d_i}$  in  $\mathbb{K}$  s.t.  $\tilde{x}_{i,k}$  are distinct;
13    for  $k := 1$  to  $d_i$  do
14      for  $\ell := 0$  to  $t - 1$  do
15         $v_{k,\ell} := \text{Eval}_f(\tilde{x}_1^\ell, \dots, \tilde{x}_{i-1}^\ell, \tilde{x}_{i,k}, \tilde{x}_{i+1,0}, \dots, \tilde{x}_{n,0})$ ;
16      Find  $(c_{0,k}, \dots, c_{t-1,k})^t$  a solution of  $V^t \cdot \mathbf{c} = (v_{k,0}, \dots, v_{k,t-1})^t$ ;
17    for  $\ell := 0$  to  $t - 1$  do
18       $\alpha_\ell := \text{Interpolation}([\tilde{x}_{i,0}, \dots, \tilde{x}_{i,d_i}], [\alpha_\ell, c_{\ell,1}, \dots, c_{\ell,d_i}], x_i)$ ;
19     $F := \sum_{\ell=0}^{t-1} \alpha_\ell m_\ell$ ;
20  else return fail;
21 return  $F$ ;
```

---

Then, we randomly choose  $\tilde{x}_{2,1} = 1$  and  $\tilde{x}_{2,2} = 4$  and we compute  $v_{2,k,\ell} = f(\tilde{x}_{1,2}^\ell, \tilde{x}_{2,k})$  for  $\ell = 0, 1, 2$  and  $k = 1, 2$  and we obtain the two vectors

$$\begin{pmatrix} v_{2,1,0} \\ v_{2,1,1} \\ v_{2,1,2} \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \\ 7 \end{pmatrix} = \mathbf{v}_1 \text{ and } \begin{pmatrix} v_{2,2,0} \\ v_{2,2,1} \\ v_{2,2,2} \end{pmatrix} = \begin{pmatrix} 0 \\ 9 \\ 3 \end{pmatrix} = \mathbf{v}_2.$$

Afterwards, we solve the two linear systems  $V_T \cdot \mathbf{c}_k = \mathbf{v}_k$  for  $k = 1, 2$  where  $\mathbf{c}_k$  is the column vector  $(c_{1,k}, c_{2,k}, c_{3,k})^t$ . We obtain  $\mathbf{c}_1 = (8, 7, 5)^t$  and  $\mathbf{c}_2 = (8, 2, 3)^t$  and we can reconstruct the two polynomials  $f_{1,k} = f_2(x_1, \tilde{x}_{2,k}) = c_{1,k}x_1^2 + c_{2,k}x_1 + c_{3,k}$ . Hence, at this step we know

$$\begin{aligned} f_{1,0} &= f_2(x_1, \tilde{x}_{2,0}) = 8x_1^2 + 11x_1 + 3 \\ f_{1,1} &= f_2(x_1, \tilde{x}_{2,1}) = 8x_1^2 + 7x_1 + 5 \\ f_{1,2} &= f_2(x_1, \tilde{x}_{2,2}) = 8x_1^2 + 2x_1 + 3 \end{aligned}$$

Finally, we can interpolate each coefficients of  $f_2$  as a univariate polynomial in  $x_2$  using the

Lagrange interpolation and we obtain

$$f_2(x_1, x_2) = (8)x_1^2 + (7x_2)x_1 + (12x_2^2 + 6) = f.$$

### 7.3.2 Complexity and probability of success of Zippel's algorithm

At step  $i$ , we have to solve  $d_i$  transpose Vandermonde systems of size  $t_{i-1} \times t_{i-1} \leq t \times t$ . From [KL89] a transpose Vandermonde system of size  $n \times n$  can be solved in  $O(M(n) \log(n))$  arithmetic operations. Then, we have to interpolate  $t_{i-1}$  univariate polynomials of degree at most  $d_i$  which can be done in  $O(t_{i-1} M(d_i) \log(d_i))$  arithmetic operations. The total number of evaluations required by Zippel's algorithm is given by  $1 + \sum_{i=1}^n d_i t_{i-1}$  which can be bounded by  $O(ndt)$  evaluations. Indeed, to interpolate the first variable we need to evaluate the polynomial to  $d_1 + 1$  distinct points and at step  $i > 1$  we need to evaluate the polynomial to interpolate at  $d_i t_{i-1}$  distinct points. By consequence, we get the following result.

**Theorem 7.17** ([Zip90, KL89]). *Let  $f \in \mathbb{K}[x_1, \dots, x_n]$  and  $d$  be a bound on the degree of  $f$  in each variable. Assuming, we can evaluate the polynomial  $f$ , Zippel's algorithm requires  $O(ndt)$  evaluations of  $f$  to compute it in  $\tilde{O}(ndt)$  arithmetic operations where  $t$  is the number of monomials in  $f$ .*

The probabilistic nature of Zippel's algorithm is twofold. First, at step  $i$  we assume that all zero coefficients of a power of  $x_i$ :  $x_i^s$  in  $c_{\alpha_j}$  for  $j = 1, \dots, t_i - 1$  do not come from the vanishing of a polynomial but means that there is no monomial of the form  $x^{\alpha_j} x_i^s x_{i+1}^{\beta_1} \dots x_n^{\beta_{n-i}}$  in the polynomial  $f$  to interpolate for any  $(\beta_1, \dots, \beta_{n-i}) \in \mathbb{N}^{n-i}$ . Hence, the error in the interpolation depends only on the initial evaluation point  $(\tilde{x}_{1,0}, \dots, \tilde{x}_{n,0})$ . The probability of failure of Zippel's algorithm is then given by the probability that at each step, the coefficients we want to interpolate do not vanish at the initial evaluation point. At step  $i$ , there are at most  $t$  coefficients to interpolate that is to say  $t$  polynomials in  $n - i$  variables whose zeroes must be avoided *i.e.* which should not vanish at  $(\tilde{x}_{1,0}, \dots, \tilde{x}_{n,0})$ . The total number of non zero terms in each of these polynomials is bounded by  $t$  and their degrees in each variables are bounded by  $d = \max\{d_1, \dots, d_n\}$ . Hence, the probability of failure of Zippel's algorithm is bounded by  $\frac{n^2 dt}{q}$  where  $q$  is the size of the finite field  $\mathbb{K}$ , see [Zip90] for details.

**Proposition 7.18** ([Zip90]). *Assume that all the Vandermonde systems involved in Zippel's algorithm to interpolate  $f \in \mathbb{K}[x_1, \dots, x_n]$  are non singular. Then the probability of success of Zippel's algorithm is bounded below by  $1 - \frac{n^2 dt}{q}$  where  $q$  is the size of the field  $\mathbb{K}$ ,  $t$  is the number of monomials in  $f$  and  $d$  is a bound on the degree of  $f$  in each variable.*

The second point which can fail in Zippel's algorithm is at each step  $i$  finding a non singular transpose Vandermonde matrix. That is to say, to find  $\tilde{x}_{1,i}, \dots, \tilde{x}_{i-1,i}$  such that all the  $\pi_i$ 's are distinct. From [Zip90], the probability that such a system is singular is bounded by  $\frac{dt^2}{2q}$ .

**Proposition 7.19** ([Zip90]). *The probability that all the Vandermonde systems involved in Zippel's algorithm to interpolate  $f \in \mathbb{K}[x_1, \dots, x_n]$  are non singular (*i.e.* the algorithm returns a polynomial) is bounded below by  $1 - \frac{ndt^2}{2q}$  where  $q$  is the size of the field  $\mathbb{K}$ ,  $t$  is the number of monomials in  $f$  and  $d$  is a bound on the degree of  $f$  in each variable.*

In the next section, we take advantage of the compact representation (Section 7.1) of summation polynomials to compute them by evaluation-interpolation. In order to use Zippel's algorithm to compute summation polynomials, we need an efficient way to evaluate them.

## 7.4 Summation polynomials by implicit sparse multivariate interpolation

Suppose one wants to evaluate  $S_n^t = \Omega_{n,n}(S_n) \in \mathbb{K}[s_1, S_2, \dots, S_{n-1}, E_n]$  in the evaluation point  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n \in \mathbb{K}$ . If one wants to use equation (7.5) to evaluate  $S_n^t$  one needs to invert the change of coordinates  $\Omega_{n,n}$  in order to find the corresponding evaluation points of  $S_{n-k+1}^p$  and  $S_{k+1}^p$ . That is to say, we look for one solution (as mentioned in the introduction of the chapter, one can choose any solution) in  $\overline{\mathbb{K}}^n$  of the system

$$\left\{ \begin{array}{lcl} \tilde{s}_1 & = & s_{1,n-k} + s_{1,k} \\ \tilde{S}_2 & = & S_{2,n-k} + S_{2,k} + \alpha_1 \alpha_2 \\ \tilde{S}_3 & = & S_{3,n-k} + S_{3,k} + \alpha_1 S_{2,k} + \alpha_2 S_{2,n-k} \\ \tilde{S}_4 & = & S_{4,n-k} + S_{4,k} + \alpha_1 S_{3,k} + \alpha_2 S_{3,n-k} + S_{2,n-k} S_{2,k} \\ & \vdots & \\ \tilde{S}_{n-2} & = & E_{n-k,n-k}^2 S_{k-2,k} + S_{n-k-1,n-k} S_{k-1,k} + S_{n-k-2,n-k} E_{k,k}^2 \\ \tilde{S}_{n-1} & = & E_{n-k,n-k}^2 S_{k-1,k} + S_{n-k-1,n-k} E_{k,k}^2 \\ \tilde{E}_n & = & E_{n-k,n-k} E_{k,k} \end{array} \right. \quad (7.9)$$

where  $\alpha_1 = s_{1,n-k}^2 + s_{1,n-k}$  and  $\alpha_2 = s_{1,k}^2 + s_{1,k}$ .

---

**Algorithm 21:** Evaluating summation polynomials.

---

**Input** : An evaluation point  $(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \in \mathbb{K}^n$  and the summation polynomials partially symmetrized  $S_i^p$  for  $i \in \{3, \dots, n-1\}$ .

**Output:**  $S_n^t(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \in \mathbb{K}$  or *fail*.

- 1 Choose  $k$  in  $\{2, \dots, n-2\}$ ;
  - 2 Find  $\tilde{s}_{1,k}, \tilde{S}_{2,k}, \dots, \tilde{S}_{k-1,k}, \tilde{E}_{k,k}, \tilde{s}_{1,n-k}, \tilde{S}_{2,n-k}, \dots, \tilde{S}_{n-k-1,n-k}, \tilde{E}_{n-k,n-k} \in \mathbb{L}^n$  a solution of the system in equation (7.9) where  $\mathbb{L}$  is an extension of  $\mathbb{K}$ ;
  - 3  $f_1 := S_{k+1}^p(\tilde{s}_{1,k}, \tilde{S}_{2,k}, \dots, \tilde{S}_{k-1,k}, \tilde{E}_{k,k}, X) \in \mathbb{L}[X]$ ;
  - 4  $f_2 := S_{n-k+1}^p(\tilde{s}_{1,n-k}, \tilde{S}_{2,n-k}, \dots, \tilde{S}_{n-k-1,n-k}, \tilde{E}_{n-k,n-k}, X) \in \mathbb{L}[X]$ ;
  - 5 **if**  $\deg(f_1) = \deg_X(S_{k+1}^p)$  **and**  $\deg(f_2) = \deg_X(S_{n-k+1}^p)$  **then**
  - 6     **return** Resultant( $f_1, f_2$ );
  - 7 **else return** *fail*;
- 

The difficulty in evaluating  $S_n^t$  is that one only has an implicit evaluation function. Indeed, the evaluation is not given by a parametrization but it is defined by polynomials equations. Moreover, due to the solving of system in equation (7.9) corresponding to step (2) of Algorithm 21, the workspace can change. For an input in  $\mathbb{K}^n$  and an output in  $\mathbb{K}$  the evaluation function of summation polynomials can work with elements in  $\mathbb{L}$  where  $\mathbb{L}$  is an extension of  $\mathbb{K}$ .

Step (2) of Algorithm 21 can be solved using Gröbner bases as presented in Chapter 2. However, this will be not efficient enough. From Lemma 7.24 the degree of  $S_n^t$  in each variable is at most  $2^{n-3}$ . To compute  $S_n^t$  by evaluation and interpolation we thus need to perform  $O(nt2^{n-3})$  evaluations.

**Remark 7.20.** The evaluation of  $S_8^t$  in 1000 evaluation points in  $(\mathbb{F}_{2^{32}})^8$  using Gröbner bases takes about 1437 seconds. Since,  $S_8^t$  contains exactly 470,369 terms the number of evaluation required to compute  $S_8^t$  is bounded by  $nt2^{n-3} = 120,414,464$ . Consequently, computing  $S_8^t$  may take 5.5 years.

Because of the large number of evaluations required to compute  $S_n^t$  by interpolation, we need an evaluation function as efficient as possible. In the next section, we tackle this issue. We present how to evaluate  $S_n^t$  by using only factorizations and resultants of univariate polynomials and thus how to avoid Gröbner bases computations.

#### 7.4.1 Evaluation of summation polynomials using factorization and resultant of univariate polynomials

In order to solve the system (7.9) more efficiently we use the fact that

$$\begin{aligned}\tilde{s}_1 &= e_1(\tilde{x}_1^2, \dots, \tilde{x}_n^2) \\ \tilde{S}_i &= e_i(\tilde{x}_1^4 + \tilde{x}_1^2, \dots, \tilde{x}_n^4 + \tilde{x}_n^2) \text{ for } i = 1, \dots, n \\ \tilde{E}_n &= e_n(\tilde{x}_1^2 + \tilde{x}_1, \dots, \tilde{x}_n^2 + \tilde{x}_n)\end{aligned}$$

and in the same way

$$\begin{aligned}\tilde{s}_{1,n-k} \mid \tilde{s}_{1,k} &= e_1(\tilde{x}_{i_1}^2, \dots, \tilde{x}_{i_{n-k}}^2) \mid e_1(\tilde{x}_{j_1}^2, \dots, \tilde{x}_{j_k}^2) \\ \tilde{S}_{i,n-k} &= e_i(\tilde{x}_{i_1}^4 + \tilde{x}_{i_1}^2, \dots, \tilde{x}_{i_{n-k}}^4 + \tilde{x}_{i_{n-k}}^2) \text{ for } i = 1, \dots, n-k \\ \tilde{S}_{j,k} &= e_i(\tilde{x}_{j_1}^4 + \tilde{x}_{j_1}^2, \dots, \tilde{x}_{j_k}^4 + \tilde{x}_{j_k}^2) \text{ for } j = 1, \dots, k \\ \tilde{E}_{n-k,n-k} &= e_n(\tilde{x}_{i_1}^2 + \tilde{x}_{i_1}, \dots, \tilde{x}_{i_{n-k}}^2 + \tilde{x}_{i_{n-k}}) \\ \tilde{E}_{k,k} &= e_n(\tilde{x}_{j_1}^2 + \tilde{x}_{j_1}, \dots, \tilde{x}_{j_k}^2 + \tilde{x}_{j_k})\end{aligned}$$

where  $\{i_1, \dots, i_{n-k}, j_1, \dots, j_k\} = \{1, \dots, n\}$ . Moreover, we note that in characteristic two,

$$\begin{aligned}e_1(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) &= \sum_{i=1}^n (x_i^4 + x_i^2) = \sum_{i=1}^n x_i^4 + \sum_{i=1}^n x_i^2 \\ &= e_1^2(x_1^2, \dots, x_n^2) + e_1(x_1^2, \dots, x_n^2)\end{aligned}$$

and

$$\begin{aligned}e_n(x_1^4 + x_1^2, \dots, x_n^4 + x_n^2) &= \prod_{i=1}^n (x_i^4 + x_i^2) = \left( \prod_{i=1}^n (x_i^2 + x_i) \right)^2 \\ &= e_n^2(x_1^2 + x_1, \dots, x_n^2 + x_n).\end{aligned}$$

By consequence, from  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n$  we can compute  $\tilde{S}_i$  for  $i = 1, \dots, n$  in the following way

$$\begin{aligned}\tilde{S}_1 &= \tilde{s}_1 + \tilde{s}_1^2 \\ \tilde{S}_i &= \tilde{S}_i \text{ for } i = 2, \dots, n-1 \\ \tilde{S}_n &= \tilde{E}_n^2\end{aligned}$$

Since  $S_1, \dots, S_n$  are the elementary symmetric polynomials evaluated in  $x_1^4 + x_1^2, \dots, x_n^4 + x_n^2$ , we can find  $\tilde{x}_1^4 + \tilde{x}_1^2, \dots, \tilde{x}_n^4 + \tilde{x}_n^2$  by factorizing a univariate polynomial. Indeed, it is well-known that

$$f(X) = X^n + \tilde{S}_1 X^{n-1} + \dots + \tilde{S}_{n-1} X + \tilde{S}_n = \prod_{i=1}^n (X + (\tilde{x}_i^4 + \tilde{x}_i^2)).$$

In the same way we have

$$f_{n-k}(X) = X^{n-k} + \tilde{S}_{1,n-k}X^{n-k-1} + \cdots + \tilde{S}_{n-k} = \prod_{\ell=1}^{n-k} (X + (\tilde{x}_{i_\ell}^4 + \tilde{x}_{i_\ell}^2))$$

and

$$f_k(X) = X^k + \tilde{S}_{1,k}X^{k-1} + \cdots + \tilde{S}_{k-1}X + \tilde{S}_k = \prod_{\ell=1}^k (X + (\tilde{x}_{j_\ell}^4 + \tilde{x}_{j_\ell}^2)) .$$

Thus, we have  $f(X) = f_{n-k}(X)f_k(X) \in \mathbb{K}[X]$ . Moreover, since we look for any solution of the system (7.9), we can choose any partition of  $\{1, \dots, n\}$  in two sets of size  $n-k$  and  $k$  with  $k \geq 2$ . Hence, from  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n$ , we can find  $\tilde{S}_{i,n-k}$  for  $i = 1, \dots, n-k$  and  $\tilde{S}_{j,k}$  for  $j = 1, \dots, k$  by factorizing the polynomial  $f$  in two polynomials one of degree  $k$  and the other of degree  $n-k$ , in some extension of  $\mathbb{K}$ .

**Lemma 7.21.** *Given,  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n \in \mathbb{K}$  we can compute  $\tilde{S}_{1,n-k}, \dots, \tilde{S}_{n-k,n-k} \in \mathbb{L}$  and  $\tilde{S}_{1,k}, \dots, \tilde{S}_{k,k} \in \mathbb{L}$  by factorizing the univariate polynomial*

$$\begin{aligned} f(X) &= X^n + (\tilde{s}_1^2 + \tilde{s}_1)X^{n-1} + \sum_{i=2}^{n-1} \tilde{S}_i X^{n-i} + \tilde{E}_n^2 \\ &= \left( X^k + \sum_{i=1}^k \tilde{S}_{i,k} X^{k-i} \right) \left( X^{n-k} + \sum_{i=1}^{n-k} \tilde{S}_{i,n-k} X^{n-k-i} \right) \\ &= f_k(X) f_{n-k}(X) \end{aligned}$$

with  $f \in \mathbb{K}[X]$  and  $f_k, f_{n-k} \in \mathbb{L}[X]$  where  $\mathbb{L}$  is an extension of  $\mathbb{K}$ .

Then, the value of  $\tilde{E}_{n-k,n-k}$  and  $\tilde{E}_{k,k}$  are respectively given by the square roots of  $\tilde{S}_{n-k,n-k}$  and  $\tilde{S}_{k,k}$ . Note that for any finite field  $\mathbb{K}$  of characteristic two, any element of  $\mathbb{K}$  has a square root in  $\mathbb{K}$ . Finally, to find  $\tilde{s}_{1,n-k}$  and  $\tilde{s}_{1,k}$  we have to factorize the two polynomials

$$X^2 + X + \tilde{S}_{1,n-k} \tag{7.10}$$

$$X^2 + X + \tilde{S}_{1,k} . \tag{7.11}$$

Note that the two equations (7.10) and (7.11) have respectively two solutions. Hence,  $\tilde{s}_{1,n-k}$  and  $\tilde{s}_{1,k}$  are respectively given by a solution of the equation (7.10) and the equation (7.11) verifying  $\tilde{s}_1 = \tilde{s}_{1,n-k} + \tilde{s}_{1,k}$ .

Finally, the evaluation of  $S_n^t$  is given by the corresponding univariate resultant as in Algorithm 21.

**Remark 7.22.** *To evaluate the two polynomials  $S_{k+1}^p$  and  $S_{n-k+1}^p$  at their corresponding evaluation points, we can proceed in two ways. We can either compute the two polynomials  $S_{k+1}^p$  and  $S_{n-k+1}^p$  and evaluate them. Or we can use the summation polynomials totally expressed w.r.t. the change of variables  $\Omega_{n,n}$ . Indeed, for any  $n \geq 3$  we have*

$$S_n^p(s_{1,n-1}, S_{2,n-1}, \dots, S_{n-2,n-1}, E_{n-1,n-1}, X) = S_n^t(s_1, S_2, \dots, S_{n-1}, E_n) \tag{7.12}$$

where  $s_1, S_2, \dots, S_{n-1}, E_n$  are given by equation (7.9) with  $k = 1$ ,  $s_{1,1} = X^2$  and  $E_{1,1} = X^2 + X$ .

---

**Algorithm 22:** Computing  $\tilde{E}_{k,k}, \tilde{E}_{n-k,n-k}, \tilde{s}_{1,k}, \tilde{s}_{1,n-k}$ .

---

**Input** :  $\tilde{S}_{k,k}, \tilde{S}_{n-k,n-k}, \tilde{S}_{1,k}, \tilde{S}_{1,n-k} \in \mathbb{L}$  and  $\tilde{s}_1 \in \mathbb{K}$  with  $\mathbb{L}$  an extension of  $\mathbb{K}$ .  
**Output**:  $\tilde{E}_{k,k}, \tilde{E}_{n-k,n-k}, \tilde{s}_{1,k}, \tilde{s}_{1,n-k} \in \mathbb{L}_2$  with  $\mathbb{L}_2$  is an extension of  $\mathbb{L}$  of degree at most 2.

```

1  $\tilde{E}_{k,k} := \sqrt{\tilde{S}_{k,k}}; \tilde{E}_{n-k,n-k} := \sqrt{\tilde{S}_{n-k,n-k}};$ 
2 if  $\text{Trace}_{\mathbb{L}}(\tilde{S}_{1,k}) = 0$  and  $\text{Trace}_{\mathbb{L}}(\tilde{S}_{1,n-k}) = 0$  then
3    $\tilde{s}_{1,k} :=$  a root of  $X^2 + X + \tilde{S}_{1,k}$  in  $\mathbb{L}$ ;
4    $\tilde{s}_{1,n-k} :=$  a root of  $X^2 + X + \tilde{S}_{1,n-k}$  in  $\mathbb{L}$ ;
5    $\mathbb{L}_2 := \mathbb{L}$ ;
6 else if  $\text{Trace}_{\mathbb{L}}(\tilde{S}_{1,k}) = 0$  then
7    $\tilde{s}_{1,k} :=$  a root of  $X^2 + X + \tilde{S}_{1,k}$  in  $\mathbb{L}$ ;
8    $\mathbb{L}_2 := \mathbb{L}[w]/(w^2 + w + \tilde{S}_{1,n-k}); \tilde{s}_{1,n-k} := w;$ 
9 else if  $\text{Trace}_{\mathbb{L}}(\tilde{S}_{1,n-k}) = 0$  then
10   $\tilde{s}_{1,n-k} :=$  a root of  $X^2 + X + \tilde{S}_{1,n-k}$  in  $\mathbb{L}$ ;
11   $\mathbb{L}_2 := \mathbb{L}[w]/(w^2 + w + \tilde{S}_{1,k}); \tilde{s}_{1,k} := w;$ 
12 else
13   $\mathbb{L}_2 := \mathbb{L}[w]/(w^2 + w + \tilde{S}_{1,k}); \tilde{s}_{1,k} := w;$ 
14   $\tilde{s}_{1,n-k} :=$  a root of  $X^2 + X + \tilde{S}_{1,n-k}$  in  $\mathbb{L}_2$ ;
15 if  $\tilde{s}_1 + \tilde{s}_{1,k} + \tilde{s}_{1,n-k} \neq 0$  then  $\tilde{s}_{1,n-k} := \tilde{s}_{1,n-k} + 1;$ 
16 return  $\tilde{E}_{k,k}, \tilde{E}_{n-k,n-k}, \tilde{s}_{1,k}, \tilde{s}_{1,n-k};$ 

```

---

In Table 7.4, we give for different values of  $n$  and  $k$ , the running time to evaluate  $S_n^t$  at 1000 random evaluation points chosen in  $\mathbb{K}^n$ . The column labeled by “ $f = f_k f_{n-k}$ ” contains the time to factorize the polynomial  $f$  in two polynomials of degree  $k$  and  $n - k$ . The column “ $\tilde{s}_{1,n-k}, \tilde{s}_{1,k}$ ” gives the time to solve the two equations (7.11) and (7.10). The column “ $S^p/S^t$ ” gives the times to evaluate  $S_{k+1}^p$  and  $S_{n-k+1}^p$  at their corresponding evaluation points by using either the partially symmetrized polynomials  $S_{k+1}^p$  and  $S_{n-k+1}^p$  or the totally symmetrized polynomials  $S_{k+1}^t$  and  $S_{n-k+1}^t$ , see Remark 7.22. Finally, the column labeled “Resultant” gives the time to compute the univariate resultant.

In Table 7.4, we can observe that the evaluation of  $S_7^p$  or  $S_7^t$  is very costly. Hence, to obtain an efficient evaluation function we force  $k + 1$  and  $n - k + 1$  to be less than seven. In that case, we can note that the most time-consuming step in the evaluation of  $S_n^t$  is the factorization of the univariate polynomial  $f$  in two polynomials of degree  $k$  and  $n - k$ . Moreover, we observe that the use of  $S_i^t$  to evaluate  $S_i^p$  is interesting when  $i$  is sufficiently large *i.e.*  $i \geq 5$ .

In order to decrease the time required for the factorization, we do not fix the value of  $k$ . Indeed, for each evaluation point we choose  $k$  such that it minimizes the degree of the extension of  $\mathbb{K}$  in which  $f$  can be factorized as the product of  $f_k f_{n-k}$ . We give the corresponding timings in Table 7.5. In this table, for  $n = 8$  the line labeled  $k \neq 2$  means we force  $n - k + 1 < 7$ . We summarize this algorithm to evaluate summation polynomials in Algorithm 23.

We can observe in Table 7.5 that not fixing  $k$  allows to decrease the overall time for the factorization and the total time to evaluate  $S_n^t$ . For instance, for  $n = 8$  with a fixed  $k$ , the best running time for 1000 evaluations is 49.100 seconds compared to 42.785 seconds when  $k$

| $\mathbb{K}$          | $n$ | $k$ | Evaluation time (seconds) |                                      |                                 |           |                 |
|-----------------------|-----|-----|---------------------------|--------------------------------------|---------------------------------|-----------|-----------------|
|                       |     |     | $f = f_k f_{n-k}$         | $\tilde{s}_{1,n-k}, \tilde{s}_{1,k}$ | $S^p/S^t$                       | Resultant | Total           |
| $\mathbb{F}_{2^{16}}$ | 5   | 2   | <b>2.800</b>              | 1.318                                | 0.095/ <b>0.118</b>             | 0.026     | 4.284/4.308     |
|                       |     | 3   | <b>3.904</b>              | 1.182                                | <b>0.286</b> /0.204             | 0.086     | 5.490/5.408     |
|                       | 6   | 2   | <b>4.498</b>              | 2.428                                | 0.154/ <b>0.166</b>             | 0.086     | 7.198/7.210     |
|                       |     | 3   | <b>6.286</b>              | 1.798                                | <b>3.536</b> /1.472             | 0.124     | 11.790/9.736    |
|                       | 7   | 2   | <b>7.368</b>              | 3.172                                | <b>0.400</b> /0.304             | 0.116     | 11.216/11.030   |
|                       |     | 3   | 9.460                     | 2.200                                | <b>315.980</b> / <b>131.460</b> | 0.460     | 328.280/143.760 |
|                       |     | 4   | <b>10.670</b>             | 4.620                                | <b>4.420</b> /2.030             | 0.190     | 19.990/17.600   |
|                       | 8   | 4   | <b>10.210</b>             | 2.740                                | <b>0.690</b> /0.490             | 0.310     | 14.040/13.840   |
| $\mathbb{F}_{2^{32}}$ | 5   | 2   | <b>9.940</b>              | 3.530                                | <b>0.180</b> /0.100             | 0.080     | 13.820/13.740   |
|                       |     | 3   | <b>14.830</b>             | 3.030                                | 0.350/ <b>0.380</b>             | 0.190     | 18.510/18.540   |
|                       | 6   | 2   | <b>17.360</b>             | 6.850                                | <b>0.260</b> /0.250             | 0.150     | 24.710/24.700   |
|                       |     | 3   | <b>19.620</b>             | 3.940                                | <b>3.650</b> /1.840             | 0.160     | 27.510/25.700   |
|                       | 7   | 2   | <b>28.210</b>             | 7.210                                | <b>0.960</b> /0.500             | 0.260     | 36.770/36.310   |
|                       |     | 3   | 33.020                    | 4.700                                | <b>360.470</b> / <b>210.250</b> | 0.760     | 399.150/248.930 |
|                       | 8   | 2   | <b>35.790</b>             | 9.830                                | <b>6.270</b> /2.900             | 0.470     | 52.470/49.100   |
|                       |     | 4   | <b>44.660</b>             | 7.930                                | <b>1.060</b> /0.770             | 0.590     | 54.300/54.010   |

Table 7.4: Running time to evaluate  $S_n^t$  at 1000 random evaluation points in  $\mathbb{K}^n$  with MAGMA (v2-19.4) on one core of a 2.00GHz Intel<sup>®</sup> E7540 CPU.

| $\mathbb{K}$          | $n$        | Evaluation time (seconds) |                                      |                                |           |        |
|-----------------------|------------|---------------------------|--------------------------------------|--------------------------------|-----------|--------|
|                       |            | $f = f_k f_{n-k}$         | $\tilde{s}_{1,n-k}, \tilde{s}_{1,k}$ | Eval. $S_{k+1}^p, S_{n-k+1}^p$ | Resultant | Total  |
| $\mathbb{F}_{2^{16}}$ | 5          | <b>3.065</b>              | 1.370                                | 0.130                          | 0.055     | 4.685  |
|                       | 6          | <b>3.995</b>              | 2.210                                | 0.225                          | 0.060     | 6.550  |
|                       | 7          | <b>4.470</b>              | 1.785                                | 0.370                          | 0.075     | 6.740  |
|                       | 8          | <b>5.290</b>              | 0.920                                | 7.315                          | 0.155     | 13.700 |
|                       | $k \neq 2$ | <b>7.020</b>              | 1.390                                | 0.525                          | 0.140     | 9.110  |
| $\mathbb{F}_{2^{32}}$ | 5          | <b>9.665</b>              | 3.275                                | 0.135                          | 0.085     | 13.240 |
|                       | 6          | <b>13.380</b>             | 6.290                                | 0.320                          | 0.125     | 20.190 |
|                       | 7          | <b>17.835</b>             | 4.820                                | 1.010                          | 0.200     | 23.920 |
|                       | 8          | <b>26.725</b>             | 3.535                                | 18.065                         | 0.335     | 48.750 |
|                       | $k \neq 2$ | <b>35.790</b>             | 5.265                                | 1.220                          | 0.440     | 42.785 |

Table 7.5: Running time to evaluate  $S_n^t$  at 1000 random evaluation points in  $\mathbb{K}^n$  with MAGMA (v2-19.4) on one core of a 2.00GHz Intel<sup>®</sup> E7540 CPU. The parameter  $k$  is chosen to minimize the degree of the extension of  $\mathbb{K}$  required for the factorization.

is not fixed and  $k+1, n-k+1 < 7$ . Nevertheless, the factorization is still the most expensive step in the evaluation of  $S_n^t$ . In the next section, we present how saving factorizations when interpolating summation polynomials.

### 7.4.2 Sparing factorizations

The most time consuming step in the evaluation of  $S_n^t$  is to find the corresponding evaluations points of  $S_{k+1}^p$  and  $S_{n-k+1}^p$ . That is to say to find a solution of the system (7.9).

---

**Algorithm 23:** Evaluating summation polynomials by factorization and resultant of univariate polynomials.

---

**Input** : An evaluation point  $(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \in \mathbb{K}^n$  and the summation polynomials partially and totally symmetrized  $S_i^p$  and  $S_i^t$  for  $i \in \{3, \dots, n-1\}$ .

**Output:**  $S_n^t(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \in \mathbb{K}$  or *fail*.

- 1  $f := X^n + (\tilde{s}_1^2 + \tilde{s}_1)X^{n-1} + \sum_{i=2}^{n-1} \tilde{S}_i X^{n-i} + \tilde{E}_n^2$ ;  $\mathbb{L} := \mathbb{K}$ ;
- 2  $f_1, \dots, f_r :=$  Factorization of  $f$  in  $\mathbb{K}$ ;
- 3  $d_1, \dots, d_r :=$  Degree of  $f_1, \dots, f_r$  with  $d_1 \leq \dots \leq d_r$ ;
- 4 **if** there not exists  $k \in \{2, \dots, \frac{n-2}{2}\}$  s.t. there exist  $j_1, \dots, j_s \in \{1, \dots, r\}$  all pairwise distinct s.t.  $k = \sum_{i=1}^s d_{j_i}$  **then**
- 5      $i := \min_{j=1, \dots, r} \{j \mid d_j \neq 1\}$ ;
- 6      $\mathbb{L} := \mathbb{K}[\alpha]/f_i(\alpha)$  extension of  $\mathbb{K}$  of degree  $d_i$ ;
- 7      $f_1, \dots, f_r :=$  Factorization of  $f$  in  $\mathbb{L}$ ;
- 8      $d_1, \dots, d_r :=$  Degree of  $f_1, \dots, f_r$  with  $d_1 \leq \dots \leq d_r$ ;
- 9 Let  $k \in \{2, \dots, \frac{n-2}{2}\}$  s.t. there exist  $j_1, \dots, j_s \in \{1, \dots, r\}$  all pairwise distinct s.t.  $k = \sum_{i=1}^s d_{j_i}$ ;
- 10  $F_k := \prod_{i=1}^s f_{j_i}$ ;  $F_{n-k} := f/F_k$ ;
- 11 **for**  $i := 1$  to  $k$  **do**  $\tilde{S}_{i,k} :=$  Coefficient of  $X^{k-i}$  in  $F_k$ ;
- 12 **for**  $i := 1$  to  $n-k$  **do**  $\tilde{S}_{i,n-k} :=$  Coefficient of  $X^{n-k-i}$  in  $F_{n-k}$ ;
- 13 Computing  $\tilde{E}_{k,k}, \tilde{E}_{n-k,n-k}, \tilde{s}_{1,k}, \tilde{s}_{1,n-k}$  using Algorithm 22;
- 14 **if**  $k+1 < 5$  **then**  $f_1 := S_{k+1}^p(\tilde{s}_{1,k}, \tilde{S}_{2,k}, \dots, \tilde{S}_{k-1,k}, \tilde{E}_{k,k}, X) \in \mathbb{L}_2[X]$ ;
- 15 **else**  $f_1 := S_{k+1}^t(\tilde{s}_{1,k} + X^2, \tilde{S}_{2,k} + (X^4 + X^2)(\tilde{s}_{1,k}^2 + \tilde{s}_{1,k}), \dots, \tilde{E}_{k,k}(X^2 + X))$ ;
- 16 **if**  $n-k+1 < 5$  **then**
- 17      $f_2 := S_{n-k+1}^p(\tilde{s}_{1,n-k}, \tilde{S}_{2,n-k}, \dots, \tilde{S}_{n-k-1,n-k}, \tilde{E}_{n-k,n-k}, X) \in \mathbb{L}_2[X]$ ;
- 18 **else**  $f_2 := S_{n-k+1}^t(\tilde{s}_{1,n-k} + X^2, \dots, \tilde{E}_{n-k,n-k}(X^2 + X))$ ;
- 19 **if**  $\deg(f_1) = \deg_X(S_{k+1}^p)$  and  $\deg(f_2) = \deg_X(S_{n-k+1}^p)$  **then**
- 20     **return** Resultant( $f_1, f_2$ );
- 21 **else return fail**;

---

Assume  $\tilde{s}_{1,k}, \tilde{S}_{2,k}, \dots, \tilde{S}_{k-1,k}, \tilde{E}_{k,k}, \tilde{s}_{1,n-k}, \tilde{S}_{2,n-k}, \dots, \tilde{S}_{n-k-1,n-k}, \tilde{E}_{n-k,n-k}$  is a solution of the system (7.9). Then, we can easily construct a solution of this system for the evaluation point  $(\tilde{s}_1 + 1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n)$ . Indeed,  $\tilde{s}_1 = \tilde{s}_{1,k} + \tilde{s}_{1,n-k}$  implies that  $\tilde{s}_1 + 1 = (\tilde{s}_{1,k} + 1) + \tilde{s}_{1,n-k}$ . Moreover, the  $n-1$  last equations of the system (7.9) either do not depend on  $s_{1,k}$  or depend on  $s_{1,k}$  by  $\alpha_2 = s_{1,k} + s_{1,k}^2$ . Since  $\alpha_2$  is invariant when adding 1 to  $s_{1,k}$  we obtain that  $\tilde{s}_{1,k} + 1, \tilde{S}_{2,k}, \dots, \tilde{S}_{k-1,k}, \tilde{E}_{k,k}, \tilde{s}_{1,n-k}, \tilde{S}_{2,n-k}, \dots, \tilde{S}_{n-k-1,n-k}, \tilde{E}_{n-k,n-k}$  is a solution of the system (7.9) when  $\tilde{s}_1$  is replaced by  $\tilde{s}_1 + 1$ .

Therefore, to speed up the evaluation step in the interpolation of the variable  $s_1$  of  $S_n^t$  instead of choosing  $d_1 = \deg_1(S_n^t)$  random  $\tilde{s}_1^{(i)}$  for  $i = 1, \dots, d_1$  and computing the polynomials  $S_n^t(\tilde{s}_1^{(i)}, S_2, \dots, S_{n-1}, E_n)$  for  $i = 1, \dots, d_1$  as described in Section 7.3, we choose only  $\left\lceil \frac{d_1}{2} \right\rceil$  random  $\tilde{s}_1^{(i)}$  for  $i = 1, \dots, \left\lceil \frac{d_1}{2} \right\rceil$ . Then, for  $i = 1, \dots, \left\lceil \frac{d_1}{2} \right\rceil$  by using the same evaluation

of the  $n - 1$  last variables we compute the two polynomials  $S_n^t(\tilde{s}_1^{(i)}, S_2, \dots, S_{n-1}, E_n)$  and  $S_n^t(\tilde{s}_1^{(i)} + 1, S_2, \dots, S_{n-1}, E_n)$ . By consequence, we divide by a factor two the number of factorizations of univariate polynomials required to interpolate the variable  $s_1$ .

To take full advantage of this trick the variable  $s_1$  is the last interpolated variable. Indeed, the number of evaluations required to interpolate a variable depends on the number of terms interpolated until now, which increases as the number of interpolated variables.

Except an efficient evaluation function, to be efficient Zippel's algorithm needs a sharp bound on the degree in each variables of the interpolated polynomial. The next section tackle this issue for  $S_n^t$ .

### 7.4.3 Degree of summation polynomials

#### Bound on the total degree of $S_n^t$

In Section 5.4 Theorem 5.10, it is shown that the  $n$ th summation polynomial has degree  $2^{n-2}$  in each variable. Hence, by construction the  $n$ th summation polynomial expressed in terms of the elementary symmetric polynomial  $e_1(\mathbf{x}), \dots, e_n(\mathbf{x})$  where  $\mathbf{x} = (x_1, \dots, x_n)$  are of total degree at most  $2^{n-2}$ . Since the polynomials  $e_1(\mathbf{x}_2), e_2(\mathbf{y}_2), \dots, e_{n-1}(\mathbf{y}_2), e_n(\mathbf{y})$  where  $\mathbf{x}_2 = (x_1^2, \dots, x_n^2)$ ,  $\mathbf{y}_2 = (x_1^2 + x_1^4, \dots, x_n^2 + x_n^4)$  and  $\mathbf{y} = (x_1 + x_1^2, \dots, x_n + x_n^2)$  are symmetric they can thus be expressed in terms of  $e_1(\mathbf{x}), \dots, e_n(\mathbf{x})$ . That is to say, there exist  $n$  polynomials  $\rho_1, \dots, \rho_n$  such that

$$\begin{cases} \rho_1(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) &= e_1(\mathbf{x}_2) \\ \rho_2(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) &= e_2(\mathbf{y}_2) \\ &\vdots \\ \rho_{n-1}(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) &= e_{n-1}(\mathbf{y}_2) \\ \rho_n(e_1(\mathbf{x}), \dots, e_n(\mathbf{x})) &= e_n(\mathbf{y}) \end{cases} \quad (7.13)$$

Clearly,  $\rho_1$  and  $\rho_n$  are of degree 2 and  $\rho_2, \dots, \rho_{n-1}$  are of degree 4. We do not have a formal proof that  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are algebraically independent. However, we have checked this algebraic independence by using MAGMA for many values of  $n$ . Hence, we follows Hypothesis 7.23.

**Hypothesis 7.23.** *For any  $n \geq 3$ , the polynomials  $\rho_1^{(h)}, \dots, \rho_n^{(h)}$  are algebraically independent.*

By using the result of Lemma 3.6 we can thus bound the degree of  $S_n^t$ .

**Lemma 7.24.** *Under Hypothesis 7.23, for all  $n \geq 3$  we have  $\deg(S_n^t) \leq 2^{n-3}$ .*

*Proof.* Let us write  $S_n^t = \sum_{\alpha \in \mathbb{N}^n} c_\alpha s_1^{\alpha_1} S_1^{\alpha_2} \dots S_{n-1}^{\alpha_{n-1}} E_n^{\alpha_n}$ . From Lemma 3.6 for the weights system  $(2, 4, \dots, 4, 2)$  we have  $\text{wdeg}(S_n^t) = \max \{ \sum_{i=1}^n \alpha_i w_i \mid c_\alpha \neq 0 \} = \deg(F_n) \leq 2^{n-2}$  where  $F_n$  is the  $n$ th summation polynomial expressed in terms of  $e_1(x), \dots, e_n(x)$ . Since the  $w_i$ 's are all divisible by 2, the degree of  $S_n^t$  which is given by  $\max \{ \sum_{i=1}^n \alpha_i \mid c_\alpha \neq 0 \}$  is at least divided by 2 in comparison to its weighted degree. Consequently, we have  $\deg(S_n^t) \leq 2^{n-3}$ .  $\square$

#### Exact degree in each variable of $S_n^t$

From Lemma 7.24,  $S_n^t$  has degree at most  $2^{n-3}$  in each variable. We can thus use Zippel's algorithm to interpolate it. In order to get a more efficient interpolation of  $S_n^t$ , from this bound on the degree of  $S_n^t$ , we look for the exact degree in each variables of  $S_n^t$ . To this

aim, we proceed as follows: we choose an initial evaluation point  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n$  and we interpolate the  $n$  following univariate polynomials:

$$\begin{aligned} F_1 &= S_n^t(s_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \\ F_i &= S_n^t(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{i-1}, S_i, \tilde{S}_{i+1}, \dots, \tilde{S}_{n-1}, \tilde{E}_n) \text{ for } i = 2, \dots, n-1 \\ F_n &= S_n^t(\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_{n-1}, E_n) \end{aligned}$$

With good probability, we thus have  $\deg(F_i) = \deg_i(S_n^t)$ . In Table 7.6, we give the degree in each variables of the summation polynomials obtained in this way.

| $n$ | Degree in each variable      |
|-----|------------------------------|
| 6   | (8,2,4,2,4,5)                |
| 7   | (16,5,8,4,8,5,16)            |
| 8   | (32,10,16,8,16,10,16,24)     |
| 9   | (64,24,32,20,32,20,32,24,64) |

Table 7.6: Degree in each variables of the summation polynomials.

At each step the number of evaluations required to interpolate the current variable depends on the number of terms already interpolated. Thus, our strategy is to interpolate the variables by increasing order of the  $d_i$ 's in order to try to minimize the growth of the number of terms interpolated at each step. For instance for  $n = 8$  we interpolate the variables in the following order:  $S_4, S_2, S_6, S_3, S_5, S_7, E_8, s_1$ . Note that we no longer use bounds on the  $d_i$ 's but the exact values of the  $d_i$ 's.

We can note that for  $n \leq 9$  the degree of  $S_n^t$  in any variables is less than or equal to the degree of  $S_n^t$  in the variable  $s_1$ . By consequence, interpolating at last the variable  $s_1$  (as mentioned in Section 7.4.2) is consistent with our strategy of interpolating the variables in increasing order of the  $d_i$ 's.

In Section 7.2, the compact representation of summation polynomials in characteristic two allows us to compute until the 7th summation polynomial by using the usual method. Using sparse multivariate interpolation algorithm of Zippel and the evaluation of  $S_8^t$  as just presented, we are now able to compute the 8th summation polynomial of a given curve *i.e.* the parameter  $t$  of the curve is fixed in some finite field  $\mathbb{F}_{2^k}$ . We now present some details about our implementation and timings.

#### 7.4.4 Computation of the eighth summation polynomial

##### Computing $S_8^t$ without knowing its support

In Table 7.7 we give detailed timings about the computation of the eighth summation polynomial expressed w.r.t.  $\Omega_{n,n}$  of a curve defined over  $\mathbb{F}_{2^{32}}$  by using sparse multivariate polynomial interpolation. The computations have been done with MAGMA. At each step *i.e.* for each interpolated variable, the evaluation step has been parallelized on eight cores. It is the only part of Zippel's algorithm which has been parallelized. Each step of Zippel's algorithm corresponds to a line of Table 7.7. For each step we give the current variable that we want to interpolate, the degree of  $S_8^t$  in this variable and the number of evaluations required to interpolate the current variable. We recall that this number is given by  $d_i t_{i-1}$  where  $t_{i-1}$  is the number of terms interpolated at the previous step (given in the last column of Table 7.7). We also detail

the time of each step to interpolate the current variable. More precisely, we give the time to perform all the required evaluations (the total CPU time and the wall-clock-time using eight cores); the time to solve the  $d_i$  Vandermonde systems and the time to reconstruct the  $t_{i-1}$  coefficients of the polynomial interpolated until now as univariate polynomial in the current variable. The “Data gathering” column in Table 7.7 gives the time to collect the evaluations of  $S_8^t$  which has been saved in files.

As expected, one can note in Table 7.7 that the most time consuming step in the computation of  $S_8^t$  is the evaluation step. Indeed, the evaluations step requires approximately 39 CPU hours of the 40.5 CPU hours required to compute  $S_8^t$ . Note that by using eight cores the eighth summation polynomial can be computed in approximately 22,500 seconds that is to say 6.25 hours.

One can notice that for the 5,085,889 evaluations of  $S_8^t$  the total CPU time required is of 141,000 CPU seconds. That is to say approximately 27.7 CPU seconds for 1000 evaluations. Which is less than the 42.785 CPU seconds expected from Table 7.5. This is a consequence of interpolating at least the variable  $s_1$ . Indeed, in that case the total number of evaluations is dominated by the number of evaluations required to interpolate the variable  $s_1$ . Moreover, as previously mentioned, to interpolate  $s_1$  we can save one half of the factorizations of univariate polynomials which is the most costly part of the evaluation of  $S_8^t$ . Thus it is natural to expect an average time for the evaluation step divided by a factor slightly less than two in comparison to the same number of evaluations at random points.

The obtained eighth summation polynomial consists of 470,369 terms that is to say its density is about 0.79%. In view of the number of terms in  $S_8^t$  it seems very difficult to compute it with the usual method involving bigger polynomials (see Section 7.2).

### Computing $S_8^t$ knowing its support

Once we have computed the eighth summation polynomial for a given curve, we know the support of the summation polynomial and to compute it for another curve, we just have to interpolate its coefficients. That is to say we need to evaluate  $S_8^t$  at exactly 470,369 evaluation points chosen in the following way: we randomly choose  $\tilde{s}_1, \tilde{S}_2, \dots, \tilde{S}_7, \tilde{E}_8$  such that the monomials in the support of  $S_8^t$  evaluated in this point are all distinct. Then we evaluate  $S_8^t$  at the evaluation points  $\tilde{s}_1^i, \tilde{S}_2^i, \dots, \tilde{S}_7^i, \tilde{E}_8^i$  for  $i = 0, \dots, t - 1$  with  $t = 470,369$ . Finally, to recover the coefficients of the eighth summation polynomial we have to solve a transpose Vandermonde system of size  $t \times t$ .

The evaluation of  $S_8^t$  at the 470,369 evaluation points can be done in 15,900 CPU seconds using MAGMA (v2-19.4). This can be easily parallelized and by using eight cores we can compute it in 2,160 seconds. Solving the transpose Vandermonde system can be done in 706 seconds CPU seconds (including time to data gathering and to reconstruct  $S_8^t$ ). Hence, computing  $S_8^t$  once its support is known can be done in 16,600 CPU seconds and 2,870 seconds by using eight cores for the evaluation.

### 7.4.5 Discussion about the computation of the ninth summation polynomial

Since, we are able to compute the eighth summation polynomial a natural issue would be to compute the ninth summation polynomial. Before throwing oneself headlong into this

| Interpolated<br>variables | $\deg_i(S_8^t)$ | Number of<br>evaluations | Time           |                       |                        |                   |               |                  |                       | $t_i$   |
|---------------------------|-----------------|--------------------------|----------------|-----------------------|------------------------|-------------------|---------------|------------------|-----------------------|---------|
|                           |                 |                          | Evaluation     |                       | Vandermonde<br>systems | Data<br>gathering | Interpolation | Total            |                       |         |
|                           |                 |                          | CPU<br>time    | WCT<br>eight<br>cores |                        |                   |               | CPU<br>time      | WCT<br>eight<br>cores |         |
| $S_4$                     | 8               | 9                        | 1.16s          | 0.28s                 | 0s                     | 0s                | 0.02s         | 1.18s            | 0.3s                  | 9       |
| $S_2$                     | 10              | 90                       | 2.95s          | 0.48s                 | 0.04s                  | 0.01s             | 0.03s         | 3.03s            | 0.56s                 | 63      |
| $S_6$                     | 10              | 630                      | 30.7s          | 5.79s                 | 0.09s                  | 0.06s             | 0.04s         | 30.9s            | 5.98s                 | 291     |
| $S_3$                     | 16              | 4,656                    | 180s = 3min    | 24s                   | 1.01s                  | 0.55s             | 0.31s         | 182s ~ 3min      | 25.8s                 | 1,468   |
| $S_5$                     | 16              | 23,488                   | 939s ~ 16min   | 122s ~ 2min           | 4.91s                  | 2.71s             | 1.93s         | 949 ~ 16min      | 131s ~ 2min           | 5,981   |
| $S_7$                     | 16              | 95,696                   | 3,750s ~ 1h    | 481s ~ 8min           | 30.3s                  | 12.7s             | 7.42s         | 3,800s ~ 1h      | 532s ~ 9min           | 26,711  |
| $E_8$                     | 24              | 641,064                  | 26,700s ~ 7.5h | 3,380s ~ 1h           | 281s ~ 5min            | ~1.5min           | ~1min         | 27,200s ~ 7.5h   | 3,810s ~ 1h           | 135,008 |
| $s_1$                     | 32              | 4,320,256                | 109,000s ~ 30h | 13,700s ~ 4h          | 3,020s~ 50min          | ~11min            | ~9min         | 113,000s ~ 31.5h | 17,900s ~ 5h          | 470,369 |
| Total                     |                 | 5,085,889                | ~39h           | ~5h                   | ~1h                    | ~13min            | ~10min        | ~40.5h           | ~6.2h                 |         |

Table 7.7: Running time to compute  $S_8^t$  of an elliptic curve defined over  $\mathbb{F}_{2^{32}}$  with MAGMA (v2-19.4) on a 2.00GHz Intel<sup>®</sup> E7540 CPU by using Zippel's multivariate polynomial interpolation algorithm.

computation, we give an estimation of the CPU time required to compute  $S_9^t$  by using Zippel's algorithm.

First, we need an estimation of the total number of terms contained in  $S_9^t$ . Assume that the density of  $S_9^t$  is divided by two in comparison of that of  $S_8^t$  (which is the ratio between the density of  $S_6^t$  and that of  $S_7^t$  see Table 7.1 and approximately also the ratio between the density of  $S_7^t$  and that of  $S_8^t$ ). The density of  $S_9^t$  would be about 0.40%. Thus,  $S_9^t$  would consist of 342,358,598 terms.

For  $n = 8$  we already need a field of size at least  $2^{32}$  to ensure that Zippel's algorithm give the right result. Indeed, from our experiments we observe that the bound on the probability of success of Zippel's algorithm,  $1 - \frac{n^2 dt}{q}$ , seems to be sufficient to ensure a correct result and that the algorithm does not return *fail*. Hence, for  $n = 8$  by choosing  $\mathbb{K} = \mathbb{F}_{2^{32}}$  the corresponding probability bound is 0.78. Therefore, for  $n = 9$  we cannot choose a smaller field. Moreover, if we use the probability bound of Zippel's algorithm to ensure a failure probability less than 0.5 for  $n = 9$ , we need to choose a field of size at least  $2^{42}$ .

The CPU time to compute the evaluations of  $S_9^t$  at 1000 points randomly chosen in  $(\mathbb{F}_{2^{32}})^9$  (respectively  $(\mathbb{F}_{2^{42}})^9$ ) is about 72 seconds (respectively 630 seconds). Since the number of evaluations required to compute  $S_9^t$  is lower bounded by the number of terms, if we assume that we can divide the total time for the evaluation step by two (by interpolating the variable  $s_1$  at least) then the total CPU time for the evaluation part of Zippel's algorithm to compute  $S_9^t$  would be lower bounded by 4.8 months (respectively 3.5 years) by using the field  $\mathbb{F}_{2^{32}}$  (respectively  $\mathbb{F}_{2^{42}}$ ).

We do not pretend that these estimations are tight but they show that the ninth summation polynomial seems very difficult to handle.

## 7.5 Application to the Discrete Logarithm Problem

In this section, we present the impact of such symmetries to the point decomposition problem solving. We want to compare the point decomposition problem solving using these symmetries and using only the action of the symmetric group as presented in [Gau09] see Chapter 5.

### 7.5.1 Using symmetries to speed up the PDP solving in characteristic two

Under Hypothesis 7.23 (which has been checked for various  $n$ ), we can use results of Chapter 3 to estimate the impact of such symmetries on the PDP solving.

As mentioned in Chapter 5 and Chapter 6 in practice we observe that polynomial systems obtained from a Weil descent from  $\mathbb{F}_{q^n}$  to  $(\mathbb{F}_q)^n$  over the  $(n + 1)$ th summation polynomial evaluated in the abscissa of the point we want to decompose and expressed in terms of the elementary symmetric polynomials are regular. By consequence, from Corollary 3.25, using the symmetries presented in Section 7.1.1 of binary summation polynomials allows to divide the complexity of the *Point Decomposition Problem* by a factor of  $(\prod_{i=1}^n \deg(\rho_i))^\omega = 2^{2\omega(n-1)}$ .

**Theorem 7.25.** *Let  $E$  be a binary elliptic curve defined over  $\mathbb{K} = \mathbb{F}_{2^{nk}}$  by*

$$E : y^2 + xy = x^3 + \alpha \quad (7.14)$$

*where  $\alpha \in \mathbb{K}$ . Under Hypothesis 5.18 and 7.23, the arithmetic complexity of the Point Decomposition Problem is bounded by*

- (proven complexity)  $\tilde{O}(n2^{3(n-1)(n-2)})$ ;
- (heuristic complexity)  $\tilde{O}(ne^{\omega n}2^{\omega(n-1)(n-2)})$ .

*Proof.* In Section 5.2.4 we saw that binary elliptic curves defined by equation (7.14) can be put in universal Edwards model. Hence, we can compute the summation polynomial for the curve  $E$  that admits the symmetries presented in Section 7.1.1. From Chapter 5 Section 5.6.1 the complexity of solving the PDP by using the action of the symmetric group is bounded by  $\tilde{O}(ne^{\omega n}2^{\omega n(n-1)} + n2^{3n(n-1)})$  arithmetic operations where the first part of the complexity corresponds to the  $F_5$  step and the second corresponds to the change of ordering step. The polynomials  $\rho_1, \dots, \rho_n$  in equation (7.13) give the change of variables between the system  $\mathcal{S}_1$  in  $\mathbb{K}[e_1, \dots, e_n]$  using the action of the symmetric group and the system  $\mathcal{S}_2$  in  $\mathbb{K}[s_1, S_2, \dots, S_{n-1}, E_n]$  using symmetries presented in Section 7.1.1 equation (7.4). Hence, from Theorem 3.14 by equipping the ring  $\mathbb{K}[s_1, S_2, \dots, S_{n-1}, E_n]$  of the weights  $(2, 4, \dots, 4, 2)$  corresponding to the degrees of the  $\rho_i$ 's; the complexity of solving  $\mathcal{S}_2$  in comparison to that of solving  $\mathcal{S}_1$  is divided by  $2^{2\omega(n-1)}$  (respectively  $2^{6(n-1)}$ ) for the  $F_5$  step (respectively change of ordering step). By consequence, we obtain the proven complexity.

The heuristic complexity is obtained by using the complexity of change of ordering for Gröbner bases in *Shape Position*, see Chapter 4. Indeed, we observe that the lexicographical Gröbner basis of  $\langle \mathcal{S}_2 \rangle$  is in *Shape Position*. In order to preserve the quasi-homogeneous structure, we do not apply the randomization strategy (with low probability of success since  $p = 2$ ). However, we observe that very few normal forms are required to compute the multiplication matrix by the smallest variable. Hence, its computation is still negligible in the total solving process.  $\square$

**Remark 7.26.** For  $\mathcal{S}_1$  we observe that its LEX Gröbner basis is a triangular set. In that case, we need to compute all the multiplication matrices. However, since the degree of the equations depends exponentially on  $n$ , results of Theorem 4.11 does not apply.

## 7.5.2 Benchmarks on the PDP solving

We conclude this section by giving experimental results showing the impact of the symmetries of binary curves on the PDP solving.

In Table 7.8 we present timings for solving the PDP for  $n = 4, 5$  by using the action of the symmetric group as presented in [Gau09] corresponding to lines labelled by “ $\mathfrak{S}_n$  [Gau09]” and by using the action of the 2-torsion as presented in Section 7.1.1 corresponding to line labelled “This work”. We give the time to compute the (W)DRL Gröbner basis using the implementation of  $F_4$  of MAGMA. We give also the real and theoretical maximal degree (column “ $d_{\max}/d_{\text{theo}}$ ”) reached by the polynomials during the computation of the (W)DRL Gröbner basis. The theoretical bound is obtained by using Corollary 2.76. The timings for the change of ordering using the implementation of FGLM of MAGMA is given in column “FGLM”. The column “# solutions” gives the degree of the ideal.

We can observe that as expected by the Bézout's bound for quasi-homogeneous system, the degree of the ideal  $\langle \mathcal{S}_2 \rangle$  is divided by  $2^{2(n-1)} = \prod_{i=1}^n \deg(\rho_i) = \prod_{i=1}^n w_i$  in comparison of the degree of  $\langle \mathcal{S}_1 \rangle$ ; where the  $(w_1, \dots, w_n)$  are the weights of which  $\mathbb{K}[s_1, S_2, \dots, S_{n-1}, E_n]$  is equipped. For  $n = 4$ , we observe that for  $k = 16$  (respectively  $k = 32$ ) our method is approximately 900 (respectively 5500) times faster than the initial method in [Gau09].

<sup>1</sup>Out of memory on barbecue at Nancy with 512GB of RAM.

| $n$ | $k$ |                          | $F_4$      | $d_{\max}/d_{\text{theo}}$ | FGLM                                                                                           | # solutions | Total                                                                               |
|-----|-----|--------------------------|------------|----------------------------|------------------------------------------------------------------------------------------------|-------------|-------------------------------------------------------------------------------------|
| 4   | 16  | $\mathfrak{S}_n$ [Gau09] | 0.540s     | 24/29                      | 44.290s                                                                                        | 4096        | 44.830s                                                                             |
|     |     | This work                | 0.040s     | 24/24                      | 0.010s                                                                                         | 64          | 0.050s                                                                              |
|     | 32  | $\mathfrak{S}_n$ [Gau09] | 1.410s     | 24/29                      | 660.890s                                                                                       | 4096        | 662.300s                                                                            |
|     |     | This work                | 0.070s     | 24/24                      | 0.050s                                                                                         | 64          | 0.120s                                                                              |
| 5   | 16  | $\mathfrak{S}_n$ [Gau09] | 17080.590s | 68/76                      |  <sup>1</sup> | 1048576     |  |
|     |     | This work                | 60.020s    | 68/68                      | 1193.140s                                                                                      | 4096        | 1253.160s                                                                           |
|     | 32  | This work                | 194.520s   | 68/68                      | 12430.540s                                                                                     | 4096        | 12625.060s                                                                          |

Table 7.8: CPU time to solve the PDP for  $n = 4, 5$  with MAGMA (v2-19.4) on one core of a 2.00GHz Intel<sup>®</sup> E7540 CPU.

For  $n = 5$  we are now able to solve the PDP using MAGMA in approximately 20 minutes for  $k = 16$  and 3.5 hours for  $k = 32$ . With the algorithm of Gaudry, we cannot perform the change of ordering as it requires a lot of memory. Note that with our method, for  $k = 16$  the time to compute the (W)DRL Gröbner basis is divided by a factor of about 280.

One can notice that the most time consuming step in the PDP solving is the change of ordering. Moreover, when using the symmetries presented in Section 7.1.1 the obtained lexicographical Gröbner basis is in *Shape Position* (which is not the case by using only the action of the symmetric group). Hence, we can use the change of ordering algorithm for *Shape Position* ideals presented in Section 2.3.2. In Table 7.9 we give the timings for solving the PDP by using symmetries presented in this chapter and a first implementation in MAGMA of the change of ordering algorithm for *Shape Position* ideals. Note that since the field is of characteristic two, the randomized strategy in Section 4.5 has a very low probability of success. In practice, we observe that with or without random linear change of variables there are very few normal forms to compute.

| $n$ | $k$ | $F_4$    | $d_{\max}/d_{\text{theo}}$ | Fast FGLM | # solutions | Total     |
|-----|-----|----------|----------------------------|-----------|-------------|-----------|
| 5   | 16  | 60.020s  | 68/68                      | 257.510s  | 4096        | 317.530s  |
|     | 32  | 194.520s | 68/68                      | 2680.190s | 4096        | 2874.710s |

Table 7.9: CPU time to solve the PDP for  $n = 5$  with MAGMA (v2-19.4) on one core of a 2.00GHz Intel<sup>®</sup> E7540 CPU by using change of ordering for *Shape Position* ideals.

Whether in theory or in practice using the action of the two torsion of elliptic curves allows to significantly improve the *Point Decomposition Problem* solving for binary elliptic curves. Indeed, the complexity is divided by an exponential factor,  $2^{2\omega(n-1)}$  where  $2 \leq \omega < 2.3727$  is the linear algebra constant. Moreover, for  $n = 5$  and a base field of 16 bits the PDP was intractable while one can now solve it in approximately 5 minutes using MAGMA. Note that by using FGb the timings are much faster and the timings with MAGMA could be improve with an efficient implementation of change of ordering algorithm for *Shape Position* ideals. For instance for  $k = 31$  and  $n = 5$  solving the PDP can be achieved in approximately 10 seconds with FGb on one core of a 1.70GHz Intel<sup>®</sup> i7-4650U CPU.



# List of Tables

|     |                                                                                                                                                                                                                                                                                 |     |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 2.1 | Complexity to solve a univariate polynomial of degree $d$ in number of operations in $\mathbb{K}$ . When $\mathbb{K} = \mathbb{Q}$ , $s$ denotes the size of the coefficients of the polynomial and the complexity is given in number of word operations. . . . .               | 61  |
| 4.1 | A worst case example: comparison of the usual algorithm for solving the PoSSo problem and Algorithm 16, the proposed algorithm. Computation with FGb on a 3.47 GHz Intel Xeon X5677 CPU. . . . .                                                                                | 100 |
| 6.1 | Computing time of Gröbner basis with MAGMA (V2-19.1) on one core of a 2.00 GHz Intel® E7540 CPU for $n = 4$ . The last column (number of operations) is based on FGb. . . . .                                                                                                   | 142 |
| 6.2 | Computing time of Gröbner basis with FGb on a 3.47 GHz Intel® X5677 CPU for $n = 5$ . . . . .                                                                                                                                                                                   | 143 |
| 6.3 | Computing time of Gröbner basis with MAGMA (V2-19.1) on one core of a 2.00 GHz Intel® E7540 CPU for $n = 5$ and decomposition in $n - 1$ points. Operation counts are obtained using FGb. . . . .                                                                               | 144 |
| 6.4 | Computing time of DRL Gröbner basis with FGb on a 3.47 GHz Intel® X5677 CPU for $n = 6$ and decomposition in $n - 1$ points. . . . .                                                                                                                                            | 144 |
| 6.5 | Number of operations needed to solve the ECDLP defined over $\mathbb{F}_{q^n}$ for $n = 4, 5, 6$ and $32 \leq \log_2(q) \leq 128$ . . . . .                                                                                                                                     | 147 |
| 6.6 | Domain parameters according to the security level given in number of boolean operations needed to solve the ECDLP. . . . .                                                                                                                                                      | 148 |
| 7.1 | Density of summation polynomials of binary elliptic curves expressed in terms of the polynomial change of variables $\Omega_{n,n}$ . . . . .                                                                                                                                    | 158 |
| 7.2 | CPU time to compute the $n$ th summation polynomial with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU by using resultant and elimination ideals. . . . .                                                                                                           | 160 |
| 7.3 | CPU time to compute the $n$ th summation polynomial with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU by using resultant and normal forms. . . . .                                                                                                                 | 160 |
| 7.4 | Running time to evaluate $S_n^t$ at 1000 random evaluation points in $\mathbb{K}^n$ with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU. . . . .                                                                                                                     | 169 |
| 7.5 | Running time to evaluate $S_n^t$ at 1000 random evaluation points in $\mathbb{K}^n$ with MAGMA (v2-19.4) on one core of a 2.00GHz Intel® E7540 CPU. The parameter $k$ is chosen to minimize the degree of the extension of $\mathbb{K}$ required for the factorization. . . . . | 169 |

|     |                                                                                                                                                                                                                                   |     |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 7.6 | Degree in each variables of the summation polynomials. . . . .                                                                                                                                                                    | 172 |
| 7.7 | Running time to compute $S_g^t$ of an elliptic curve defined over $\mathbb{F}_{2^{32}}$ with MAGMA (v2-19.4) on a 2.00GHz Intel <sup>®</sup> E7540 CPU by using Zippel's multivariate polynomial interpolation algorithm. . . . . | 174 |
| 7.8 | CPU time to solve the PDP for $n = 4, 5$ with MAGMA (v2-19.4) on one core of a 2.00GHz Intel <sup>®</sup> E7540 CPU. . . . .                                                                                                      | 177 |
| 7.9 | CPU time to solve the PDP for $n = 5$ with MAGMA (v2-19.4) on one core of a 2.00GHz Intel <sup>®</sup> E7540 CPU by using change of ordering for <i>Shape Position</i> ideals. . . . .                                            | 177 |

# List of Figures

|     |                                                                                                                                                                                                                                     |     |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 1.1 | Résolution de systèmes polynomiaux par bases de Gröbner. . . . .                                                                                                                                                                    | 6   |
| 1.2 | Exemples de courbes elliptiques définies sur les réels. . . . .                                                                                                                                                                     | 8   |
| 1.3 | Comparaison des complexités des deux étapes de la résolution du problème PoSSo par base de Gröbner. Le nombre de variables est fixé à $n = 20$ et le degré des équations $d$ tend vers l'infini. . . . .                            | 12  |
| 2.1 | Intersection of sections of the quotient ring $\mathcal{R} = \mathbb{K}[x_1, \dots, x_n]/\mathcal{I}$ by $x_1^{d_1}, \dots, x_{i-1}^{d_{i-1}}, x_{i+1}^{d_i}, \dots, x_{n-1}^{d_{n-2}}$ with $\mathcal{I}$ a generic ideal. . . . . | 32  |
| 2.2 | Steps of height one and generators of $\text{in}_{>\text{drl}}(\mathcal{I})_d$ . . . . .                                                                                                                                            | 35  |
| 2.3 | Macaulay matrix of $(f_1, \dots, f_s)$ in graduation $d$ w.r.t. $>$ . . . . .                                                                                                                                                       | 42  |
| 4.1 | Complexity of change of ordering and $F_5$ steps in the polynomial system solving process with $d \rightarrow \infty$ and $n = 20$ . . . . .                                                                                        | 81  |
| 4.2 | Shape of the matrix $M$ of Algorithm 15. . . . .                                                                                                                                                                                    | 91  |
| 5.1 | Group law of elliptic curves. . . . .                                                                                                                                                                                               | 105 |
| 5.2 | Jacobi intersection curve over the real numbers. . . . .                                                                                                                                                                            | 108 |
| 5.3 | Edwards curve over the real numbers. . . . .                                                                                                                                                                                        | 109 |
| 5.4 | Pollard $\rho$ method. . . . .                                                                                                                                                                                                      | 113 |



# List of Algorithms

|    |                                                                                                                                                     |     |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 1  | Computing normal forms by linear algebra. . . . .                                                                                                   | 31  |
| 2  | Applying a polynomial change of variables (1). . . . .                                                                                              | 38  |
| 3  | Applying a polynomial change of variables (2). . . . .                                                                                              | 38  |
| 4  | Computing Gröbner bases by linear algebra: Lazard's algorithm. . . . .                                                                              | 42  |
| 5  | Computing Gröbner bases by linear algebra: Matrix $F_5$ algorithm. . . . .                                                                          | 44  |
| 6  | Computing the multiplication matrices: the original algorithm. . . . .                                                                              | 46  |
| 7  | UPDATE( $T_1, \dots, T_n, t, \text{nf}$ ) . . . . .                                                                                                 | 47  |
| 8  | A change of ordering algorithm for Gröbner bases: FGLM. . . . .                                                                                     | 48  |
| 9  | Probabilistic change of ordering algorithm for <i>Shape Position</i> ideals. . . . .                                                                | 51  |
| 10 | Computing $h_n$ deterministically. . . . .                                                                                                          | 51  |
| 11 | Polynomial systems solving . . . . .                                                                                                                | 62  |
| 12 | Solving polynomial systems admitting polynomial change of variables. . . . .                                                                        | 70  |
| 13 | LEX Gröbner basis computation as a triangular set. . . . .                                                                                          | 85  |
| 14 | Probabilistic change of ordering for <i>Shape Position</i> ideals. . . . .                                                                          | 86  |
| 15 | Building multiplication matrices (in the following    does not mean parallel code<br>but gives details about pseudo code on the left side). . . . . | 88  |
| 16 | Another algorithm for PoSSo. . . . .                                                                                                                | 97  |
| 17 | Pohlig-Hellman reduction. . . . .                                                                                                                   | 112 |
| 18 | Baby step giant step for DLP. . . . .                                                                                                               | 112 |
| 19 | Index calculus attack for ECDLP. . . . .                                                                                                            | 121 |
| 20 | Sparse multivariate polynomial interpolation: Zippel's algorithm. . . . .                                                                           | 163 |
| 21 | Evaluating summation polynomials. . . . .                                                                                                           | 165 |
| 22 | Computing $\tilde{E}_{k,k}, \tilde{E}_{n-k,n-k}, \tilde{s}_{1,k}, \tilde{s}_{1,n-k}$ . . . . .                                                      | 168 |
| 23 | Evaluating summation polynomials by factorization and resultant of univariate<br>polynomials. . . . .                                               | 170 |



# Bibliography

- [AD94] Leonard M. Adleman and Jonathan DeMarrais. A subexponential algorithm for discrete logarithms over all finite fields. In *Advances in Cryptology—CRYPTO’93*, pages 147–158. Springer, 1994.
- [ADH94] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyper-elliptic curves over finite fields. In *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Comput. Sci.* Springer-Verlag, 1994. 6th International Symposium.
- [AH96] Leonard M. Adleman and Ming-Deh A. Huang. Counting rational points on curves and abelian varieties over finite fields. In Henri Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 1996.
- [Bar04] Magali Bardet. *Étude des Systèmes Algébriques Surdéterminés. Applications aux Codes Correcteurs et à la Cryptographie*. PhD thesis, Université Pierre et Marie Curie, 2004.
- [BBJ<sup>+</sup>08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Chistiane Peters. Twisted edwards curves. In *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, AFRICACRYPT’08, pages 389–405, Berlin, Heidelberg, 2008. Springer-Verlag.
- [BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J-SYMBOLIC-COMP*, 24(3–4):235–265, 1997.
- [BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, volume 3(issue 3):177–197, 2009.
- [BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. *Designs, Codes and Cryptography*, pages 1–52, 2012.
- [BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving - ICPSS*, pages 71–75, November 2004.

- [BFSS13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.
- [BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and BY Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In P. Gianni, editor, *The Effective Methods in Algebraic Geometry Conference, Mega 2005*, pages 1–14, May 2005.
- [BGJT13] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. <http://hal.inria.fr/hal-00835446>, 2013.
- [BGY80] Richard P Brent, Fred G Gustavson, and David YY Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1(3):259–295, 1980.
- [BH74] James Bunch and John Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.
- [BL] Daniel J. Bernstein and Tanja Lange. Explicit-Formulas Database. <http://www.hyperelliptic.org/EFD/>.
- [BL07] Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In *Advances in Cryptology : ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.
- [BLP93] J.P. Buhler, Hendrik W. Lenstra, and Carl Pomerance. Factoring integers with the number field sieve. In Arjen K. Lenstra and Hendrik W. Lenstra, editors, *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*, pages 50–94. Springer Berlin Heidelberg, 1993.
- [BM74] Allan Borodin and Robert Moenck. Fast modular transforms. *J. Comput. Syst. Sci.*, 8(3):366–386, June 1974.
- [BMMT94] Eberhard Becker, Teo Mora, Maria Grazia Marinari, and Carlo Traverso. The shape of the shape lemma. In *Proceedings of the international symposium on Symbolic and algebraic computation*, ISSAC '94, pages 129–133, New York, NY, USA, 1994. ACM.
- [BOT88] Michael Ben-Or and Prasoona Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 301–309, New York, NY, USA, 1988. ACM.
- [BPW06] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. A zero-dimensional Gröbner basis for AES-128. In *Fast Software Encryption*, pages 78–88. Springer, 2006.
- [BRSS12] Saugata Basu, Marie-Françoise Roy, Mohab Safey El Din, and Éric Schost. A baby step-giant step roadmap algorithm for general algebraic sets. *CoRR*, abs/1201.6439, 2012.

- [BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical computer science*, 22(3):317–330, 1983.
- [BS87a] David Bayer and Michael Stillman. A criterion for detecting m-regularity. *Inventiones mathematicae*, 87(1):1–11, 1987.
- [BS87b] David Bayer and Michael Stillman. A theorem on refining division orders by the reverse lexicographic order. *Duke Mathematical Journal*, 55(2):321–328, 1987.
- [BSS03] Alin Bostan, Bruno Salvy, and Éric Schost. Fast algorithms for zero-dimensional polynomial systems using duality. *Applicable Algebra in Engineering, Communication and Computing*, 14(4):239–272, 2003.
- [Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, Mathematical Institute, University of Innsbruck, 1965.
- [Buc06] Bruno Buchberger. Bruno Buchberger’s PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *J. Symb. Comput.*, 41(3-4):475–511, March 2006.
- [Can93] John F. Canny. Computing roadmaps of general semi-algebraic sets. *Comput. J.*, 36(5):504–514, 1993.
- [CC86] David V. Chudnovsky and Gregory V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, 7(4):385–434, 1986.
- [CCS11] Arjeh M. Cohen, Hans Cuypers, and Hans Sterk. *Some Tapas of Computer Algebra*. Algorithms and Computation in Mathematics Series. Springer, 2011.
- [CF05] Henri Cohen and Gerhard Frey, editors. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC Press, 2005.
- [Che55] Claude Chevalley. Invariants of finite groups generated by reflections. *American Journal of Mathematics*, 77(4):pp. 778–782, 1955.
- [CK91] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inf.*, 28(7):693–701, October 1991.
- [CL08] Jean-Marc Couveignes and Reynald Lercier. Galois invariant smoothness basis. *Series on Number Theory and Its Applications*, 5:142–167, May 2008. World Scientific.
- [CLO07] David Cox, John Little, and Donal O’Shea. *Ideals, Varieties, and Algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra*, volume 10. Springer, 2007.
- [Coh93] Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer, 1993.

- [Col97] Antoine Colin. *Théorie des Invariants Effective. Applications à la Théorie de Galois et à la Résolution de Systèmes Algébriques. Implantation en Axiom*. PhD thesis, Université Pierre et Marie Curie, 1997.
- [Cop93] Don Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, 1993.
- [Cou01] Jean-Marc Couveignes. Algebraic groups and discrete logarithm. In *Public-key cryptography and computational number theory*, pages 17–27, 2001.
- [CP05] Richard E Crandall and Carl Pomerance. *Prime numbers: a computational perspective*, volume 182. Springer, 2005.
- [Dat03] Ruchira S Datta. Universality of Nash equilibria. *Mathematics of Operations Research*, 28(3):424–432, 2003.
- [DBP11] Mario De Boer and Ruud Pellikaan. *Some Tapas of Computer Algebra*, chapter Gröbner Bases for Codes. In *Algorithms and Computation in Mathematics Series [CCS11]*, 2011.
- [DF12] Oumar Diao and Emmanuel Fouotsa. Edwards model of elliptic curves defined over any fields. Cryptology ePrint Archive, Report 2012/346, 2012. <http://eprint.iacr.org/>.
- [DGG<sup>+</sup>02] Jean-Guillaume Dumas, Thierry Gautier, Mark Giesbrecht, Pascal Giorgi, Bradford Hovinen, Erich Kaltofen, B. David Saunders, Will J. Turner, and Gilles Villard. LinBox: A generic library for exact linear algebra. In Arjeh M. Cohen, Xiao-Shan Gao, and Nobuki Takayama, editors, *ICMS'2002, Proceedings of the 2002 International Congress of Mathematical Software, Beijing, China*, pages 40–50. World Scientific Pub., August 2002.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September November 1976.
- [Die06] Claus Diem. An index calculus algorithm for plane curves of small degree. In *Algorithmic number theory ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 543–557. Springer, 2006.
- [Die11a] Claus Diem. On the discrete logarithm problem in class groups of curves. *Math. Comp*, 80:443–475, 2011.
- [Die11b] Claus Diem. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147:75–104, 2011.
- [DIK06] Christophe Doche, Thomas Icart, and David R Kohel. Efficient scalar multiplication by isogeny decompositions. In *Public Key Cryptography-PKC 2006*, pages 191–206. Springer, 2006.
- [DK02] Harm Derksen and Gregor Kemper. *Computational invariant theory*, volume 131. Springer, 2002.

- [DT08] Claus Diem and Emmanuel Thomé. Index calculus in class groups of non-hyperelliptic curves of genus three. *Journal of Cryptology*, 21(4):593–611, 2008.
- [Duq07] Sylvain Duquesne. Improving the arithmetic of elliptic curves in the Jacobi model. *Information Processing Letters*, 104(3):101–105, 2007.
- [Edw07] Harold M. Edwards. A normal form for elliptic curves. In *Bulletin of the American Mathematical Society*, volume 44, pages 393–422, July 2007.
- [EG02] Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith*, 102(1):83–103, 2002.
- [EG07] Andreas Enge and Pierrick Gaudry. An  $l(1/3 + \epsilon)$  algorithm for the discrete logarithm problem for low degree curves. In *Advances in Cryptology-EUROCRYPT 2007*, pages 379–393. Springer, 2007.
- [Eis95] David Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Springer, 1995.
- [Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, June 1999.
- [Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.
- [Fau10] Jean-Charles Faugère. FGb: A library for computing Gröbner bases. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.
- [FGHR12a] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaél Renault. Fast change of ordering with exponent  $\omega$ . *ACM Commun. Comput. Algebra*, 46:92–93, September 2012.
- [FGHR12b] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaél Renault. Using symmetries and fast change of ordering in the index calculus for elliptic curves discrete logarithm. In *SCC '12: Proceedings of the Third International Conference on Symbolic Computation and Cryptography*, pages 113–118, July 2012.
- [FGHR13a] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaél Renault. Polynomial systems solving by fast linear algebra, 2013. <http://arxiv.org/abs/1304.6039>.
- [FGHR13b] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaél Renault. Using symmetries in the index calculus for elliptic curves discrete logarithm. *Journal of Cryptology*, pages 1–41, 2013. doi 10.1007/s00145-013-9158-5.

- [FGLM93] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
- [Fid72] Charles M. Fiduccia. Polynomial evaluation via the division algorithm the fast fourier transform revisited. In *Proceedings of the fourth annual ACM symposium on Theory of computing*, STOC '72, pages 88–93, New York, NY, USA, 1972. ACM.
- [FJ03] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In Boneh Dan, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer Berlin / Heidelberg, 2003.
- [Flo67] Robert W. Floyd. Nondeterministic algorithms. *J. ACM*, 14(4):636–644, October 1967.
- [FLR11] Jean-Charles Faugère, David Lubicz, and Damien Robert. Computing modular correspondences for abelian varieties. *Journal Of Algebra*, 343(1):248–277, 2011.
- [FM11] Jean-Charles Faugère and Chenqi Mou. Fast algorithm for change of ordering of zero-dimensional Gröbner bases with sparse multiplication matrices. In *ISSAC '11: Proceedings of the 2011 international symposium on Symbolic and algebraic computation*, ISSAC '11, pages 1–8, New York, NY, USA, 2011. ACM.
- [FM13] Jean-Charles Faugère and Chenqi Mou. Sparse FGLM algorithms. <http://hal.inria.fr/hal-00807540>, 2013.
- [FNW10] Rongquan Feng, Menglong Nie, and Hongfeng Wu. Twisted jacobi intersections curves. *Theory and Applications of Models of Computation*, pages 199–210, 2010.
- [FP09] Jean-Charles Faugère and Ludovic Perret. An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography. *Journal of Symbolic Computation*, 44(12):1676–1689, 2009.
- [FPPR12] Jean-Charles Faugère, Ludovic Perret, Christophe Petit, and Guénaél Renault. Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 27–44. Springer Berlin / Heidelberg, 2012.
- [FR09] Jean-Charles Faugère and Sajjad Rahmany. Solving systems of polynomial equations with symmetries using SAGBI-Gröbner bases. In *ISSAC '09: Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, ISSAC '09, pages 151–158, New York, NY, USA, 2009. ACM.
- [Fre01] Gerhard Frey. Applications of arithmetical geometry to cryptographic constructions. In *International Conference on Finite Fields and Applications*, pages 128–161, 2001.
- [FS13] Jean-Charles Faugère and Jules Svartz. Gröbner bases of ideals invariant under a commutative group: the non-modular case. In *ISSAC*, pages 347–354, 2013.

- [FSS11] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity. *Journal Of Symbolic Computation*, 46(4):406–437, 2011.
- [FSS13] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. On the complexity of the Generalized MinRank Problem. *Journal of Symbolic Computation*, 55:30–58, 2013.
- [FSV13] Jean-Charles Faugère, Mohab Safey El Din, and Thibaut Verron. On the complexity of computing Gröbner bases for quasi-homogeneous systems. In *ISSAC*, pages 189–196, 2013.
- [Gal] André Galligo. Algorithmes de calcul de base standards. Université de Nice.
- [Gal73] André Galligo. *A Propos du Théorème de Préparation de Weierstrass*. PhD thesis, Institut de Mathématique et Sciences Physiques de l’Université de Nice, 1973.
- [Gau09] Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, 44(12):1690–1702, 2009.
- [GG99] Karin Gatermann and Frédéric Guyard. Gröbner bases, invariant theory and equivariant dynamics. *Journal of Symbolic Computation*, 28(1):275–302, 1999.
- [GGMZ13] Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. On the function field sieve and the impact of higher splitting probabilities: Application to discrete logarithms in  $\mathbb{F}_{2^{1971}}$  and  $\mathbb{F}_{2^{3164}}$ . Cryptology ePrint Archive, Report 2013/074, 2013. <http://eprint.iacr.org/>.
- [GGR03] Joachim von zur Gathen, Jaime Gutierrez, and Rosario Rubio. Multivariate polynomial decomposition. *Applicable Algebra in Engineering, Communication and Computing*, 14(1):11–31, 2003.
- [GHS02] Pierrick Gaudry, Florian Hess, and Nigel Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002.
- [GM89] Patrizia Gianni and Teo Mora. Algebraic solution of systems of polynomial equations using Gröbner bases. In *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, Proceedings of AAEECC-5, volume 356 of LNCS*, pages 247–257. Springer, 1989.
- [GS11] Aurélien Greuet and Mohab Safey El Din. Deciding reachability of the infimum of a multivariate polynomial. In *ISSAC 2011—Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, pages 131–138. ACM, New York, 2011.
- [GS12] Pierrick Gaudry and Éric Schost. Genus 2 point counting over prime fields. *Journal of Symbolic Computation*, 47(4):368–400, 2012.

- [GTTD07] Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. A double large prime variation for small genus hyperelliptic index calculus. *Mathematics of Computation*, 76:475–492, 2007.
- [Hes04] Florian Hess. Computing relations in divisor class groups of algebraic curves over finite fields. Preprint, 2004.
- [HI98] Ming-Deh Huang and Doug Ierardi. Counting points on curves over finite fields. *Journal of Symbolic Computation*, 25(1):1 – 21, 1998.
- [HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Berlin Heidelberg, 1998.
- [JM89] Edmund Jonckheere and Chingwo Ma. A simple Hankel interpretation of the Berlekamp-Massey algorithm. *Linear Algebra and its Applications*, 125:65–76, 1989.
- [Jou13a] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 177–193. Springer Berlin Heidelberg, 2013.
- [Jou13b] Antoine Joux. A new index calculus algorithm with complexity  $L(1/4 + o(1))$  in very small characteristic. Cryptology ePrint Archive, Report 2013/095, 2013. <http://eprint.iacr.org/>.
- [JV12] Antoine Joux and Vanessa Vitse. Cover and decomposition index calculus on elliptic curves made practical. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 9–26. Springer Berlin Heidelberg, 2012.
- [JV13] Antoine Joux and Vanessa Vitse. Elliptic curve discrete logarithm problem over small degree extension fields - application to the static Diffie-Hellman problem on  $E(\mathbb{F}_{q^5})$ . *J. Cryptology*, 26(1):119–143, 2013.
- [Kan01] Richard Kane. *Reflection Groups and Invariant Theory*. Springer, 2001.
- [KG85] Walter Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theor. Comput. Sci.*, 36:309–317, June 1985.
- [KL89] Erich Kaltofen and Yagati Lakshman. Improved sparse multivariate polynomial interpolation algorithms. In *Symbolic and Algebraic Computation*, pages 467–474. Springer, 1989.
- [Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.
- [Kob89] Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

- [Lak90] Yagati N. Lakshman. On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, STOC '90, pages 555–563, New York, NY, USA, 1990. ACM.
- [Laz83] Daniel Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In J. van Hulzen, editor, *Computer Algebra*, volume 162 of *Lecture Notes in Computer Science*, pages 146–156. Springer Berlin / Heidelberg, 1983.
- [Laz92] Daniel Lazard. Solving zero-dimensional algebraic systems. *Journal of symbolic computation*, 13(2):117–131, 1992.
- [Len87] Hendrik W Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.
- [LL91] Yagati N. Lakshman and Daniel Lazard. On the complexity of zero-dimensional algebraic systems. In *Effective methods in algebraic geometry*, volume 94, page 217. Birkhauser, 1991.
- [LY97] Philippe Loustau and Eric V York. On the decoding of cyclic codes using Gröbner bases. *Applicable Algebra in Engineering, Communication and Computing*, 8(6):469–483, 1997.
- [Mac94] Francis S. Macaulay. *The Algebraic Theory of Modular Systems*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1994. Revised reprint of the 1916 original, With an introduction by Paul Roberts.
- [Mas69] James Massey. Shift-register synthesis and bch decoding. *Information Theory, IEEE Transactions on*, 15(1):122–127, 1969.
- [MB72] Robert Moenck and Allan Borodin. Fast modular transforms via division. In *Switching and Automata Theory, 1972., IEEE Conference Record of 13th Annual Symposium on*, pages 90–96, 1972.
- [MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and Christoph G. Günther, editors, *Advances in Cryptology — EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer Berlin Heidelberg, 1988.
- [Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.
- [Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
- [Mou13] Chenqi Mou. *Solving Polynomial Systems over Finite Fields*. PhD thesis, Université Pierre et Marie Curie, 2013.

- [MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646, 1993.
- [MP98] Bernard Mourrain and Victor Y Pan. Asymptotic acceleration of solving multivariate polynomial systems of equations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 488–496. ACM, 1998.
- [MS91] Guillermo Moreno-Socias. *Autour de la fonction de Hilbert-Samuel (escaliers d’ideaux polynomiaux)*. PhD thesis, Ecole Polytechnique, 1991.
- [MS03] Guillermo Moreno-Socias. Degrevlex Gröbner bases of generic complete intersections. *Journal of Pure and Applied Algebra*, 180(3):263–283, 2003.
- [MVO91] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC ’91, pages 80–89, New York, NY, USA, 1991. ACM.
- [MVOV10] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC press, 2010.
- [Nag10] Koh-Ichi Nagao. Decomposed attack for the jacobian of a hyperelliptic curve over an extension field. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *Algorithmic Number Theory*, volume 6197 of *Lecture Notes in Comput. Sci.* Springer–Verlag, 2010. 9th International Symposium, Nancy, France, ANTS-IX, July 19-23, 2010, Proceedings.
- [Nat09] National Institute of Standards and Technology. Digital signature standard (dss). Technical Report FIPS PUB 186-3, U.S. Department of Commerce, June 2009.
- [Pan02] Victor Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *Journal of Symbolic Computation*, 33(5):701 – 733, 2002.
- [Par94] Keith Pardue. *Nonstandard Borel-Fixed Ideals*. PhD thesis, Brandeis University, 1994.
- [Pat95] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt’88. In Don Coppersmith, editor, *Advances in Cryptology — CRYPTO’95*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer Berlin Heidelberg, 1995.
- [Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT ’96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg, 1996.
- [PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over  $\mathbb{Z}/p\mathbb{Z}$ . *Information Theory, IEEE Transactions on*, 24(1):106–110, 1978.

- [Pil90] Jonathan Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Mathematics of Computation*, 55(192):745–763, 1990.
- [Pol78] John M. Pollard. Monte carlo methods for index computation mod  $p$ . *Math. Comp.*, 32(143):918–924, July 1978.
- [PQ12] Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a Weil descent. In *Asiacrypt 2012*, Lecture Notes in Computer Science (LNCS). Springer, 12 2012.
- [RSA78] Ron L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital dignatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.
- [Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.
- [Sch85] René Schoof. Elliptic curves over finite fields and the computation of square roots mod  $p$ . *Mathematics of Computation*, 44(170):pp. 483–494, 1985.
- [Sem98] Igor Semaev. Evaluation of discrete logarithms in a group of  $p$ -torsion points of an elliptic curve in characteristic  $p$ . *Mathematics of Computation of the American Mathematical Society*, 67(221):353–356, 1998.
- [Sem04] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004. <http://eprint.iacr.org/>.
- [Sha71] Daniel Shanks. Class number, a theory of gactorization, and genera. In *1969 Number Theory Institute (Proc. Sympos. Pure Math., Vol. XX, State Univ. New York, Stony Brook, N.Y., 1969)*, pages 415–440. Providence, R.I., 1971.
- [Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, pages 256–266. Springer-Verlag, 1997.
- [Sil09] Joseph H Silverman. *The Arithmetic of Elliptic Curves*, volume 106. Springer, 2009.
- [Sma99] Nigel P Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12(3):193–196, 1999.
- [Sma01] Nigel P Smart. The Hessian form of an elliptic curve. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 118–125. Springer, 2001.
- [Smi95] Larry Smith. *Polynomial Invariants of Finite Groups*. Research Notes in Mathematics, Vol 6. A.K Peters, Wellesley, Mass, 1995. second printing 1997.
- [Spa12] Pierre-Jean Spaenlehauer. *Solving multi-homogeneous and determinantal systems. Algorithms - Complexity - Applications*. PhD thesis, PhD thesis, Université Paris 6, 2012.

- [SS11] Mohab Safey El Din and Éric Schost. A baby steps/giant steps probabilistic algorithm for computing roadmaps in smooth bounded real hypersurface. *Discrete & Computational Geometry*, 45(1):181–220, 2011.
- [ST54] Geoffrey C. Shephard and John A. Todd. Finite unitary reflection groups. *Canadian J. Math.*, 6:274–304, 1954.
- [ST13] Michael Shantz and Edlyn Teske. Solving the elliptic curve discrete logarithm problem using Semaev polynomials, Weil descent and Gröbner basis methods – an experimental study. Cryptology ePrint Archive, Report 2013/596, 2013. <http://eprint.iacr.org/>.
- [Sta78] Richard P Stanley. Hilbert functions of graded algebras. *Advances in Mathematics*, 28(1):57–83, 1978.
- [Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.
- [Stu02] Bernd Sturmfels. *Solving Systems of Polynomial Equations*, volume 97. American Mathematical Society, 2002.
- [Stu08] Bernd Sturmfels. *Algorithms in Invariant Theory (Texts and Monographs in Symbolic Computation)*. Springer Publishing Company, Incorporated, 2nd ed.; vii, 197 pp.; 5 figs. edition, 2008.
- [Thé03] Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology : ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92, 2003.
- [VW12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th symposium on Theory of Computing*, pages 887–898. ACM, 2012.
- [VZG90a] Joachim Von Zur Gathen. Functional decomposition of polynomials: The tame case. *Journal of Symbolic Computation*, 9(3):281 – 299, 1990. Computational algebraic complexity editorial.
- [VZG90b] Joachim Von Zur Gathen. Functional decomposition of polynomials: The wild case. *Journal of Symbolic Computation*, 10(5):437 – 452, 1990.
- [VZGG03] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.
- [Wei49] André Weil. Numbers of solutions of equations in finite fields. *Bulletin of the American Mathathematical Society*, 55(5):497–508, 1949.
- [Wie86] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theor.*, 32(1):54–62, 1986.
- [YJSPT13] Huang Yun-Ju, Naoyuki Shinohara, Christophe Petit, and Tsuyoshi Takagi. Improvement of Faugère et al.’s method to solve ECDLP. In *IWSEC 2013*, 11 2013.

- 
- [Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In EdwardW. Ng, editor, *Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer Berlin Heidelberg, 1979.
- [Zip90] Richard Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375 – 403, 1990. Computational algebraic complexity editorial.

---

## Résumé

Depuis ces dix dernières années, les attaques sur le logarithme discret sur les courbes elliptiques (ECDLP) mettant en jeu la résolution de systèmes polynomiaux connaissent un large succès. C'est dans ce contexte que s'inscrit cette thèse dont les contributions sont doubles.

D'une part, nous présentons de nouveaux outils de résolution de systèmes polynomiaux par bases de Gröbner. Nous montrons que la résolution de systèmes avec symétries est étroitement liée à la résolution de systèmes quasi-homogènes. Nous proposons ainsi de nouveaux résultats de complexité pour la résolution de tels systèmes. Nous nous intéressons également à l'étape bloquante de la résolution de systèmes : le changement d'ordre pour bases de Gröbner. La complexité classique de cette étape est cubique en le nombre de solutions et domine la complexité totale de la résolution. Nous proposons pour la première fois des algorithmes de changement d'ordre de complexité sous-cubique en le nombre de solutions.

D'autre part, nous nous intéressons à l'attaque du logarithme discret sur les courbes elliptiques par calcul d'indice proposée par Gaudry. Nous mettons en évidence des familles de courbes elliptiques possédant des symétries particulières. Ces symétries impliquent un gain exponentiel sur la complexité de la résolution du ECDLP. Nous obtenons ainsi de nouveaux paramètres de sécurité pour certaines instances du ECDLP. Une des étapes principales de cette attaque nécessite le calcul de polynômes de sommation introduits par Semaev. Les symétries des courbes elliptiques binaires nous permettent d'élaborer un nouvel algorithme par évaluation-interpolation pour le calcul des polynômes de sommation. Munis de cet algorithme nous établissons un nouveau record pour le calcul de ces polynômes.

---

## Abstract

Since the last decade, attacks on the elliptic curve discrete logarithm problem (ECDLP) which requires to solve polynomial systems have been quite successful. This thesis takes place in this context and the contributions are twofold.

On the one hand, we present new tools for solving polynomial systems by using Gröbner bases. First, we investigate polynomial systems with symmetries. We show that solving such a system is closely related to solving quasi-homogeneous systems. We thus propose new complexity bounds for solving systems with symmetries. Then, we study the bottleneck of polynomial systems solving: the change of ordering for Gröbner bases. The usual complexity of such algorithms is cubic in the number of solutions and dominates the overall complexity of polynomial systems solving. We propose for the first time change of ordering algorithms with sub-cubic complexity in the number of solutions.

On the other hand, we investigate the index calculus attack of Gaudry to solve the elliptic curve discrete logarithm problem. We highlight some families of elliptic curves that admit particular symmetries. These symmetries imply an exponential gain in the complexity of solving the ECDLP. As a consequence, we obtain new security parameters for some instances of the ECDLP. One of the main steps of this attack requires to compute Semaev summation polynomials. The symmetries of binary elliptic curves allow us to propose a new algorithm based on evaluation-interpolation to compute their summation polynomials. Equipped with this algorithm we establish a new record for the computation of these polynomials.