UNIVERSITÀ DEGLI STUDI DI UDINE
HIC SUNT FUTURA

DEPARTMENT OF MATHEMATICS, COMPUTER SCIENCE AND PHYSICS

ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND CYBERSECURITY

MASTER THESIS IN
INTERNATIONAL MASTER DEGREE IN ARTIFICIAL INTELLIGENCE
CYBERSECURITY

Gröbner Basis Cryptanalysis and its Application to Arithmetization Oriented Symmetric Primitives

CANDIDATE
Luca Campa

SUPERVISOR
Prof. Marino Miculan

CO-SUPERVISOR
Prof. Arnab Roy

CO-SUPERVISOR
Prof. Elisabeth Oswald

Academic Year 2023-2024

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
`https://www.dmif.uniud.it/`

© 2024 Luca Campa
This work is shared under the Creative Commons 4.0 License Attribution-NonCommercial-ShareAlike.

Acknowledgements

It was a great pleasure to write this thesis after a fantastic period at the University of Innsbruck. Therefore, I would like to thank the Security and Privacy Lab for hosting me and, in particular, Prof. Arnab Roy for his guidance and priceless suggestions. Moreover, a big thank you to Prof. Marino Miculan who, with his guidance and passion for his work, represents an important example to follow and the reason why I decided to attend the Master Degree at the University of Udine. In addition, I cannot forget Sandro Campigotto, the professor (now a friend) who gave me the solid mathematical foundations that allowed me to increase my knowledge and passion in the field of cryptography.

Lastly, I am grateful to my family, in particular to my mother, father and grandmother, who are perhaps happier than me for this big achievement, for their encouragement and for having raised me with love and knowledge, and to my girlfriend Sara who, making my life better every day, always supported and sustained me over these years.

Finally, I would like to thank my friends and, in particular, “CasaUD” for making my journey light and fun.

Abstract

Newer cryptography designs are gaining more interest in the field of algebraic cryptanalysis due to their Arithmetization-Oriented (AO) nature. As a consequence, their main threat comes from the application of algebraic cryptanalysis techniques. One of them is the so-called Gröbner basis cryptanalysis which requires mathematical foundations from algebraic geometry. In this thesis we cover all the necessary theoretical concepts and their immediate application in the context of computational algebra. On top of this, we present the structure of a Gröbner basis cryptanalysis and an in-depth application on `Anemoi`, one of the recently proposed Arithmetization-Oriented primitives, based on our new polynomial modelling called ACICO. The new model permits to precisely define the complexity of the Gröbner basis computation step for `Anemoi`, and to show that the security of `Anemoi` should not rely on it, no matter the number of branches. Afterwards, we present a complexity analysis with respect to the problem of computing the variety (the solutions) of the system by using different approaches and, in particular, by exploiting the `Wiedemann algorithm` which permits to challenge the security of full-round instances. Moreover, the refined Gröbner basis cryptanalysis methodology and the considerations on the modelling choices can be extended to other AO primitives.

Contents

List of Tables

List of Figures

1 Introduction

In algebraic geometry, solving a system of simultaneous equations is a well-known problem which gained further interests in the recent years due to its application to the security analysis of modern ciphers. Cryptanalysis of ciphers based on "solving a system of equations" in a number of unknowns was already considered by Claude E. Shannon [34]. However, because this system solving strategies did not pose any security threat to the security of modern block ciphers like AES [9, 10, 42], the cryptanalytic impact of such techniques remained unexplored, until recently. Newer cryptography designs, such as MPC (Multiparty Computation), FHE (Fully Homomorphic Encryption) and ZK-friendly (Zero-knowledge) primitives, are gaining more interest in the field of algebraic cryptanalysis due to their Arithmetization-Oriented (AO) nature. One of the main techniques that is still under exploration is the Gröbner basis (GB) cryptanalysis which represents one of the main security threats.

The main aim of Gröbner basis cryptanalysis is to solve a system of polynomials for obtaining the values of a set of target variables. For example, in case of a hash function the target variables can be the input to the hash function. Thus, obtaining the value of this target variable means finding a pre-image of the hash function. The common *modus operandi* requires to model the cryptographic function (e.g. permutation) as a system of polynomials, say $\mathcal{F}$, over a suitable finite field $\mathbb{K}$. Next, a Gröbner basis G w.r.t a suitable monomial ordering is computed for the ideal $I = \langle \mathcal{F} \rangle$, generated by $\mathcal{F}$. Finally, one can solve the polynomial system defining the Gröbner basis G to obtain the values of the target variables. However, even in this last step, several methods can be applied depending on the inner structure and on the properties of the considered system. Computing a Gröbner basis particularly facilitates the process of solving the system of polynomials in it. In particular, one can employ basis conversion techniques to obtain a favourable set of polynomials that is easy to solve, or eigenvalue methods to obtain univariate polynomial (possibly for each target variables).

An important outcome that emerges from the GB analysis (in both previous and our results) of the AO primitives is that their security against GB analysis is not well-defined and, in particular, it is not always determined by the high computational complexity of the Gröbner basis computation step. AO primitives like MiMC, GMiMC, Poseidon etc. do not provide useful bound on the complexity of the GB computation step. Hence, their security parameters should rely upon other steps of the process.

Giving the exact complexity of Gröbner basis computation is a non-trivial problem. Previous works rely on row reduction complexity bound of Macaulay matrices (and it is a function of the *solving degree* which is the highest degree reached during the GB computation). It is well known that the bound is loose, and often the experimental complexity shows or indicates better bound. Thus, as said before, from both cryptanalytic and design perspectives it does not represent a good choice for defining the security of the analysed primitive. M. Steiner [38] tried to tighten the GB theoretical complexity by mathematically proving the solving degree bound for iterated polynomial systems and, in particular, for the attacks on MiMC, Feistel-MiMC, Feistel-MiMC-Hash, Hades and GMiMC, showing the relation between the *fall degree* and the *Castelnuovo-Mumford regularity*. Moreover, as shown by [1], the designers of symmetric-key primitives should pay particular attention to the algebraic structure of their cipher and its components, due to the fact that there can be a clever representation of those components which can lead to experimental complexities far better that the theoretical estimates. An example of how the algebraic structure of a cipher can be exploited is given again in [1], where the authors show how the modelling choices matter e.g. the choice of the variables and the field to operate with. In this thesis we address this issues by providing a new model for `Anemoi` called A_{CICO}, thanks to which we compute the exact complexity of the GB computation w.r.t. the Buchberger's algorithm. As a result, a threat model should not consider the GB computation as a metric for defining the security parameters of a primitive.

One of the main objectives in GB cryptanalysis is to obtain a "nice" set of polynomials (from the Gröbner basis G of I) that is convenient to solve. This step has a significant complexity and an important role in GB cryptanalysis. Moreover, differently from the GB computation, it is based on well-defined complexity bounds that can represent a good metric for the definition of the cipher's security parameters.

After computing the Gröbner basis G, one may choose to employ basis conversion techniques like FGLM to obtain a GB G', typically w.r.t lexicographic monomial ordering. The only requirement is that I is a zero-dimensional ideal. The main advantage of this basis conversion is that, if I meets some conditions, G' consists of one univariate polynomial, say $g(x)$, and other polynomials which are of the form $y_i - f_i(x)$ for each variable y_i. This means that the solution of the variables y_i can be obtained by substituting the solutions of $g(x) = 0$. Usually, obtaining such form, which is commonly referred to as shape form, requires the condition that the ideal $I = \langle \mathcal{F} \rangle$ is a radical ideal. Alternatively, if the radicality condition is not met, one has to show that the GB G' of I w.r.t lexicographic monomial ordering consists of polynomials in such nice form. Checking that condition for proving the shape form for the Gröbner basis of I is often omitted or concluded from experimental results in GB cryptanalysis of AO primitives.

When it is not possible to confirm the conditions for applying basis conversion techniques like FGLM one can resort to *eigenvalue* method. The main objective of this technique is to obtain a univariate polynomial corresponding to each target variable and find at least one solution to the polynomial (if it is not irreducible). An additional advantage of this technique is that it can sometimes be more efficient than the basis conversion technique. Indeed, we will show that using probabilistic version of this eigenvalue method is more efficient than the technique used in [3, 29].

In this thesis we will discuss what are the possible paths to follow, and what are the main advantages gained by each one in the context of Gröbner basis cryptanalysis.

1.1 Outline

Now, we provide the outline of the thesis. In particular, the thesis is organized in 4 chapters: Mathematical foundations, Computational algebra, Gröbner basis cryptanalysis and Application to `Anemoi`. Figure 1.1 depicts a schematic overview of the sections and advises the reader about the mandatory sections and dependencies that are necessary to understand the main objective of this thesis: the application of GB cryptanalysis to `Anemoi`.

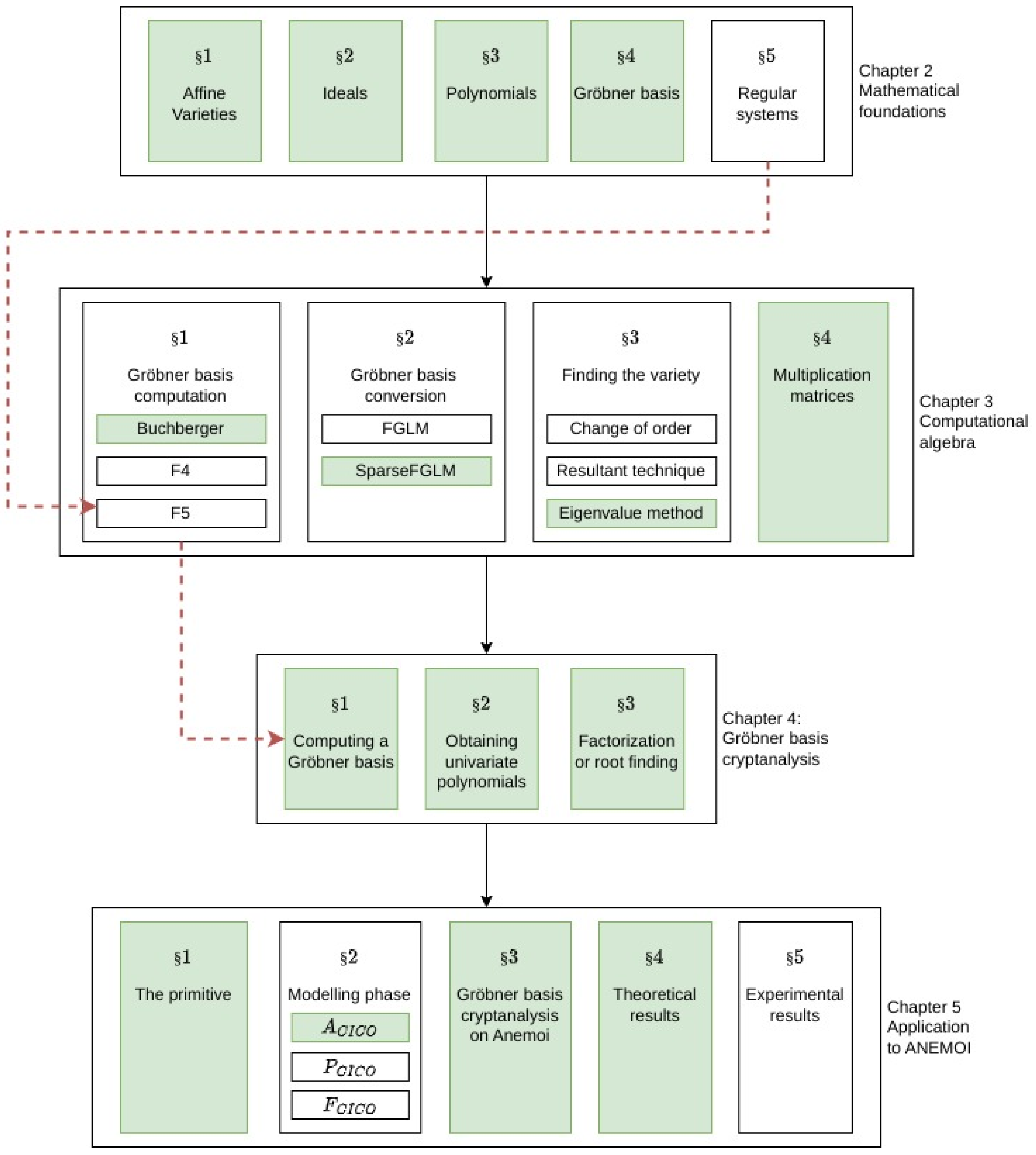

§1
Affine Varieties
§2
Ideals
§3
Polynomials
§4
Gröbner basis
§5
Regular systems
Chapter 2 Mathematical foundations
§1
Gröbner basis computation
Buchberger
F4
F5
§2
Gröbner basis conversion
FGLM
SparseFGLM
§3
Finding the variety
Change of order
Resultant technique
Eigenvalue method
§4
Multiplication matrices
Chapter 3 Computational algebra
§1
Computing a Gröbner basis
§2
Obtaining univariate polynomials
§3
Factorization or root finding
Chapter 4: Gröbner basis cryptanalysis
§1
The primitive
§2
Modelling phase
$\mathcal{A}_{\text{CICO}}$
$\mathcal{P}_{\text{CICO}}$
$\mathcal{F}_{\text{CICO}}$
§3
Gröbner basis cryptanalysis on Anemoi
§4
Theoretical results
§5
Experimental results
Chapter 5 Application to ANEMOI

Figure 1.1: Outline of the thesis. § denotes the sections. The green boxes highlight the mandatory sections. The red dashed line denotes that the starting point is required to understand the end point.

Chapter 2 deals with mathematical fundamentals. We provide an in-depth explanation of the tools and concepts needed to understand the reasons under the application of certain algorithms. A quick example is the notion of *ideal* we used above.

Chapter 3 brings the reader to the direct application of the mathematical concepts introduced in Chapter 2. For the first two steps of the GB cryptanalysis methodology, we present the possible algorithms, their complexities and the main advantages. Furthermore, sometimes we require some additional theoretical concepts which will be introduced just before the presentation of the corresponding algorithm.

Chapter 4 takes all the ingredients discussed in Chapter 3 and tries to outline a walkthrough of the possible combinations of those ingredients in order to make GB cryptanalysis applicable and easily reproducible by the reader.

Finally, Chapter 5 presents the application of the methodology to one of the most recent AO primitive proposals: `Anemoi`. Firstly, we describe the primitive and its components, then we present our GB analysis and its results. Recall one of the main steps in GB cryptanalysis: modelling the primitive as a set of polynomials. Choosing a "good" model is necessary to reduce the complexity of the subsequent procedures. Here, we present a new modelling choice for `Anemoi`, which we called A_{CICO}. As you will discover, this model makes the GB computation step "negligible" w.r.t the other steps, confirming that making the security of the primitive relying upon the GB computation does not represent a good design choice. Secondly, we show how to improve the existing attack complexities by adopting the Wiedemann algorithm. Finally, we demonstrate our attack strategies with some experimental results, showing that we are able not only to attack more rounds with respect to previous results, but also to apply the attack to more than 2 `Anemoi` branches.

At the end of the thesis we give the reader more information about the algebraic cryptanalysis methodologies and one of the most recent applications against `Anemoi`. Appendix A presents one of the most recent attacks against `Anemoi` with 2 branches. Moreover, we show that the methodology used by [3] (forcing `Anemoi` with 2 branches to be a FreeLunch system) is not applicable to the multi-branches case. Appendix B briefly discuss two more algebraic cryptanalysis methodologies: *interpolation attacks* and *Higher-order differential attacks.*

2

Mathematical foundations

In this chapter we present the basic mathematical knowledge required to follow the inner working of the algebraic cryptanalysis section. We'll start from the basic definitions to more advance theorems and applications.

We use $\mathbb{K}$ to denote a field and $R := \mathbb{K}[x_1, \dots, x_n]$ to denote the polynomial ring over $\mathbb{K}$ in the variables $x_1, \dots, x_n$. When $\mathbb{K} = \mathbb{F}_p$ the ring is denoted as R_p. Due to ease of notation, at times we use the notation $\mathbb{K}[\mathbf{x}]$ to denote the ring where the variables x_i are made clear in the corresponding context. Most of the time $\mathbb{K} = \mathbb{F}_p$ where $p > 2$ is possibly a large prime, unless stated otherwise. We use $LM_k(p)$ to denote the k-th leading monomial of the polynomial p with respect to a certain monomial order which will be clear from the context. In addition, when we refer to the first leading monomial, we simply use $LM(p)$. Sometimes we also use $LC(p) \in \mathbb{K}$ to denote the leading coefficient of the polynomial p and $LT(p)$ to denote the leading term of p where $LT(p) = LC(p) \cdot LM(p)$. Finally, $|\mathcal{B}|$ will denote the cardinality of the set $\mathcal{B}$. Throughout the thesis, with abuse of notation, we will use $\mathsf{deg}(x)$ to denote both the degree of the variable x, both the degree of the polynomial x. The correct interpretation will be clear from the context. The following definitions are mainly taken from [15, 16].

2.1 Affine Varieties

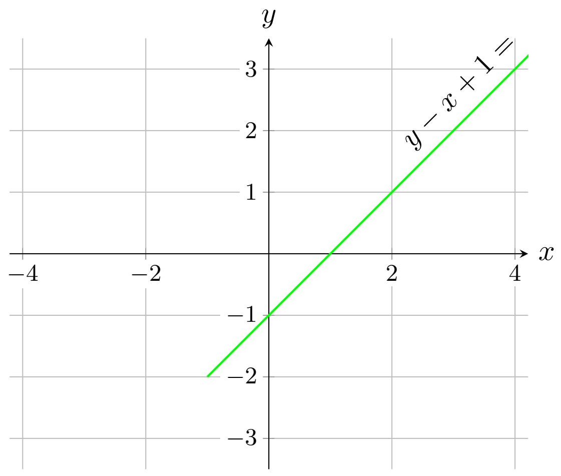
Definition 1 (Affine Variety). *Let $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$. Then, the set*

$$\mathbb{V}(f_1, \dots, f_s) = \{(a_1, \dots, a_n) \in \mathbb{K}^n \mid f_i(a_1, \dots, a_n) = 0, 1 \leq i \leq s\}$$

is called the ***affine variety*** *defined by the polynomials $f_1, \dots, f_s$.*

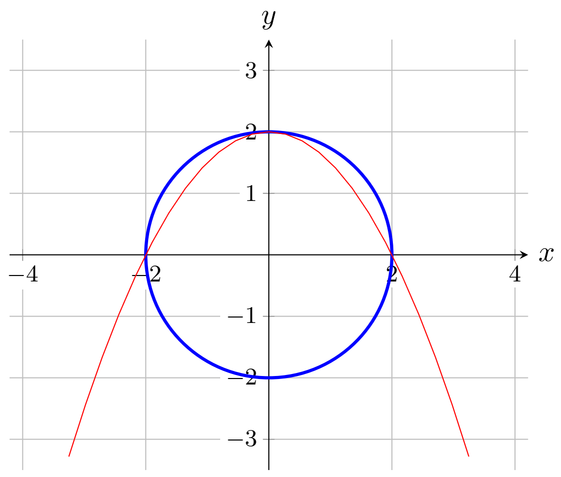
From a geometrical point of view the affine variety is the set of points (in the n-th dimensional affine space) such that the polynomials f_i are evaluated to 0. As a simple example, let us consider one instance of the line equation $y - mx - q = 0$.

Example 1. Let us consider the equation $y - x + 1$ which is represented by the following graph:



All the points that lay on the represented line are the affine variety defined by $y - x + 1$.

Example 2. Now, let us consider two polynomials, a circle $x^2 + y^2 - 1 = 0$ and a parabolic curve $x^2 + 2y - 4 = 0$. The points in their intersection $((-2, 0), (0, 2), (2, 0))$ represent the affine variety defined by the two equations.



Lemma 1. *If $V = \mathbb{V}(f_1, \dots, f_m), W = \mathbb{V}(g_1, \dots, g_n)$ are two affine varieties, then also $V \cap W$ and $V \cup W$ are affine varieties. In particular:*

$$V \cap W = \mathbb{V}(f_1, \dots, f_m, g_1, \dots, g_n)$$
$$V \cup W = \mathbb{V}(f_i g_j | 1 \leq i \leq m, 1 \leq j \leq n)$$

Example 3. Recall Example 2. The affine variety $\mathbb{V}(x^2 + y^2 - 1, x^2 + 2y - 4)$ is the intersection of two bigger affine varieties, in particular:

$$\mathbb{V}(x^2 + y^2 - 1, x^2 + 2y - 4) = \mathbb{V}(x^2 + y^2 - 1) \cap \mathbb{V}(x^2 + 2y - 4).$$

2.2 Ideals

Definition 2 (Ideal). *A subset $I \subseteq R$ is an **ideal** if it satisfies the following conditions:*

- $0 \in I$

- *if $f, g \in I$, then $f + g \in I$*
- *if $f \in I$ and $h \in \mathbb{K}[x_1, \dots, x_n]$, then $hf \in I$*

Let $f_1, \dots, f_s$ be polynomials in $\mathbb{K}[x_1, \dots, x_n]$, then the set

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

is an ideal in R generated by f_i for $1 \le i \le s$.

The set of polynomials which defines the ideal is called the generating basis of the ideal. However, a given ideal can have many different basis. From an algebraic point of view, the ideal is the set of polynomials which are linear combinations of the polynomials in the generating set (basis). Therefore, it makes sense to link the concept of ideal to the variety and vice versa.

Definition 3 (Variety of ideal). *Let $s, n \in \mathbb{N}$ and let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in $\mathbb{K}[x_1, \dots, x_n]$. Then, the set*

$$\mathbb{V}(I) = \mathbb{V}(f_1, \dots, f_s) := \{(a_1, \dots, a_n) \in \mathbb{K}^n \mid f_i(a_1, \dots, a_n) = 0, 1 \le i \le s\}$$

is called the affine variety of the ideal I. Moreover, for any field $\mathbb{K} \subset \mathbb{K}'$, $\mathbb{V}_{\mathbb{K}'}(I)$ denotes the set of solutions $(a_1, \dots, a_n)$ over the n-dimensional affine space $A^n(\mathbb{K}')$. In particular, $\mathbb{V}_{\bar{\mathbb{K}}}(I)$ denotes the variety of I over the algebraic closure $\bar{\mathbb{K}}$ of $\mathbb{K}$.

The variety of an ideal is independent of the generating set of polynomials. There could be multiple sets of polynomials which define the same set of solutions. Therefore, if $f_1, \dots, f_m$ and $g_1, \dots, g_n$ are basis of the same ideal, that is $\langle f_1, \dots, f_m \rangle = \langle g_1, \dots, g_n \rangle$, then $\mathbb{V}(f_1, \dots, f_m) = \mathbb{V}(g_1, \dots, g_n)$.

Example 4. Given the ideal $I = \langle f_1 = x^4 + 2x^2 + 2y^2 - 17, f_2 = 2x^4 + 4x^2 + 3y^2 - 33 \rangle \subset \mathbb{R}[x, y]$, we can find another basis which helps us to determine its variety $\mathbb{V}(I)$.

In this simple example, $2f_1 - f_2 = 0$ gives us the polynomial $y^2 - 1 = 0$. From it we obtain the polynomial involving only the variable x: $x^4 + 2x^2 - 15 = 0$.

The ideal $\langle x^4 + 2x^2 - 15, y^2 - 1 \rangle = \langle f_1, f_2 \rangle$, therefore their variety is the same. It is straightforward to notice that finding the solutions of the new basis polynomials is easier. The roots of $y^2 - 1$ are $y = \pm 1$. As regards $x^4 + 2x^2 - 15$, it can be factorized in $(x^2 + 5)(x^2 - 3)$ where the possible roots are $x = \pm\sqrt{3}$. The variety of the ideal is then given by the couples $(\pm\sqrt{3}, \pm 1)$.

In the same way we can define the ideal of the variety.

Definition 4 (Ideal of variety). *If $\mathbb{V} \in \mathbb{K}^n$ is an affine variety, then $I(\mathbb{V}) \subseteq \mathbb{K}[x_1, \dots, x_n]$ where*

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

is an ideal, and we call it the ***ideal of the variety****.*

Taking all together, given an ideal I, the complexity of defining $\mathbb{V}(I)$ can differ depending on the chosen generating basis. Moreover, the ability to change the basis (meaning the view point) of the ideal could lead to an easier way of computing the variety and then the solutions of the system of equations defined by the generating set. One important class of these "nice" sets is the class of *Gröbner Basis* which we are going to analyse in the next sections. An immediate question that can rise from the above definitions is whether $\langle f_1, \dots, f_m \rangle \overset{?}{=} I(\mathbb{V}(f_1, \dots, f_m))$. For arbitrary fields the relation between the ideal generated by the set of polynomials and the ideal generated by their affine variety is not well-defined. Example 5 shows a case where the inequality holds. However, for algebraically closed fields[1] this relation is explained by the Nullstellensatz Theorem, in particular

$$\langle f_1, \dots, f_m \rangle = I(\mathbb{V}(f_1, \dots, f_m)).$$

On the other hand, the ideal of a variety always uniquely determine the variety.

Example 5. Let $J = \langle x^4, y^2 \rangle$ be an ideal in $\mathbb{C}[x, y]$. The equations $x^4 = 0$ and $y^2 = 0$ imply that $\mathbb{V}(J) = \{(0, 0)\}$. Now the question is: if we compute the ideal of the variety $\{(0, 0)\}$, will it be J? Otherwise, what is the relation between the new ideal and J?

It can be easily proven that $I(\{(0, 0)\}) = \langle x, y \rangle$. Hence, $G = I(\mathbb{V}(J)) = \langle x, y \rangle \neq J$. Now, what is the relation between G and J?

We can notice that G is larger than J due to the fact that $x \in G = \langle x, y \rangle$, but $x \notin J = \langle x^4, y^2 \rangle$. Indeed, each polynomial in J must have the form $h_1x^4 + h_2y^2$ where $h_1, h_2 \in \mathbb{C}[x, y]$, meaning that each polynomial in J should have total degree at least 2: x does not satisfy this condition.

As a result, we can conclude that $J \subset G$, and the equality does not hold.

Proposition 1. *Let V, W be two affine varieties, then*

$$V \subseteq W \iff I(V) \supseteq I(W)$$
$$V = W \iff I(V) = I(W)$$

Before going to the definition of Gröbner Basis and how they can be used to describe the ideals and their properties, it is worth describing how the polynomials can be represented and used to change the complexity of some operations related to ideals and varieties.

2.3 Polynomials: different view points

One of the crucial operation in defining ideals is the polynomial division. Since the secondary school, we are used to compute the division algorithm between univariate polynomials and one of the main properties that we implicitly apply is the term ordering. When computing the division we sort the terms depending on the degree of the variable in descending order. What happens with multivariate polynomials?

The multivariate polynomial division strongly depends on the term order, meaning that the result could be (and often is) completely different. There exist multiple ways of sorting the terms in a polynomial, and we are going to list (also through examples) some of them. Generically speaking, all the monomial (term) orderings can be described by the following definition.

[1]A field $\mathbb{K}$ is algebraically closed if every non-constant polynomial in $\mathbb{K}[\mathbf{x}]$ has a root in $\mathbb{K}$. Given a generic field $\mathbb{K}$, we define its algebraic closure as $\overline{\mathbb{K}}$.

Definition 5 (Monomial Ordering). *A monomial ordering $\prec$ on $\mathbb{K}[x_1,\ldots,x_n]$ is a relation $\prec$ on $\mathbb{Z}^n_{\geq 0}$, or rather a relation on the set of monomials x^α where $\alpha \in \mathbb{Z}^n_{\geq 0}$ satisfying the following conditions:*

- *$\prec$ is a total (or linear) order on $\mathbb{Z}^n_{\geq 0}$*
- *if $\alpha \prec \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$, then $\alpha + \gamma \prec \beta + \gamma$*
- *$\prec$ is a well-ordering on $\mathbb{Z}^n_{\geq 0}$, meaning that every non empty subset of $\mathbb{Z}^n_{\geq 0}$ has a smallest element under the relation $\prec$.*

What follows is a brief description of what are the well-known monomial orderings:

- Lexicographic (LEX)
- Reverse lexicographic (RLEX)
- Degree lexicographic (DL)
- Degree reverse lexicographic (DRL)
- Elimination order (Elim)
- Weighted order (the general one)

For each of them we will provide an example of how they look like with respect to the following multivariate polynomial

$$f = x^3z - y^3 + xy^2z - xy + 1$$

where the lexicographic order of the chosen variables is $x > y > z$.

Definition 6 (Lexicographic Order). *Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be in $\mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{lex} \beta$ if the leftmost nonzero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is positive. We say $x^\alpha >_{lex} x^\beta$ if $\alpha >_{lex} \beta$.*

Example 6. We will sort the monomials in f w.r.t the lexicographic order. As an example, let us take x^3z and xy^2z into account. Their degrees are represented by the vectors $\alpha = (3, 0, 1)$ and $\beta = (1, 2, 1)$. From the vector difference $(3, 0, 1) - (1, 2, 1) = (2, -2, 0)$ we can notice that the leftmost nonzero entry is positive, therefore $x^3z >_{\mathsf{LEX}} xy^2z$.

As a result, the monomials in f would be sorted as: $x^3z + xy^2z - xy - y^3 + 1$

Definition 7 (Reverse Lexicographic Order). *Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ be in $\mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{revlex} \beta$ if the rightmost nonzero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is negative. We say $x^\alpha >_{revlex} x^\beta$ if $\alpha >_{revlex} \beta$.*

Example 7. We will sort the monomials in f w.r.t the reverse lexicographic order. As an example, let us take 1 and xy^2z into account. Their degrees are represented by the vectors $\alpha = (0, 0, 0)$ and $\beta = (1, 2, 1)$. From the vector difference $(0, 0, 0) - (1, 2, 1) = (-1, -2, -1)$ we can notice that the rightmost nonzero entry is negative, therefore $1 >_{\mathsf{RLEX}} xy^2z$.

As a result, the monomials in f would be sorted as: $+1 - xy - y^3 + x^3z + xy^2z$

Definition 8 (Degree Lexicographic Order). *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{\mathsf{DL}} \beta$ if*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and } \alpha >_{lex} \beta$$

We say $x^\alpha >_{\mathsf{DL}} x^\beta$ if $\alpha >_{\mathsf{DL}} \beta$.

Example 8. We will sort the monomials in f w.r.t the degree lexicographic order. As an example, let us take x^3z and xy^2z into account. Their degrees are represented by the vectors $\alpha = (3, 0, 1)$ and $\beta = (1, 2, 1)$. $\sum_{i=1}^{n} \alpha_i = 4$ and $\sum_{i=1}^{n} \beta_i = 4$, therefore, we need to use the lexicographic order to determine how they must be sorted. As shown by Example 6, $x^3z >_{\mathsf{LEX}} xy^2z$.

As a result, the monomials in f would be sorted as: $x^3z + xy^2z - y^3 - xy + 1$

Definition 9 (Degree Reverse Lexicographic Order). *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{\mathsf{DRL}} \beta$ if*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \text{ or } |\alpha| = |\beta| \text{ and } \alpha >_{revlex} \beta$$

We say $x^\alpha >_{\mathsf{DRL}} x^\beta$ if $\alpha >_{\mathsf{DRL}} \beta$.

Example 9. We will sort the monomials in f w.r.t the degree reverse lexicographic order. As an example, let us take x^3z and xy^2z into account. Their degrees are represented by the vectors $\alpha = (3, 0, 1)$ and $\beta = (1, 2, 1)$. $\sum_{i=1}^{n} \alpha_i = 4$ and $\sum_{i=1}^{n} \beta = 4$, therefore, we need to use the reverse lexicographic order to determine how they must be sorted. Recall Example 7, $x^3z >_{\mathsf{RLEX}} xy^2z$.

As a result, the monomials in f would be sorted as: $x^3z + xy^2z - y^3 - xy + 1$

Definition 10 (Elimination order). *A monomial ordering on $\mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_m]$ is an* ***elimination ordering*** *for $x_1, \ldots, x_n$ if*

$$f \in \mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_m] \wedge LM(f) \in \mathbb{K}[y_1, \ldots, y_m] \rightarrow f \in \mathbb{K}[y_1, \ldots, y_m]$$

Example 10. The elimination order is used in conjunction with one of the previous monomial orderings. In particular, we say that $\prec$ is an elimination order for variable x if and only if $x > y^m$ and $x > z^m$ for all $m \geq 0$. The monomial ordering for the remaining variables y, z is one of the previously described (LEX, RLEX, $\ldots$).

This means that the variable x must be involved in all the monomials at the beginning of the polynomial: $x^3z + xy^2z - xy - y^3 + 1$.

Moreover, it can never happen that a monomial involving the variable x is between two monomials that does not involve x: $y^3 + xy + z$ is impossible.

Every monomial ordering can be defined by a series of weights. Moreover, different monomial orderings can be combined to exploit different properties.

Definition 11 (Weighted monomial order). *Given a weight vector $w = (w_1, \ldots, w_n) \in \mathbb{R}^n_{\geq 0}$, where*

$w_1 \neq 0$. *We say that w is associated with the monomial ordering $\prec$, defined by:*

$$\prod_{i=1}^{n} x_i^{\alpha_i} \prec \prod_{i=1}^{n} x_i^{\beta_i} \iff \begin{cases} \sum_{i=1}^{n} w_i\alpha_i > \sum_{i=1}^{n} w_i\beta_i \\ \sum_{i=1}^{n} w_i\alpha_i = \sum_{i=1}^{n} w_i\beta_i, \alpha \prec_M \beta \end{cases}$$

where M is another monomial order like ***LEX****,* ***RLEX****, etc..*

Example 11. Let us define the following weights with respect to x, y, z:

$$w_x = 1,\ w_y = 2,\ w_z = 3.$$

Moreover, assume that $M =$ RLEX.

As an example, let us take x^3z and xy^2z into account. Their degrees are represented by the vectors $\alpha = (3, 0, 1)$ and $\beta = (1, 2, 1)$. Hence, the weighted sum is given by $\sum_{i=1}^{n} w_i\alpha_i = 6$ and $\sum_{i=1}^{n} w_i\beta_i = 8$, and, as a result, $xy^2z >_{\mathsf{W}_{DRL}} x^3z$.

The original polynomial would be rewritten as: $xy^2z - y^3 + x^3z - xy + 1$

2.4 Gröbner basis

The idea behind the Gröbner basis is to find a unique basis (point of view) of the ideal which can be used to answer the following questions:

1) ideal description problem: does every ideal have a finite basis?

2) ideal membership problem: given $f \in \mathbb{K}[x_1, \ldots, x_n]$ and $I = \langle f_1, \ldots, f_s \rangle$, does f belong to I?

3) solving systems of polynomial equations: finding the variety of the ideal.

Generically, the ideal membership problem can be solved by looking at the remainder of the multivariate polynomial division. The tricky part comes from the fact that the multivariate polynomial division highly depends on the order of the polynomials within an ideal basis and on the chosen monomial ordering. Sometimes, the result is not uniquely determined and that is the reason why we absolutely need a "good" representative of the ideal, the Gröbner basis. The following example is taken from [15, Ch. 2, §3, Example 5].

Example 12. Let $f_1 = xy - 1,\ f_2 = y^2 - 1 \in \mathbb{K}[x, y]$ with the LEX order. Dividing $f = xy^2 - x$ by $F = (f_1, f_2)$, the result is

$$xy^2 - x = y(xy - 1) + 0(y^2 - 1) + (-x + y).$$

With $F = (f_2, f_1)$, the result of the division by F is:

$$xy^2 - x = x(y^2 - 1) + 0(xy - 1) + 0.$$

From the second result, the zero remainder tells us that $f \in \langle f_1, f_2 \rangle$ even if from the first computation we obtained a non-zero remainder. As said before, the order of the "generators" matters.

The ideal description problem is solved by the Hilbert Basis theorem.

Theorem 1 (Hilbert Basis theorem). *Every ideal $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ has a finite generating set. In other words, $I = \langle g_1, \dots, g_s \rangle$ for some $g_1, \dots, g_s \in I$.*

Moreover, we are searching for basis with "good" properties that can guarantee the uniqueness we are looking for. This basis are called Gröbner basis.

Definition 12 (Gröbner Basis). *Let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in $\mathbb{K}[x_1, \dots, x_n]$ and let $\prec$ be a valid monomial ordering. A finite subset $G = \{g_1, \dots, g_t\}$ of I different from $\{0\}$ is said to be a **Gröbner Basis** (or Standard Basis) w.r.t. $\prec$ if*

$$\langle LM(g_1), \dots, LM(g_t) \rangle = \langle LM(I) \rangle$$

where $LM()$ denotes the leading monomial of a polynomial with respect to $\prec$.

With Gröbner basis, we are solving the issue given by the order of the generators, as shown in Example 12. With respect to the monomial ordering, there is not a unique basis. For each monomial ordering the computed Gröbner basis could be different. Therefore, we will often denote a Gröbner basis w.r.t to a monomial ordering $\prec$ as $G_\prec$.

Remark 1. Let $I \in \mathbb{K}[x_1, \dots, x_n]$ be an ideal and let $G = \{g_1, \dots, g_s\}$ be a Gröbner basis for I. Given $f \in \mathbb{K}[x_1, \dots, x_n]$ there is a unique remainder $r \in \mathbb{K}[x_1, \dots, x_n]$ with the following properties:

- No term of r is divisible by any of $LT(g_1), LT(g_2), \dots, LT(g_s)$.
- There is $g \in I$ such that $f = g + r$.

The polynomial r is the remainder, and it is unique, no matter how the generators are sorted when using the division algorithm. Moreover, f can be shown as $q_1 g_1 + \dots + q_s g_s + r$ where the remainder r is uniquely determined and the quotients q_i depend on how the generators are sorted during the division algorithm.

Due to ease of notation, we will use $\overline{f}^G$ to denote the remainder on division of f by the set G.

Example 13. Given $G = \{g_1 = x + z, g_2 = y - z\}$ a Gröbner basis with respect to the LEX monomial order. Let $f = xy$ be a generic polynomial in $\mathbb{K}[x, y, z]$, we are going to compute the division of f by (g_1, g_2) and (g_2, g_1).

1) $f = y(x + z) - z(y - z) - z^2$

2) $f = x(y - z) + z(x + z) - z^2$

As previously remarked, the remainder is the same $(-z^2)$ whilst the quotients are different.

Fixed a monomial ordering, there exist a unique **reduced Gröbner basis**.

Definition 13 (Reduced Gröbner Basis). *Let G be a Gröbner basis for ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ with respect to a monomial ordering $\prec$, if*

- *$LC(g) = 1$ for all $g \in G$*
- *for all $g \in G$, no monomial of g lies in $\langle LT(G \setminus \{g\})\rangle$*

*G is said to be a **reduced Gröbner basis**.*

An important property of reduced Gröbner basis is that the polynomial division modulo a Gröbner basis yields unique division remainders. Then, if G is a reduced Gröbner basis for the ideal I we are able to uniquely represent residue classes in the quotient ring R/I. The quotient ring R/I is a $\mathbb{K}$-vector space, called the quotient space. The basis which defines that quotient space can be finite or infinite-dimensional. In particular, a standard basis for $\mathbb{K}[x_1, \ldots, x_n]/I$ is given by the set of monomials

$$B_I := \{X^\alpha \mid X^\alpha \notin \langle LM(I)\rangle\} = \{X^\alpha = \prod_{i=1}^{n} x_i^{\alpha_i} \mid X^\alpha \notin \langle LM(G)\rangle\}.$$

Definition 14 (Zero-dimensional Ideal). *Let I be a non-zero ideal in $\mathbb{K}[x_1, \ldots, x_n]$, let $\prec$ a valid monomial ordering and let G be a reduced Gröbner basis for I with respect to $\prec$. If the quotient space $\mathbb{K}[x_1, \ldots, x_n]/I$ is finite-dimensional, that is*

$$d_I = dim_{\mathbb{K}}(\mathbb{K}[x_1, \ldots, x_n]/I) = |B_I| < \infty$$

*then I is called **Zero-dimensional ideal**.*

If the quotient space is finite-dimensional, each remainder can be written as vector in the basis monomials of B_I. That means we can define a linear matrix $T_j : \mathbb{K}[x_1, \ldots, x_n]/I \to \mathbb{K}[x_1, \ldots, x_n]/I$ corresponding to the multiplication by x_j for all $1 \leq j \leq n$.

Definition 15 (Multiplication matrix). *Let I be a zero-dimensional ideal in $\mathbb{K}[x_1, \ldots, x_n]$, G a reduced Gröbner basis for I with respect to a monomial ordering $\prec$ and $B_I = (e_1, \ldots, e_{d_I})$ the standard basis for the quotient space $\mathbb{K}[x_1, \ldots, x_n]/I$, the **multiplication matrix** T_j of x_j is defined by the square matrix whose i-th vector is represented by the coefficients of the monomials of the basis B_I contained in the reduction of $x_j e_i$ modulo the Gröbner Basis G.*

There is a strong connection between zero dimensional ideals, their quotient space and their variety. In particular, for zero dimensional ideals, the number of solutions to an equation system equals the dimension of the quotient space (counted with multiplicities). We note that, in general, counting the number of solutions or finding a good bound on it could be a difficult problem.

Theorem 2. *Let I be a zero-dimensional ideal in $\mathbb{K}[x_1, \ldots, x_n]$. There exists well-defined multiplicities m_P at each point $P \in \mathbb{V}_{\mathbb{K}}(I)$ such that*

$$d_I = \sum_{P \in \mathbb{V}_{\mathbb{K}}(I)} m_P.$$

This means that the number of solutions over the algebraic closure $\overline{\mathbb{K}}$ counted with multiplicities is equal to the dimension of the quotient ring space.

Definition 16 (Ideals in shape position). *Let $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ be an ideal. We say that I is in **shape position** if the reduced LEX Gröbner Basis of I has the form*

$$\{x_1 - g_1(x_n), x_2 - g_2(x_n), \dots, g_n(x_n)\}$$

where $\mathsf{deg}(g_i) < \mathsf{deg}(g_n)$ for each $1 \leq i < n$

Shape position ideals are a subclass of zero-dimensional ideals. Due to their structure, it is straightforward to see that $d_I = \deg(g_n)$ and the complexity of computing the variety of such an ideal is equivalent to the complexity of factorizing the univariate polynomial g_n.

2.5 Regular systems of polynomials

In this section we introduce the concept of regular sequences of homogeneous polynomials which will be necessary in Chapter 3 to understand one of the algorithms for computing Gröbner basis. Let us start with a basic definition.

Definition 17 (Zero-divisor). *A non-zero element a is said to be a zero-divisor in a commutative ring R if $a \cdot b = 0$ for some non-zero element $b \in R$.*

Definition 18 (Regular sequence). *A sequence of polynomials $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ is **regular** if, for all $1 \leq i \leq s$, the equivalence class of f_i, denoted as $[f_i]$, is not a zero-divisor in the quotient ring $\mathbb{K}[x_1, \dots, x_n]/\langle f_1, \dots, f_{i-1}\rangle$. In other words, if there exist $gf_i \in \langle f_1, \dots, f_{i-1}\rangle$, then g (absolutely not f_i) belongs to $\langle f_1, \dots, f_{i-1}\rangle$.*

Usually, regular sequences come into play when dealing with homogeneous polynomials. Hence, in the rest of the section we will focus on homogeneous polynomials and ideals.

Proposition 2. *Let $\mathbb{K}$ be an algebraically closed field and let $I \subseteq R = \mathbb{K}[x_1, \dots, x_n]$ be a homogeneous ideal. Let $f \in \mathbb{K}[x_1, \dots, x_n]$ be a nonconstant homogeneous polynomial. If the equivalence class $[f]$ is not a zero-divisor of the quotient ring R/I, then*

$$\dim(\mathbb{V}(I + \langle f \rangle)) = \dim(\mathbb{V}) - 1 \text{ if } \dim(\mathbb{V}) > 0$$
$$\dim(\mathbb{V}(I + \langle f \rangle)) = 0 \text{ if } \dim(\mathbb{V}) = 0$$

In order to determine if we are dealing with a regular sequence or not, we can check the zero-divisor condition. Indeed, given a homogeneous ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ such that $I \neq \langle 1 \rangle$ and a homogeneous polynomial $f \in \mathbb{K}[x_1, \dots, x_n]$ of degree d, f is not a *zero-divisor* in $\mathbb{K}[x_1, \dots, x_n]/I$ if and only if the Hilbert series $\mathbf{H}_{I+\langle f \rangle}(z) = (1 - z^d)\mathbf{H}_I(z)$. An immediate consequence of this result is the following proposition.

Proposition 3. *Given a sequence $\mathcal{F}$ of homogeneous polynomials $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ with degrees $d_i = \mathsf{deg}(f_i)$, $\mathcal{F}$ is a regular sequence if and only if its Hilbert series is defined as*

$$\mathbf{H}_I(z) := \frac{\prod_{j=1}^{s}(1 - z_j^d)}{(1 - z)^n}.$$

Moreover, if $s = n$ (meaning that the number of equations is equal to the number of variables), the sequence $\mathcal{F}$ is regular if and only if $\mathbf{H}_I(z)$ is a polynomial (that is, $(1-z)^n$ divides $\prod_{j=1}^{s}(1-z_j^d)$).

2.5.1 Variables in Noether position

Checking whether the variables are in Noether position or not represents another way to determine if we are dealing with regular sequence or not. To fully understand this notion, recall the definition of elimination order 10.

Given a set of polynomials $f_1, \dots, f_s \in R = \mathbb{K}[x_1, \dots, x_n]$, we say that a variable $x_i \in R/\langle f_1, \dots, f_s \rangle$ is an **algebraic integer** over $\mathbb{K}[x_{s+1}, \dots, x_n]$ if there exist a polynomial $g \in \mathbb{K}[x_i, x_{s+1}, \dots, x_n] \cap \langle f_1, \dots, f_s \rangle$ that is monic w.r.t x_i.

Definition 19. *The variables $x_1, \dots, x_s$ are in Noether position w.r.t. the system $f_1, \dots, f_s$ if their canonical images in $R/\langle f_1, \dots, f_s \rangle$ are algebraic integers over $\mathbb{K}[x_{s+1}, \dots, x_n]$ and $\mathbb{K}[x_{s+1}, \dots, x_n] \cap \langle f_1, \dots, f_s \rangle = \langle 0 \rangle$*

Moreover, we say that the variables $x_1, \dots, x_n$ are in **simultaneous Noether position** with respect to the system of polynomials $f_1, \dots, f_s$ if the variables $x_1, \dots, x_i$ are in Noether position with respect to $f_1, \dots, f_i$ for all $1 \leq i \leq s$. Now, we have all the ingredients to define regular sequences on Noether position variables.

Proposition 4 (Regular sequence). *Let $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ be a system of homogeneous polynomials such that $\langle f_1, \dots, f_s \rangle \neq \langle 1 \rangle$. If the variables $x_1, \dots, x_s$ are in Noether position w.r.t. the system $f_1, \dots, f_s$, then the sequence $f_1, \dots, f_s, x_{s+1}, \dots, x_n$ is regular. In addition, if the variables $x_1, \dots, x_n$ are in simultaneous Noether position w.r.t $f_1, \dots, f_s$, then $f_1, \dots, f_s, x_{s+1}, \dots, x_n$ is a regular sequence.*

The connection between Noether position variables and regular sequences works both ways. That means that if we have a regular sequence, but the involved variables are not in Noether position, then, if the field is sufficiently large, this variable condition can be reached by a linear change of coordinates.

3

Computational algebra

Chapter 2 aims at introducing the reader to the basic mathematical concepts needed to understand the algorithms and their application to the field of cryptography. From now we are going to illustrate the methodologies and the algorithms which were developed (and are currently under investigation) to solve the following problems:

1) how to compute a Gröbner basis

2) how to solve systems of polynomial equations

Due to ease of notation, we use $f \to_G 0$ to denote the reduction to zero of f by the set of polynomials G, meaning that the remainder on division of f by the set G is zero. In other words, f can be written as a linear combination of the polynomials in G. The mathematical notions that will be introduced are taken from [15, 16] unless otherwise specified.

3.1 Gröbner basis computation

Buchberger was the first one to answer at the problem of computing Gröbner basis. Moreover, his algorithm and contributions were used by researchers as a guideline and as an inspiration to develop newer and more efficient algorithms. In this subsection we are going to present the main algorithms and discuss the possible variants.

3.1.1 Buchberger's algorithm

Prerequisites From the Gröbner basis theory, we have seen that, given a sequence of polynomials $\{f_1, \ldots, f_n\}$, it could be a Gröbner basis if and only if there are no polynomial combinations of the f_i such that the leading terms are not in the ideal generated by the leading terms of each f_i.

Example 1. Given $f_1 = x^2 + y$ and $f_2 = x$, the combination

$$f_1 - xf_2 = y$$

triggers the cancellation of x^2 and x which are the leading terms of f_1 and f_2 respectively. As a consequence, the leading term of the result is y, which is not a combination of $LT(f_1)$ and $LT(f_2)$.

Hence, $\{f_1, f_2\}$ is not a Gröbner basis, and we know that, to build a GB, we must include the polynomial y (the combination of f_1 and f_2) into the set of polynomials.

Buchberger introduced the concept of *S-polynomials*. We will use `multideg(f)` to denote the tuple of integers corresponding to the degrees of $LT(f)$, e.g. if $f = x^2y + z$, $\mathsf{multideg}(f) = (2, 1, 0)$ where $2, 1$ and 0 are the exponents of the variables x, y, z respectively.

Definition 1 (S-polynomial). *Let $f, g \in \mathbb{K}[x_1, \dots, x_n]$ be non-zero polynomials.*

- *If $\mathsf{multideg}(f) = \alpha$ and $\mathsf{multideg}(g) = \beta$, then let $\gamma = (\gamma_1, \dots, \gamma_n)$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for each $1 \le i \le n$. We call $x_1^{\gamma_1} x_2^{\gamma_2} \cdots x_n^{\gamma_n} = \boldsymbol{x}^\gamma$ the **least common multiple** of $LM(f)$ and $LM(g)$: $\boldsymbol{x}^\gamma = \mathsf{lcm}(LM(f), LM(g))$.*
- *The **S-polynomial** of f and g is the combination*

$$\mathcal{S}(a, b) = \frac{\mathsf{lcm}(LM(a), LM(b))}{LT(a)} a - \frac{\mathsf{lcm}(LM(a), LM(b))}{LT(b)} b.$$

As a result of the previous definition, we can present the first Buchberger's criterion.

Theorem 1 (S-pair critetion (Buchberger's criterion)). *Let I be a polynomial ideal. Then, a basis $G = \{g_1, \dots, g_s\}$ of I is a Gröbner basis of I if and only if for all pairs $i \neq j$ where $1 \le i, j \le s$, the remainder of division of $\mathcal{S}(g_i, g_j)$ by G (listed in some order) is zero:*

$$\overline{\mathcal{S}(g_i, g_j)}^G = 0$$

Thanks to Theorem 1, it is easy to show whether a sequence of polynomials G is a Gröbner basis or not. Those results were immediately refined as follows.

Theorem 2 (Buchberger's criterion refinment). *A basis $G = \{g_1, \dots, g_s\}$ for an ideal I is a Gröbner basis if and only if $\mathcal{S}(g_i, g_j) \to_G 0$ for each $1 \le i, j \le s$ where $i \neq j$. The notation $f \to_G 0$ denotes the possibility to represent f as a linear combination of the basis polynomials $g_1, \dots, g_s$, no matter of how they are sorted to perform the polynomial division. This representation is called **Standard representation**.*

Proposition 1. *There are some situations where an S-polynomial is guaranteed to have a standard representation. Indeed, given a finite set $G \subseteq \mathbb{K}[x_1, \dots, x_n]$, suppose that we have $f, g \in G$ such that $\gcd(LM(f), LM(g)) = 1$. Then $\mathcal{S}(f, g) \to_G 0$ is guaranteed. Moreover, the same check is given by the equivalent relation*

$$\mathsf{lcm}(f, g) = LM(f) \cdot LM(g).$$

The third improvement to the Buchberger's conditions requires a generalization of the S-polynomial notion.

Definition 2. *Let $F = \{f_1, \dots, f_s\}$. A **syzygy** on the leading terms $LT(f_1), \dots, LT(f_s)$ of F is an s-tuple of polynomials $S = (h_1, \dots, h_s) \in \mathbb{K}[x_1, \dots, x_n]^s$ such that*

$$\sum_{i=1}^{s} h_i LT(f_i) = 0.$$

We denote as $S(F)$ the subset of $\mathbb{K}[x_1, \dots, x_n]^s$ consisting of all the syzygies on the leading terms of F.

From another point of view, the syzygies are the possible linear combinations of the polynomials in F that produce the cancellation of their leading terms. In other word, the S-polynomial is an example of syzygy because it leads to the cancellation of the leading term of both the involved polynomials. Moreover, the set of syzygies is closed under coordinate-wise sum and multiplication by polynomials and, given F, the set $S(F)$ has a finite basis, that is a finite set of syzygies such that every other syzygy in $S(F)$ can be represented as a linear combination of the basis syzygies. From a theoretical point-of-view the set of syzygies is called a module of the ring $\mathbb{K}[x_1, \dots, x_n]$. Further details can be found in [15, Chapter 2].

At this point we have all the necessary ingredients to define a more advanced S-pair criterion.

Definition 3. *A basis $G = \{g_1, \dots, g_s\}$ for an ideal I is a Gröbner basis if and only if for every element $S = \{H_1, \dots, H_s\}$ in a homogeneous basis for $S(G)$, $S \cdot G \to_G 0$, where $S \cdot G = \sum_{i=1}^{s} H_i g_i$.*

Corollary 1 (Second Buchberger's criterion). *Given $G = \{g_1, \dots, g_s\}$, suppose that $S \subseteq \{S_{ij} \mid i < j\}$ is a basis for $S(G)$, and we have distinct elements $g_i, g_j, g_l \in G$ such that*

$$LT(g_l) \mid \mathsf{lcm}(LT(g_i), LT(g_j)).$$

If $S_{il}, S_{jl} \in S$, then $S \setminus S_{ij}$ is also a basis of $S(G)$ because S_{ij} is nothing else that a linear composition of S_{il} and S_{jl}.

The algorithm The Buchberger's algorithm presented here makes use of the criteria defined so far. In particular, Theorem 1 and Corollary 1 will be of high relevance in the choice of the couples to be processed for generating the remaining elements of a Gröbner basis. Algorithm 1 shows a sketch of a possible implementation of the Buchberger's algorithm where `Criterion(`f_i, f_j, B`)` is `True` if and only if there exist some $l \notin \{i, j\}$ for which the pairs $[i, l]$ and $[j, l]$ are not in B and $LM(f_l)$ divides $\mathsf{lcm}(LM(f_i), LM(f_j))$.

Algorithm 1 Buchberger Gröbner basis algorithm

```
1: procedure GETGB($F = \{f_1, \dots, f_n\}$)    ▷
2:     $B := \{(i, j) \mid 1 \le i < j \le n\}$
3:     $G := F$
4:     $t := n$
5:     while $B \neq \emptyset$ do
6:         Select $(i, j) \in B$
7:         if $lcm(LM(f_i), LM(f_j)) \neq LM(f_i)LM(f_j) \wedge Criterion(f_i, f_j, B) = \text{False}$ then
8:             $r := \text{Red}_G(S(f_i, f_j))$
9:             if $r \neq 0$ then
10:                $t := t + 1$
11:                $f_t := r$
12:                $G := G \cup \{f_t\}$
13:                $B := B \cup \{(i, t) \mid 1 \le i \le t - 1\}$
14:        $B := B \setminus \{(i, j)\}$
15:    return $G$
```

The complexity of the Buchberger's algorithm has been deeply studied, and it has been defined as follows:

$$2\Big(\frac{d^2}{2}+d\Big)^{2^{n-2}}.$$

However, depending on the considered system of equations, ad-hoc complexity bounds can be derived.

3.1.2 F4

Prerequisites To introduce the family of F4 algorithms, we require the concept of Macaulay matrices. Let $F = \{f_1, \dots, f_s\}$ be any set of polynomials in $\mathbb{K}[x_1, \dots, x_n]$ and $\prec$ a generic monomial ordering. For any degree $d \in \mathbb{Z}_+$, the Macaulay matrix $M_{\leq d}$ of F has columns indexed by the terms of degree $\leq d$ in the ring $\mathbb{K}[x_1, \dots, x_n]$, sorted in decreasing order w.r.t $\prec$. The rows of $M_{\leq d}$ are indexed by the polynomials $m_j f_i$, where m_j is a term in $\mathbb{K}[x_1, \dots, x_n]$ such that $\mathsf{deg}(m_j f_i) \leq d$. Hence, the entry (i, j) of $M_{\leq d}$ is the coefficient of the monomial of column j in the polynomial corresponding to the i-th row.

Example 2 (How to build a Macaulay matrix). Let $f_1 = 2x^2+3xy+7$ and $f_2 = 4x+1 \in \mathbb{K}[x, y]$. Suppose we want to build $M_{\leq 3}$, that is the Macaulay matrix up to degree 3, with respect to the lexicographic monomial ordering. First of all, we need to identify what are the monomials up to degree 3 that will identify the columns of the matrix. These monomials, sorted w.r.t LEX in decreasing order, are:

$$1, y, x, y^2, xy, x^2, y^3, xy^2, x^2y, x^3.$$

Now, for each polynomial f_i, we need to determine the monomials $m \in \mathbb{K}[x, y]$ such that $\mathsf{deg}(mf_i) \leq 3$. As regards f_1, it can be multiplied by the following monomials:

$$1, y, x.$$

As regards f_2, it can be multiplied by the monomials:

$$1, y, x, y^2, xy, x^2.$$

Now we have all the ingredients to build $M_{\leq 3}$.

$$\begin{array}{c|cccccccccc|}
 & 1 & y & x & y^2 & xy & x^2 & y^3 & xy^2 & x^2y & x^3 \\
1 \cdot f_1 & 7 & 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 \\
y \cdot f_1 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 0 \\
x \cdot f_1 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 3 & 2 \\
1 \cdot f_2 & 1 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
y \cdot f_2 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
x \cdot f_2 & 0 & 0 & 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
y^2 \cdot f_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 & 0 \\
xy \cdot f_2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\
x^2 \cdot f_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4
\end{array}$$

The number of rows and columns can be predicted by applying simple formulas from combinatorics. In particular, if n_v denotes the number of variables involved and d the maximum degree for which we are building the Macaulay matrix, the number of columns is given by:

$$\text{number of columns} = \binom{d+n_v}{n_v}.$$

The number of rows is given by

$$\text{number of rows} = \sum_{i=1}^{n_e} \binom{d-d_i+n_v}{n_v}$$

where n_e is the number of equations and d_i is the total degree of the polynomial f_i. If we apply those formulas to the example above, we obtain

$$n_{columns} = \binom{3+2}{2} = 10 \text{ and } n_{rows} = \binom{3-2+2}{2} + \binom{3-1+2}{2} = 3+6 = 9.$$

Sometimes, in particular, when dealing with a set of homogeneous polynomials, we are interested in computing the Macaulay matrix of degree exactly d: M_d. Let $f_1 = 2x^2 + 3xy + 7y^2$ and $f_2 = 4x + y \in \mathbb{K}[x, y]$. They are homogeneous polynomials of degree 2 and 1 respectively. We want to compute the Macaulay matrix of degree 3: M_3. This means that the columns will be identified only by the monomials in $\mathbb{K}[x, y]$ of degree 3, which are:

$$y^3, xy^2, x^2y, x^3,$$

sorted in decreasing order as well. Now, for each polynomial f_i, we need to determine the monomials $m \in \mathbb{K}[x, y]$ such that $\mathsf{deg}(mf_i) = 3$. As regards f_1, it can be multiplied by the following monomials: $[y, x]$. As regards f_2, it can be multiplied by the monomials $[y^2, xy, x^2]$. Now we have all the ingredients to build M_3.

$$\begin{array}{c|cccc} & y^3 & xy^2 & x^2y & x^3 \\ \hline y \cdot f_1 & 7 & 3 & 2 & 0 \\ x \cdot f_1 & 0 & 7 & 3 & 2 \\ y^2 \cdot f_2 & 1 & 4 & 0 & 0 \\ xy \cdot f_2 & 0 & 1 & 4 & 0 \\ x^2 \cdot f_2 & 0 & 0 & 1 & 4 \end{array}$$

Even in this case the number of columns and rows can be predicted in advance:

$$n_{columns} = \binom{n_v+d-1}{d} \quad n_{rows} = \sum_{i=1}^{s} \binom{n+d-d_i-1}{d-d_i}.$$

If we apply those formulas to the example above, we obtain

$$n_{columns} = \binom{2+3-1}{3} \quad n_{rows} = \sum_{i=1}^{2} \binom{2+3-d_i-1}{3-d_i} = \binom{2}{1} + \binom{3}{2} = 5.$$

The algorithm On 1999, Faugère introduced the first algorithm for computing Gröbner basis which exploits methods from linear algebra. More than an algorithm, F4 represents a family of linear algebra based algorithms for GB computation, since there are many variants and hybrid alternatives. As the Buchberger's algorithm, F4 makes use of the S-polynomials to determine the acceptance and the termination conditions. Although, whilst Buchberger's algorithm considers one S-polynomial at a time, F4 uses linear algebra techniques like *Gaussian elimination (row-reduction)* to simultaneously compute several S-polynomial remainders. Some algorithms perform Gaussian elimination on the degree d Macaulay matrix for increasing values of d. The question is: when should we stop the computation? Derivations of the Buchberger's Criterion are used as a termination criterion which allows deciding whether a GB has been found or not. This means that a good approximation of the possible higher value of d (called *solving degree*) has to be found in order to measure the complexity of Gröbner basis computations.

The ComputeM subroutine 3 is the crucial part of the algorithm. As you can notice from the main algorithm 2, the set L contains the halves involved in the computation of the S-polynomials separately. Then, ComputeM receives the *old* basis G and the just mentioned set L. Here, the Macaulay matrices come into play.

Algorithm 3 generates the set H from L, then additional polynomials $x^\alpha f_l$ are inserted into H. In particular, for each monomial $x^\beta \in H$ that is divisible by some $LM(f_l)$, a polynomial $x^\alpha f_l$ such that $x^\beta = LM(x^\alpha f_l)$ is included in H if not already present. Including sufficient values guarantees that, at the end of the procedure, $\overline{\mathcal{S}(f_i, f_j)}^G$ is included in G and that $\mathcal{S}(f_i, f_j)$ can be described with the usual standard representation. Indeed, the ComputeM procedure returns the matrix of coefficients M which is immediately reduced to the row-echelon form (linear algebra comes into play). This reduction produces the equations $\overline{\mathcal{S}(f_i, f_j)}^G$ whose leading monomial is not in the set $LM(G)$. Hence, they are added to the set G and the loop has to be repeated. When the set of pairs B is empty the algorithm terminates returning G, a Gröbner basis for $I = \langle f_1, \dots, f_n \rangle$. Proofs of correctness and termination can be found in [15, 22].

Algorithm 2 F4

```
    procedure F4($F = \{f_1, \dots, f_n\}$)                                   ▷
        $G := f$
3:      $t := s$
        $B := \{\{i, j\} \mid 1 \le i < j \le s\}$
        while $B \neq \emptyset$ do
6:          Select $B'$ such that $B' \neq \emptyset \wedge B' \subseteq B$
            $B := B \setminus B'$
            $L := \{\frac{\text{lcm}(LM(f_i), LM(f_j))}{LT(f_i)} \mid \{i, j\} \in B'\}$
9:          $M :=$ computeM(L,G)
            $N :=$ row echelon form of $M$
            $N^+ := \{n \in \text{rows}(N) \mid LM(n) \notin \langle LM(\text{rows}(M)) \rangle\}$
12:         for $n \in N^+$ do
                $t := t + 1$
                $f_t :=$ makepoly($n$, )
15:             $G := G \cup \{f_t\}$
                $B := B \cup \{\{i, t\} \mid 1 \le i \le t\}$
        return $G$
```

Algorithm 3 computeM

```
    procedure COMPUTEM($L, G = \{f_1, \dots, f_r\}$)  ▷
        $H := L$
3:      $done := LM(H)$
        while $done \neq$ Monomials($H$) do
            Select $x^\beta := \max_\prec(\text{Monomials}(H) \setminus done)$
6:          $done := done \cup \{x^\beta\}$
            if $\exists\ f_l \in G$ s.t. $LM(f_l) \mid x^\beta$ then
                Choose randomly $f_l \in G$ s.t. $LM(f_l) \mid x^\beta$
9:              $H := H \cup \{\frac{x^\beta}{LM(f_l)} f_l\}$
        $M :=$ matrix of coefficients of $H$ where columns are Monomials($H$), sorted in decreasing order
    with respect to $\prec$
        return $M$
```

Algorithm 4 makepoly

```
    procedure MAKEPOLY($n, M =$ Monomials($H$))  ▷
        $p := 0$
3:      for $i = 0; i < |n|; i++$ do
            $p = p + n[i] \cdot M[i]$
        return $p$
```

The complexity of linear algebra based algorithms is not well-understood, especially when they do not involve homogeneous polynomials. The complexity of F4 can be represented by the following formula:

$$\mathcal{O}\left(n_e d_{solv} \binom{n_v + d_{solv}}{d_{solv}}^{\omega}\right)$$

where n_e, n_v, d_{solv} denote the number of equations, the number of variables and the solving degree respectively [2]. However, because the complexity is given by the row reduction of Macaulay matrices, this bound can be rewritten as

$$\mathcal{O}\left(\sum_{i=0}^{d_{solv}} \left(\binom{n_v + i - 1}{i} \cdot \sum_{j=1}^{n_e} \binom{n_v + i - \mathsf{deg}(f_j) - 1}{i - \mathsf{deg}(f_j)}\right)\right). \tag{3.1}$$

It is straightforward to notice that one of the key components of the formula is the so-called *solving degree*. Indeed, the complexity of linear algebra based algorithms as F4 strongly depends on a "good" approximation of the maximum degree d reached during the GB computation. [12] tried to analyse the connections between the solving degree of non-homogeneous ideals and the corresponding one derived from their homogenization. For sake of completeness, we present the main relation. Given a set of polynomials $F = \{f_1, \dots, f_s\} \subset \mathbb{K}[x_1, \dots, x_n]$, let $I = \langle F \rangle$ be the ideal generated by the set F. Consider the DRL monomial ordering, the following relation always holds:

$$\text{max.GB.deg}(F^h) = \text{solv.deg}(F^h) = \text{solv.deg}(F) \geq \text{max.GB.deg}(F) = \text{max.GB.deg}(I^h) = \text{solv.deg}(I^h)$$

where F^h denotes the set obtained by homogenizing $f_1, \dots, f_s$, I^h denotes the homogenization of the ideal I, and `max.GB.deg` and `solv.deg` denote the maximum degree of a polynomial in the reduced Gröbner basis of I and the solving degree respectively.

3.1.3 F5

Prerequisites To understand the improvements introduced by Faugère in 2002 [19], we require the generalization of the concepts introduced for the Buchberger's algorithm and F4. In particular, we are going to introduce the notions of *signatures* and s-reductions which are the main features introduced by Faugère to reduce the number of reductions performed by the algorithm, gaining a huge improvement, in particular when the system of polynomials is a *Regular sequence.*

Previously, we have seen how to use S-polynomials to reduce the number of reductions needed during the computation of the Gröbner basis starting from a generic system of polynomials $\{f_1, \dots, f_s\}$. Here, we are going to extend this notion by changing the view point towards the S-polynomials. To give you an idea, let $I = \langle f_1, \dots, f_s \rangle$ be any ideal, then the S-polynomials and the remainders produced during the Gröbner basis computation can all be written as

$$(a_1, \dots, a_s) \cdot (f_1, \dots, f_s) = \sum_{i=1}^{s} a_i f_i \tag{3.2}$$

where $(a_1, \dots, a_s) \in \mathbb{K}[x_1, \dots, x_n]^s$.

We are moving from reasoning on the ring $\mathbb{K}[x_1, \dots, x_n]$ to a module over the ring. In particular, we are going to represent each value as a tuple in $\mathbb{K}[x_1, \dots, x_n]^s$ where we have component-wise vector addition and multiplication (recall the similarity with the **syzygy** introduced in 3.1.1). As we have said before, since each element in the ideal can be represented as in Equation 3.2, we can define a map

$$\psi : \mathbb{K}[x_1, \dots, x_n]^s \to I \text{ such that } \mathbf{a} = (a_1, \dots, a_s) \to \sum_{i=1}^{s} a_i f_i.$$

We can think about ψ as the evaluation than brings the tuples back to the original polynomial. Moreover, if $s = 1$, the map is injective. On the other hand, when $s > 1$, $|\ker(\psi)| > 1$.

The definition of syzygy we gave in 3.1.1 can be generalized as follows.

Definition 4. *Given a set of polynomials $f_1, \dots, f_s$, an s-tuple $\boldsymbol{a} \in \mathbb{K}[x_1, \dots, x_n]^s$ is called* ***syzygy*** *if*

$$\psi(\boldsymbol{a}) = \sum_{i=1}^{s} a_i f_i = 0 \in \mathbb{K}[x_1, \dots, x_n].$$

With respect to the definition given in 3.1.1, here we are dealing with the full polynomials f_i, not only with their leading monomials.

The usual standard basis of $\mathbb{K}[x_1, \dots, x_n]^s$ is represented by the vectors $\mathbf{e}_1 = (1, 0, 0, \dots, 0), \mathbf{e}_2 = (0, 1, 0, \dots, 0), \dots, \mathbf{e}_s = (0, 0, 0, \dots, 1)$. Therefore, each element can be represented as a linear combination of such vectors multiplied by suitable polynomials in $\mathbb{K}[x_1, \dots, x_n]$. A special case, which is called *Koszul syzygy*, is the syzygy defined between two generic polynomials f_1, f_2.

Example 3. Given a list of polynomials $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$, for each pair (i, j) where $1 \leq i < j \leq s$, the *Koszul syzygy* is defined as

$$\mathbf{k}_{ij} = -f_j\mathbf{e}_i + f_i\mathbf{e}_j.$$

All the s-tuples in $\mathbb{K}[x_1, \dots, x_n]^s$ can be expressed with respect to the standard basis $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_s$. With respect to the definition we have given for the Buchberger's algorithm, we are missing a tuple ordering which is essential to develop the rest of the theory. These orderings are an extension of the usual monomial orders to the ring module theory.

Definition 5. *Let $\prec$ be any monomial order on $\mathbb{K}[x_1, \dots, x_n]$, we can define two main extensions:*

- *Term-over-position (TOP): $x^\alpha \boldsymbol{e}_i >_{TOP} x^\beta \boldsymbol{e}_j \iff x^\alpha >_\prec x^\beta$, or $x^\alpha = x^\beta \wedge i > j$*
- *Position-over-term (POT): $x^\alpha \boldsymbol{e}_i >_{POT} x^\beta \boldsymbol{e}_j \iff i > j$, or $i = j \wedge x^\alpha >_\prec x^\beta$*

Without losing generality, the rest of the theory will be built upon the *Position-over-term (POT)* order. Now we have all the ingredients to define the *signature* of an s-tuple.

Definition 6. *Let $\boldsymbol{g} = (g_1, \dots, g_s) \in \mathbb{K}[x_1, \dots, x_n]$. The* ***signature*** *of $\boldsymbol{g}$, denoted as $s(\boldsymbol{g})$, is the term appearing in $\boldsymbol{g}$ that is largest w.r.t. the POT order.*

Example 4. Let us suppose we have the following standard basis $\mathbf{e}_1 = (1, 0, 0), \mathbf{e}_2 = (0, 1, 0), \mathbf{e}_3 = (0, 0, 1)$, and we are using the lexicographic monomial order. Given an s-tuple $\mathbf{g} = (x^2+y, x^2+y^3, z^2+x)$, we know it can be shown as $(x^2 + y)\mathbf{e}_1 + (x^2 + y^3)\mathbf{e}_2 + (x + z^2)\mathbf{e}_3$. What is the signature in the two extension orders?

- Considering *POT* and the fact that $\mathbf{e}_3 > \mathbf{e}_2 > \mathbf{e}_1$, the standard representation of $\mathbf{g}$ would be:

  $$(x + z^2)\mathbf{e}_3 + (x^2 + y^3)\mathbf{e}_2 + (x^2 + y)\mathbf{e}_1.$$

  Hence, the signature must be within $(x + z^2)\mathbf{e}_3$. In addition, we are using the lexicographic monomial order to determine the leading monomial. As a result, the signature $s(\mathbf{g}) = x\mathbf{e}_3$.

- We know $(x^2 + y)\mathbf{e}_1 + (x^2 + y^3)\mathbf{e}_2 + (z^2 + x)\mathbf{e}_3$. Considering *TOP* and lexicographic order, the ordered $\mathbf{g}$ would be:

  $$x^2\mathbf{e}_2 + x^2\mathbf{e}_1 + x\mathbf{e}_3 + y^3\mathbf{e}_2 + y\mathbf{e}_1 + z^2\mathbf{e}_3.$$

  As a result, the signature $s(\mathbf{g}) = x^2\mathbf{e}_2$.

Next, we have to define what is a reduction in the context of signatures and how to use them to determine whether we obtained a Gröbner basis or not. Shortly, we need to extend the Buchberger's Criterion to work on signatures.

Definition 7. *Let $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{K}[x_1, \dots, x_n]^s$. Let $x^\alpha \in \mathbb{K}[x_1, \dots, x_n]$ be a monomial and let $c \in \mathbb{K}$. We say that $\boldsymbol{g} - cx^\alpha\boldsymbol{h} \in \mathbb{K}[x_1, \dots, x_n]^s$ is the result of an s-reduction of $\boldsymbol{g}$ by $\boldsymbol{h}$ if*

i. there is a term bx^β in the polynomial $\psi(\boldsymbol{g}) \in \mathbb{K}[x_1, \dots, x_n]$, such that

$$LT(cx^\alpha\psi(\boldsymbol{h})) = cx^\alpha LT(\psi(\boldsymbol{h})) = bx^\beta$$

ii. $s(\boldsymbol{g}) \geq_{POT} s(x^{\alpha}\boldsymbol{h})$

If the equality holds in ii. we say the reduction is a ***singular s-reduction****, otherwise it is a* ***regular s-reduction****.*

When the reduction is performed by a set H, we say that $\mathbf{g}$ is s-reduced to $\mathbf{k}$ by H if there is a finite sequence of s-reductions such that:

$$\mathbf{g} - c_1 x^{\alpha_1}\mathbf{h}_1 - \cdots - c_l x^{\alpha_l}\mathbf{h}_l$$

where $l \leq |H|$, $h_i \in H$ for $1 \leq i \leq l$ and $c_i \in \mathbb{K}$. If there are no possible reductions, we obtained the equivalent of the remainder on division. In particular, by mixing the definitions of *syzygy* and s-reductions, we can say that a given s-tuple $\mathbf{g} \in \mathbb{K}[x_1, \ldots, x_n]^s$ *s*-reduces to **zero** by some set of vectors H (denoted as $\mathbf{g} \rightarrow_H 0$) if there exist a syzygy $\mathbf{k}$ such that $\mathbf{g}$ can be s-reduced to $\mathbf{k}$ using vectors from H. Indeed, by definition of syzygy, $\psi(\mathbf{k})$ reduces to zero, therefore, even $\mathbf{g}$ reduces to zero as well.

Now, the idea is to replicate what we have done in the Buchberger's algorithm by using s-reductions, signatures and a generalization of the S-polynomials.

Definition 8. *Let $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{K}[x_1, \ldots, x_n]^s$ correspond to monic $\psi(\boldsymbol{g})$ and $\psi(\boldsymbol{h}) \in \mathbb{K}[x_1, \ldots, x_n]$. The corresponding S-vector (the counterpart of the S-polynomials for s-tuples) is the element in $\mathbb{K}[x_1, \ldots, x_n]^s$ defined by:*

$$\mathcal{S}(\boldsymbol{g}, \boldsymbol{h}) = \frac{\mathsf{lcm}(LM(\psi(\boldsymbol{g})), LM(\psi(\boldsymbol{h})))}{LM(\psi(\boldsymbol{g}))} \cdot \boldsymbol{g} - \frac{\mathsf{lcm}(LM(\psi(\boldsymbol{g})), LM(\psi(\boldsymbol{h})))}{LM(\psi(\boldsymbol{h}))} \cdot \boldsymbol{h}.$$

It is straightforward to notice that $\psi(\mathcal{S}(\boldsymbol{g}, \boldsymbol{h})) = \mathcal{S}(\psi(\boldsymbol{g}), \psi(\boldsymbol{h}))$.

Let I be an ideal in $\mathbb{K}[x_1, \ldots, x_n]$ and let $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_s\} \subseteq \mathbb{K}[x_1, \ldots, x_n]^s$ where all $\psi(\mathbf{g}_i)$ are monic. G is said to be a **signature Gröbner basis** for I if

$$\forall_{\mathbf{h} \in \mathbb{K}[x_1, \ldots, x_n]^s} \mathbf{h} \rightarrow_G 0.$$

If $M = x^{\alpha}\mathbf{e}_i$ is a term in $\mathbb{K}[x_1, \ldots, x_n]^s$, we say G is **signature Gröbner basis below** M if all $\mathbf{h} \in \mathbb{K}[x_1, \ldots, x_n]^s$ with $s(\mathbf{h}) <_{POT} M$ s-reduce to zero using G. In other words, the counterpart of the Buchberger's Criterion for signature based methods, can be defined as follow:

Definition 9 (Signature based Buchberger's Criterion).

i. G is a signature Gröbner basis for I if and only if for all S-vectors $\mathcal{S}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ with $1 \leq i < j \leq s$ and all $\boldsymbol{e}_i$ with $1 \leq i \leq s$, $\mathcal{S}(\boldsymbol{g}_i, \boldsymbol{g}_j) \rightarrow_G 0$.

ii. if $M = x^{\alpha}\boldsymbol{e}_i$ is a term in $\mathbb{K}[x_1, \ldots, x_n]^s$, we say G is signature Gröbner basis below M for I if and only if for all S-vectors $\mathcal{S}(\boldsymbol{g}_i, \boldsymbol{g}_j)$ with $1 \leq j \leq s$ and all $\boldsymbol{e}_i$ with $s(\boldsymbol{e}_i) <_{POT} s(M)$, $\mathcal{S}(\boldsymbol{g}_i, \boldsymbol{g}_j) \rightarrow_G 0$.

Since now we know how to obtain a signature Gröbner basis, is it possible to convert it to a usual Gröbner basis?

Proposition 2. *If $G = \{\boldsymbol{g}_1, \dots, \boldsymbol{g}_s\}$ is a signature Gröbner basis for I, then*

$$\psi(G) = \{\psi(\boldsymbol{g}_1), \dots, \psi(\boldsymbol{g}_s)\}$$

is a Gröbner basis for I.

A second criterion, as done for the Buchberger's algorithm can be defined in order to avoid unnecessary s-reductions.

Proposition 3 (Second Criterion). *Let $G = \{\boldsymbol{g}_1, \dots, \boldsymbol{g}_s\}$ and $\boldsymbol{h} = \mathcal{S}(\boldsymbol{g}_i, \boldsymbol{g}_j)$. If G is a signature Gröbner basis below $s(\boldsymbol{h})$ for I and there is syzygy $\boldsymbol{k}$ such that $s(\boldsymbol{k})$ divides $s(\boldsymbol{h})$, then $\boldsymbol{h} \to_G 0$.*

Now, we have all the ingredients to build a signature based Gröbner basis algorithm.

The algorithms During the last two decades, many variants of the original F5 algorithm were designed. Whilst the original F5 algorithm seems a signature based translation of the Buchberger's algorithm, other variants tried to develop hybrid versions between F4 and F5 by using signature based criteria to determine the couples for which is necessary to apply linear algebra steps. Here we present a F5 like algorithm following the original version developed by Faugère [19] and a version of the so-called *Matrix-F5* variant.

Signature based Buchberger-like F5 algorithm Algorithm 5 is quite difference w.r.t. the original F5. In particular, the version proposed by Faugère contains additional reduction rules that improve the process. Additionally, even Faugère suggests implementing the algorithm in an F4 fashion for efficiency reasons.

Algorithm 5 F5

```
    procedure F5($F = \{f_1, \dots, f_s\}$)                                   ▷
        $G := \emptyset$
3:      $P := \{\mathbf{e}_1, \dots, \mathbf{e}_s\}$
        $S := \{-f_j\mathbf{e}_i + f_i\mathbf{e}_j \mid 1 \le i < j \le s\}$        ▷ The Koszul syzygies are pre-computed
        while $P \neq \emptyset$ do
6:          $\mathbf{g} := \mathbf{p}_i$                                       ▷ $\mathbf{p}_i \in P \mid s(\mathbf{p}_i) = \min(\{s(\mathbf{p}_1), s(\mathbf{p}_2), \dots\})$
            $P := P \setminus \{\mathbf{g}\}$
            if Criterion($\mathbf{g}, G \cup S$) $= \perp$ then
9:              $\mathbf{h} := \overline{\mathbf{g}}^G$                        ▷ a regular s-reduction of $\mathbf{g}$ by $G$
                if $\psi(\mathbf{h}) = 0$ then
                    $S := S \cup \{\mathbf{h}\}$
12:             else
                    $\mathbf{h} := \frac{1}{LC(\psi(\mathbf{h}))}\mathbf{h}$
                    $P := P \cup \{\mathcal{S}(\mathbf{k}, \mathbf{h}) \mid \mathbf{k} \in G$ and $\mathcal{S}(\mathbf{k}, \mathbf{h})$ is regular$\}$
15:                 $G := G \cup \{\mathbf{h}\}$
        return $\psi(G)$
```

In order to describe the complexity of $F5$ algorithms, we need to distinguish between the original version proposed by Faugère in 2002, and the corresponding matrix version. In particular, given $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$, the complexity of the first $F5$ algorithm is bounded by

$$\mathcal{O}\Big(\Big(\frac{d_m^{d_m}}{(d_m-1)^{d_m-1}}\Big)^{\omega(n-s)} \cdot n^{2-\frac{\omega}{2}}\Big((d_m-1)\Big(\frac{d_m}{2\pi(d_m-1)^3}\Big)^{\frac{\omega}{2}} + \mathcal{O}(1)\Big)\Big) \tag{3.3}$$

operations in $\mathbb{K}$, where $d_m = \frac{\sum_{i=1}^{s} \mathsf{deg}(f_i)}{s}$ is the arithmetic mean of the polynomial degrees [2]. However, this bound does not hold in general, therefore a generic linear change of variables could be required. A sufficient condition for the bound to hold is that the variables should be in *simultaneous Noether position.* Moreover, if the system is regular and the field is sufficiently large, a generic linear change of variables can be applied to make the variables in this position (note that the linear change of variables is not a free operation, and it could represent the worst part of the process).

Matrix-F5 In this paragraph we present a simple version of the Matrix-F5 algorithm (Algorithm 6) where we give as input a list of homogeneous polynomials and a degree bound, denoted as $\max_{degree}$, up to which a reduced Gröbner basis will be computed. Due to ease of notation, we will use $M_{d,s}$ to denote the Macaulay matrix of degree d w.r.t. the polynomials indexed by $1, \dots, s$ (recall example 2 on how to build Macaulay matrices), and $\overline{M}_{d,s}$ to denote the matrix obtained after the Gaussian elimination on $M_{d,s}$. To keep track of what rows (polynomials) have been already taken into account, we identify each row with a couple (a, b) where a denotes the index of the original polynomial involved (e.g. f_a) and b denotes the monomial which has to multiplied by f_a to obtain the real polynomial corresponding to that row (e.g. bf_a). In other words, we can consider the couple (a, b) a sort of signature.

Further details on the correctness of Algorithm 6 can be found in [2]. As regards the complexity of the algorithm, [2, Theorem 2] provides a good estimate when dealing with a set of homogeneous polynomials $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$ of identical degree $d \geq 2$ where all the variables are in simultaneous Noether position:

$$\mathcal{O}\left(n\left(\frac{(\frac{\lambda_0+1}{\lambda_0})^{2d}-1}{\frac{1}{\lambda_0^2}-\frac{1}{(\lambda_0+1)^2}}\right)^n\left(\frac{1-d^{-1}}{2\pi}\cdot\frac{(1+\lambda_0^{-1})^3-1}{(1+\lambda_0)^{n-s+1}}+\mathcal{O}\left(\frac{1}{n}\right)\right)\right).$$

As a result, we can notice that even F5 strongly depends on a good approximation of the solving degree we have discussed in Subsection 3.1.2.

3.2 Gröbner basis conversion: from graded to LEX monomial order

Usually, it is computationally infeasible to directly compute a lexicographic Gröbner basis due to the fact that it generates higher degree polynomials. The common *modus operandi* is to compute a Gröbner basis w.r.t a graded monomial ordering, and then convert it to a lexicographic basis by using either the GröbnerWalk algorithm or one of the FGLM algorithm variants. Gröbner Walk algorithm was one of the first attempts for converting Gröbner basis between different orders. In particular, it represents a generic version of a change of order algorithm. Although, it is not used in practice due to its higher computational complexity. Therefore, other algorithms are preferred from a practical point of view, since most of the problems we are currently dealing with concern zero-dimensional ideals and well-structured

Algorithm 6 Matrix-F5 algorithm [2].

```
    procedure F5($F = \{f_1, \dots, f_s\}, \texttt{max}_{degree}$)    ▷ $f_i$ homogeneous polynomials such that $\texttt{deg}(f_i) = d_i$
        $G_1 = \emptyset, G_2 = \emptyset, \dots, G_n = \emptyset$
3:      for $d = d_1; d <= \texttt{max}_{degree}; d++$ do
            $M_{d,0} = []$, $\overline{M}_{d,0} = []$
            for $i = 1; i \leq s; i++$ do
6:              if $d < d_i$ then
                    $M_{d,i} = M_{d,i-1}$
                else if $d = d_i$ then
9:                  $M_{d_i,i} = \overline{M}_{d_i,i-1} + [f_i]$    ▷ Row addition in position $(i, 1)$
                else
                    $M_{d,i} = \overline{M}_{d,i-1}$
12:                 $\texttt{criterion-ref} := LT(\overline{M}_{d-d_i,i-1})$    ▷ Take the signatures we have already handled.
                    for $f \in \texttt{Rows}(M_{d-1,i}) \setminus \texttt{Rows}(M_{d-1,i-1})$ do
                        $(k, u) := \texttt{getIndex}(f)$    ▷ $u = x_{j_1} x_{j_1} \cdots x_{j_{d-d_i-1}}$ where $1 \leq j_1 \leq j_2 \leq \cdots \leq j_{d-d_i-1}$
15:                     for $j = d - d_i - 1; j \leq n; j++$ do
                            if $ux_j \notin \texttt{criterion-ref}$ then
                                $M_{d,i} := M_{d_i} + [x_j f]$    ▷ the new row will have index $(i, ux_j)$
18:             $\overline{M}_{d,i} = \texttt{RowEchelonForm}(M_{d,i})$
                for $f \in \texttt{Rows}(\overline{M}_{d,i})$ do
                    if $\overline{f}^{LT(G_i)} = f$ then    ▷ If $f$ is not reducible by the set $LT(G_i)$
21:                     $G_i = G_i + [f]$
        $G := \bigcup_{i=1}^{s} G_i$
        return $G$    ▷ $G$ contains the elements $g_i$ of the reduced GB such that $\texttt{deg}(g_i) \leq \max_{degree}$
```

systems of polynomials.

3.2.1 FGLM

FGLM was proposed by Faugère, Gianni, Lazard and Mora in 1993 to solve one of the major problems deriving from the Gröbner basis computation. It is known that computing a Gröbner basis w.r.t LEX monomial order requires a huge computational power. Back in 1993, the only known algorithm for computing Gröbner basis was the Buchberger's algorithm. With FGLM the authors provided not only a way to compute LEX GB in a more efficient way, but they showed that moving from a graded monomial order to the lexicographic one is more convenient and less computationally demanding. Starting from a Gröbner basis $G_{\prec_1}$ of an ideal I w.r.t a monomial order $\prec_1$, FGLM generates a lexicographic GB for the same ideal. The idea behind FGLM is to determine a univariate polynomial with respect to the smallest variable (call it x_1). So, firstly it considers all powers of x_1 and tries to write each of them as a linear combination of the previous ones modulo the ideal $I = \langle G_{\prec_1} \rangle$. Every time it finds a power of x_1 which can be completely represented as a linear combination of previous powers of x_1, it considers this representation as a polynomial of G_{LEX} (the lexicographic Gröbner basis). Then, it takes the remaining variables as the last components of G_{LEX}.

Algorithm 7 FGLM

```
    procedure FGLM($G$)                                              ▷
        Compute $B_G$
3:      $G_{\mathsf{LEX}} = \emptyset$
        $B_{G_{\mathsf{LEX}}} = \emptyset$
        for $i = 0, 1, \ldots, |B_G|$ do
6:          $r := \overline{x_1^i}^G$
            if $\nexists (a_1, a_2, \ldots, a_{|B_{G_{\mathsf{LEX}}}|}) \mid r = \sum_{i=1}^{|B_{G_{\mathsf{LEX}}}|} a_i B_{G_{\mathsf{LEX}}}[i]$ then
                $B_{G_{\mathsf{LEX}}} = B_{G_{\mathsf{LEX}}} \cup \{x_1^i\}$
9:          else
                $G_{\mathsf{LEX}} = G_{\mathsf{LEX}} \cup \{x_1^i - r\}$
        for $i = 2, \ldots, n$ do
12:         $r := \overline{x_i}^G$
            $G_{\mathsf{LEX}} = G_{\mathsf{LEX}} \cup \{r\}$
        return $G_{\mathsf{LEX}}$
```

Algorithm 7 presents a sketch of the `FGLM` algorithm. The complexity of the procedure is given by

$$\mathcal{O}(nd_I^3)$$

where n is the number of variables and d_I is the dimension of the quotient ring basis (see Chapter 2).

3.2.2 SparseFGLM

If the given ideal is known to be in shape position, the algorithm exploits the deterministic or the probabilistic version of the Wiedemann algorithm. Although, [21] also provided a generic version of the algorithm, as shown in Algorithm 10. All the algorithms and the corresponding complexities do not take into account the complexity derived by the computation of the multiplication matrices, therefore we assume them to have already been computed in advance.

The Wiedemann algorithm Wiedemann algorithm is part of the more general category of *Krylov methods*. There exist two versions of the algorithm:

- the probabilistic algorithm (Algorithm 8) has a runtime complexity of

$$\mathcal{O}(d_I(\mathcal{Z} + \log(d_I))) \tag{3.4}$$

  operation in $\mathbb{K}$, where $\mathcal{Z}$ is the number of non-zero entries in the considered matrix. It returns, with large probability, the minimal polynomial associated to the generated sequence [43].

- the deterministic algorithm (Algorithm 9) has a runtime complexity of

$$\mathcal{O}(d_I(\mathcal{Z} + d_I \log(d_I) \log\log(d_I))) \tag{3.5}$$

operations in $\mathbb{K}$, where $\mathcal{Z}$ is still the number of non-zero entries in the considered matrix.

The probability of success of the probabilistic algorithm is given by:

$$P_q(n) = \begin{cases} (1-\frac{1}{q})^{2n} & \text{if } q \geq N \\ (1-\frac{1}{q})^{2q}(1-\frac{1}{q^2})^{n-q} & \text{if } \sqrt{n} \leq q < n \end{cases} \tag{3.6}$$

where q is the field cardinality and n is the matrix dimension (for sake of simplicity we are considering square matrices) [25].

Algorithm 8 Probabilistic Wiedemann algorithm

```
procedure ProbabilisticWiedemann($G_1$)                                      ▷
    Compute $B_{G_1}$
    Compute $T_1$
    $\vec{e} := (1,0,\dots,0)^T \in \mathbb{K}^{(D\times 1)}$
    for $i = 1,\dots,2D-1$ do
        $\vec{r_i} := T_1^T \ \vec{r_{i-1}}$
    $s := [\langle \vec{r_i}, \vec{e}\rangle \mid i = 0,\dots,2D-1]$
    $f_1 :=$ BerlekampMassey($s$)
    return $f_1$
```

Algorithm 9 Deterministic Wiedemann algorithm

```
procedure DeterministicWiedemann($G_1$)                                      ▷
    Compute $B_{G_1}$
    Compute $T_1, T_2, \dots, T_n$
    $\vec{e_1} := (1,0,\dots,0)^T, \vec{e_2} := (0,1,\dots,0)^T, \dots, \vec{e_D} := (0,0,\dots,1)^T \in \mathbb{K}^{(D\times 1)}$
    $k := 1;\ F := [];\ f := 1;\ d := 0;\ \vec{b} = \vec{e_1}; S := []$
    while $\vec{b} \neq \vec{0}$ do
        $s := [\langle \vec{e_k}, T_1^i \ \vec{b}\rangle \mid i = 0,\dots,2(n-d)-1]$
        $g :=$ BerlekampMassey($s$)
        $f := f \cdot g;\ d := \mathsf{deg}(f);\ F := F + [g];\ \vec{b} := g(T_1)\ \vec{b};\ S := S + [s]$
        $k := k+1$
    $f_1 := \prod_{\tilde{f}\in F} \tilde{f}$
    return $f_1, F, S$
```

The BMS algorithm The generalization of the Wiedemann algorithm, or better of the Berlekamp-Massey algorithm, is given by the *BMS algorithm.* BMS stands for *Berlekamp-Massey-Sakata* and it is the multidimensional version of the just mentioned algorithm. Explaining how it works is not the scope of this thesis, therefore we are going to shortly describe what is its usage in the context of Gröbner basis and polynomial system solving. Both Wiedemann algorithm, both BMS are used by `SparseFGLM` to derive the characteristic polynomials of linearly recurring sequences. When dealing with shape form ideals, we only need one of them, that is the reason why we simply use the BerlekampMassey algorithm (the 1-dimensional version of BMS). On the other hand, when we are in a more general situation, the multidimensional version is necessary to determine the sequence of characteristic polynomials from which the LEX GB will be built upon. We refer the reader to [33] for the pseudocode and a well-detailed explanation of the algorithm.

General case Algorithm 10 shows the pseudocode of `SparseFGLM` in the general case where we require the usage of the BMS procedure.

Algorithm 10 BMS-based `SparseFGLM`

```
    procedure SparseFGLM($G_1, \prec_2$)                    ▷
        Compute $B_{G_1}$
3:      Compute $T_1, T_2, \dots, T_n$
        $\vec{r} \in \mathbb{K}^{D\times 1} \xleftarrow{\mathcal{R}}$
        $\vec{u} := 0$;  $F := [1]$;  $G := []$;  $E := []$;
6:      repeat
            $e := \langle \vec{r}, T_1^{u_1} T_2^{u_2} \cdots T_n^{u_n} \vec{e} \rangle$
            $E := E + [e]$
9:          $F, G :=$ BMSupdate($F, G, \vec{u}, E$)
            $\vec{u} :=$ Next($\vec{u}, \prec_2$)
            $F :=$ Reduce($F$)
12:     until NOT Termination criteria
        if isGB($F$) then                     ▷ isGB checks if $F$ is a Gröbner basis or not
            return $F$
15:     else
            return $Fail$
```

The complexity of the generic algorithm is bounded by

$$\mathcal{O}(n_v d_I (N + \tilde{N}\overline{N} d_I))$$

where N is the maximal number of non-zero entries in the multiplication matrices $T_1, \dots, T_n$, and $\tilde{N}$ and $\overline{N}$ are the number of polynomials and the maximal term number of all polynomials in the resulting lexicographic Gröbner basis respectively.

Ideals in shape form If we know in advance that our ideal is in Shape form, we can apply the 1-dimensional version of the BerlekampMassey algorithm. In particular, Algorithms 11 and 12 show the pseudocode of `SparseFGLM` with the usage of the probabilistic and deterministic versions of Wiedemann algorithm respectively.

Algorithm 11 Probabilistic `SparseFGLM`

```
    procedure SparseFGLM($G_1, \prec_2$)                                    ▷
        $f_1 :=$ ProbabilisticWiedemann($G_1$)
3:      if deg($f_1$) $= D$ then
            $H := H_D(s)$                                                   ▷ The Hankel matrix
            for $i = 2, \dots, N$ do
6:              $\vec{b} := (\langle \vec{r_j}, T_i\, \vec{e} \rangle \mid i = 0, \dots, D-1)^T$
                $\vec{c} = (c_1, \dots, c_D)^T := H^{-1}\, \vec{b}$
                $f_i := \sum_{k=0}^{D-1} \vec{c_{k+1}}\, x_1^k$
9:          return $[f_1, x_2 - f_2, \dots, x_n - f_n]$
        else
            return $Fail$
```

Algorithm 12 Deterministic `SparseFGLM`

```
    procedure SparseFGLM($G_1, \prec_2$)                                    ▷
        $f_1, F, S :=$ DeterministicWiedemann($G_1$)                        ▷ Assume $F = [f_{1,1}, f_{1,2}, \dots, f_{1,r}]$
3:      if deg($f_1$) $= D$ then
            for $i = 1, \dots, r$ do
                $d_i :=$ deg($f_{1,i}$)
6:              for $j = 2, \dots, n$ do
                    Build the Hankel matrix $H = H_j(S)$ and $\vec{b}$ from $S$
                    $\vec{c} = (c_1, \dots, c_{d_i})^T := H^{-1}\, \vec{b}$
9:                  $f_{j,i} := \sum_{k=0}^{d_i} \vec{c_{k+1}}\, x_1^k$
            $\overline{f_1} =$ squareFree($f_1$)
            if $\overline{f_1} \neq f_1$ then
12:             Compute $\{[\overline{f_{1,j}}, x_2 - \overline{f_{2,j}}, \dots, x_n - \overline{f_{n,j}}] \mid j = 1, \dots, r\}$ such that $\overline{f_1} = \prod_{j=1}^{r} \overline{f_{1,j}}$ and $\overline{f_{1,j}}$
    are pairwise coprime.
            for $j = 2, \dots, n$ do
                $\overline{f_j} :=$ CRT($[\overline{f_{j,1}}, \dots, \overline{f_{j,r}}], [\overline{f_{1,1}}, \dots, \overline{f_{1,r}}]$)
15:         return $[\overline{f_1}, x_2 - \overline{f_2}, \dots, x_n - \overline{f_n}]$
        else
            return $Fail$
```

The complexity of `SparseFGLM` in the shape form ideal case is given by

$$\mathcal{O}(n_v \cdot d_I^{\omega} \log(d_I)) \tag{3.7}$$

operation in $\mathbb{K}$ [21], where ω is the linear algebra constant. However, following the notation for the Wiedemann algorithm, the complexity of `SparseFGLM` can also be defined in terms of the sparsity of the involved matrices:

$$\mathcal{O}(d_I(\mathcal{Z} + n_v \log(d_I))) \tag{3.8}$$

where $\mathcal{Z}$ is the number of non-zero entries [21].

3.3 Finding the variety

We are discussing the methodologies to solve systems of polynomial equations, meaning finding the variety of the ideal determined by those equations. The first step is to build or to compute a Gröbner basis starting from the original equations. Afterwards, the subsequent steps are not standard, due to the fact that the paths to follow depend on the type of systems we are dealing with. In the next paragraphs we illustrate the main ways to reach the result and how we can choose the best one.

3.3.1 Change of order

In Subsection 3.2 we already tackled the problem of converting a graded monomial ordering Gröbner basis to a lexicographic one. In particular, for cryptographic purposes, we are mainly interested in shape form ideals. By converting the basis we are generating one univariate polynomial which is of high interest in terms of computing the variety. Indeed, factorizing such polynomial immediately gives the possible values that the target variable can assume. Recall the shape of the shape form:

$$G = \{x_1 - g_1(x_n), x_2 - g_2(x_n), \dots, g_n(x_n)\}$$

where the ring of interest is $\mathbb{K}[x_1, \dots, x_n]$. Algorithm 13 recovers the variety $\mathbb{V}(G)$. More in general, even if we are not in the shape form case, converting a graded GB to a lexicographic one is still important due to the fact that it generates simpler polynomials from which it is easier to determine the variety.

Algorithm 13 Variety of shape form Gröbner basis

```
   procedure FINDVARIETY($G$)                          ▷ $G$ is in Shape form
       Variety = []
3:     Roots = $findRoots(g_n)$                        ▷ Find all the roots in $\mathbb{K}$ of $g_n$
       for $r \in$ Roots do
           $\tilde{x_i} = g_i(r)$ for $1 \leq i < n$
6:         $S = (\tilde{x_1}, \tilde{x_2}, \dots, \tilde{x_{n-1}}, r)$
           Variety = Variety + [$S$]
       return Variety
```

3.3.2 Resultant technique

Solving system of polynomials with resultants has been deeply explored in the recent years and different techniques were developed. Before going to list some of this techniques that can be used to determine the variety of an ideal, we present the notion of *resultant* and the basic ideas under the polynomial system solving.

Definition 10 (Resultant). *Let $f, g \in \mathbb{K}[x_1, x_2, \dots, x_t, y]$. The resultant of f and g with respect to the variable y, denoted as $\mathcal{R}(f, g, y)$, is defined as the determinant of the Sylvester matrix of f and*

g when considered as univariate polynomials whose coefficients are in $\mathbb{K}[x_1, \dots, x_t]$. For example, let $f = \sum_{i=0}^{m} f_i y^i$ and $g = \sum_{i=0}^{n} f_i y^i$, where $f_i, g_i \in \mathbb{K}[x_1, \dots, x_t]$, then

$$\mathcal{R}(f,g,y) = \det \begin{bmatrix} f_m & f_{m-1} & f_{m-2} & \cdots & f_0 & & & & & \\ & f_m & f_{m-1} & f_{m-2} & \cdots & f_0 & & & & \\ & & f_m & f_{m-1} & f_{m-2} & \cdots & f_0 & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & & f_m & f_{m-1} & f_{m-2} & \cdots & f_0 \\ g_n & g_{n-1} & \cdots & g_0 & & & & & & \\ & g_n & g_{n-1} & \cdots & g_0 & & & & & \\ & & g_n & g_{n-1} & \cdots & g_0 & & & & \\ & & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & g_n & g_{n-1} & \cdots & g_0 & & \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\\a\\a\end{matrix}}\right\} n \text{ times} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\\a\end{matrix}}\right\} m \text{ times} \end{matrix}$$

The Sylvester matrix has dimension $(m+n) \times (m+n)$.

The value of the resultant is non-zero if and only if two polynomials are algebraically independent. In particular, $\mathcal{R}(f,g,y)$ will be a polynomial $h \in \mathbb{K}[x_1, \dots, x_t]$, such that, if $(a_1, a_2, \dots, a_t, b)$ is a root of both f and g, then $(a_1, \dots, a_t)$ is a root of the polynomial h. Moreover, there exist two polynomials $\tilde{f}, \tilde{g} \in \mathbb{K}[x_1, \dots, x_t]$ such that $\tilde{f}f + \tilde{g}g = \mathcal{R}(f,g,y)$.

Resultants can be computed in other ways. One of them consists in using the Lagrange interpolation. Indeed, if we are able to predict the *degree* of the resultant, we can select a suitable number of points in order to uniquely determine the polynomial we are interested in. Let us quickly recall what is the Lagrange Interpolation with a simple example.

Example 5. Suppose we have an oracle that evaluates a polynomial $f \in \mathbb{K}[x]$ with $\mathsf{deg}(f) = 4$ on our inputs. Are we able to recover such polynomial by only knowing a certain number of couples (input, output)? How many couples do we need?

$\mathsf{deg}(f) = 4$ implies that we require at least $\mathsf{deg}(f) + 1 = 5$ points. Assume we have used the oracle, and we obtained couples (x_i, y_i) for $1 \leq i \leq 5$. Then, f can be uniquely determined by the Lagrange interpolation formula:

$$f(x) := \sum_{i=1}^{5} \Big(y_i \cdot \prod_{j=0, j \neq i}^{5} \frac{x - x_j}{x_i - x_j} \Big).$$

Now, let $f, g \in \mathbb{K}[x,y]$, $\mathcal{R}(f,g,y) \in \mathbb{K}[x]$ can be computed by using the Lagrange interpolation. Let us suppose that the resultant will have degree n. Pick up $n+1$ distinct values for the variable x, call them $a_1, \dots, a_{n+1} \in \mathbb{K}$. Evaluate both f and g on the chosen values. As a result we will obtain $n+1$ input-output couples from f

$$(a_1, f(a_1, y)), (a_2, f(a_2, y)), \dots, (a_{n+1}, f(a_{n+1}, y)),$$

and $n+1$ couples from g

$$(a_1, g(a_1, y)), (a_2, g(a_2, y)), \dots, (a_{n+1}, g(a_{n+1}, y)).$$

Note that the evaluations return univariate polynomials in the unknown y.

Now, for all $1 \le i \le n+1$, compute $\mathcal{R}_i(a_i) = \mathcal{R}(f(a_i, y), g(a_i, y), y) \in \mathbb{K}$. What are all those $\mathcal{R}_i$? These are the evaluations of the polynomial $\mathcal{R}(f, g, y) \in \mathbb{K}[x]$ we want to determine on the $n+1$ distinct values a_i. By using the Lagrange interpolation on the $n+1$ distinct points $(a_i, \mathcal{R}(a_i))$ we can retrieve the n-degree polynomial representing $\mathcal{R}(f, g, y)$.

Since now we are able to compute resultants, we can shortly discuss what are the two main methods that can be used to solve polynomial systems:

- **μ-Resultant**
- **Hidden variable**

Resultants and how they can be used to solve systems of multivariate polynomials are out-of the scope of this thesis. Therefore, we are going to briefly explain the **Hidden variable** method which is the more used in this context.

Hidden variable The idea is to consider one of the variables as *hidden-target variable* and the others as part of the coefficients (recall what we have done at the beginning of this subsection). By doing so, we obtain another polynomial where the *hidden-target variable* is not part of it anymore. If we have more than two polynomials, we can repeat the process several times until we obtain two polynomials in two unknowns. Computing the resultant of these two polynomials w.r.t. one of the two variables leads to a univariate polynomial which can be factorized to determine the possible values of the remaining variable. Now, with back-substitution, we can retrieve the values for the other variables.

3.3.3 Eigenvalue methods

In Chapter 2 we have discussed the concept of multiplication matrices (see Definition 15). Multiplication matrices and, in general, the determinant computation are also involved in the `SparseFGLM` algorithm which exploits the Wiedemann algorithm. Sometimes, especially for cryptographic purposes, we are not interested in finding the complete variety, but only the possible values of some chosen variables. As we already know, we can compute a multiplication matrix for each variable. Then, to discover what are the possible values it can assume, we must factorize the univariate polynomial derived from the corresponding multiplication matrix, in other words we must factorize its characteristic polynomial. But, how to compute such polynomial? We present two main ways to compute it:

- the Wiedemann algorithm we already presented in Subsection 3.2.2, which is a kind of the more general category of *Krylov methods*.
- the naive application of one of the off-the-shelf algorithms for determinant computation and a special case with the FreeLunch methodology [3].

The possibility to use the characteristic polynomial of the multiplication matrices to derive the solutions of the system of polynomials is guaranteed by Stickelberger theorem [16, Chapter 2.4].

Theorem 3 (Stickelberger Theorem). *Let $\overline{\mathbb{K}}$ be the algebraic closure of the field $\mathbb{K}$ and $I \subset \overline{\mathbb{K}}[x_1, \dots, x_n]$ be a zero-dimensional ideal. For each $i = 1, \dots, n$ and any $\lambda \in \overline{\mathbb{K}}$, the value λ is an eigenvalue of the endomorphism T_i if and only if there exist a point $a \in \mathbb{V}(I)$ with $a_i = \lambda$.*

Corollary 2. *Let $\overline{\mathbb{K}}$ be an algebraically closed finite field, I be a zero dimensional ideal in $\overline{\mathbb{K}}[x_1, x_2, \dots, x_n]$, $\prec$ a valid monomial order and T_i the multiplication matrix of the variable x_i with respect to $\prec$. The set of solutions of x_i is described by the roots of the minimal univariate polynomial in x_i defined by $det(x_i \mathbf{I}_{d_I} - T_i) \in I$, where $\mathbf{I}$ is the identity matrix.*

Given a variable x and a matrix $M \in \mathbb{K}^{n \times n}$, its characteristic polynomial is the determinant of the polynomial matrix $\det(x\mathbf{I}_n - M)$. This determinant can be computed in a variety of ways. Here, we list the currently best known algorithms:

- Labahn [30]:
  - Complexity: $\mathcal{O}(n^{\omega}\lceil s \rceil)$ field operations, where s denotes the mean of the column degrees, that is the mean of the maximum degrees (of the polynomials) within each column.
  - Prerequisite: unimodular triangularization
  - Type: deterministic.
- Storjohann [39]
  - Complexity: $\mathcal{O}(n^3 d)$ field operations, where d is a bound on the degree of the entries of M.
  - Prerequisite: weak Popov form computation.
  - Type: probabilistic
- Neiger-Pernet [31]
  - Complexity: $\mathcal{O}(n^{\omega})$ field operations.
  - Prerequisite: having a subroutine which multiplies two $n \times n$ matrices in $\mathcal{O}(n^{\omega})$ field operations.
  - Type: deterministic

The Freelunch methodology Given a matrix $M \in \mathbb{K}^{n \times n}$, assume it can be represented as a block matrix with block sizes $k \times k$ where $k|n$. If it has the following structure

$$M := \begin{bmatrix} 0 & 0 & \dots & 0 & -M_1 \\ \mathbf{I}_{k\times k} & 0 & \dots & 0 & -M_2 \\ 0 & \mathbf{I}_{k\times k} & \dots & 0 & -M_3 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \mathbf{I}_{k\times k} & -M_{n/k} \end{bmatrix}$$

then, the characteristic polynomial $p(x)$ w.r.t the variable x is the determinant

$$p(x) = \det(x\mathbf{I}_{n\times n} - M) = \pm \det(x^{n/k}\mathbf{I}_{k\times k} + \sum_{i=1}^{n/k} x^{i-1}M_i).$$

The methodology is not fundamental to understand the rest of the thesis, therefore we do not present the proof, and we refer the reader to [3] for a deeper analysis of the method. The complexity of the methodology is given by the application of the algorithms presented above to the $k \times k$ matrix $x^{n/k}\mathbf{I}_{k\times k} + \sum_{i=1}^{n/k} x^{i-1}M_i$.

As far as we have computed the required univariate polynomials, we can factorize them and, by Corollary 2, determine the admissible values for the target variables. Sometimes, we don't need to compute a univariate polynomial for each variable, but we can exploit additional structures and cross-dependencies between variables to limit the amount of required univariate polynomials to the necessary variables only. The remaining ones could be found by back-substitution, gcd computation or linear algebra methods (e.g. Gaussian elimination).

3.4 On the computation of the multiplication matrices

In Section 3.3 we presented different ways of obtaining the solutions of a system of polynomials. Moreover, almost all the described methodologies make use of the multiplication matrices (see Definition 15). But, what is their contribution to the algorithms' complexity? In this section we present a brief discussion about the computation of multiplication matrices and how it can be improved.

Let us define the context. Given a set of polynomials $G = \{f_1, \dots, f_s\} \subset \mathbb{K}[x_1, \dots, x_n]$ such that the ideal $I = \langle f_1, \dots, f_s \rangle \neq \langle 1 \rangle$ is a zero-dimensional ideal and G is its reduced Gröbner basis, we can compute the standard basis B_I for the quotient ring $\mathbb{K}[x_1, \dots, x_n]/I$. Moreover, we know that the standard basis has the following structure:

$$B_I := \{X^\alpha \mid X^\alpha \notin \langle LM(I) \rangle\} = \{X^\alpha = \prod_{i=1}^{n} x_i^{\alpha_i} \mid X^\alpha \notin \langle LM(G) \rangle\}.$$

Due to ease of notation, denote as ϵ_i the i-th element of the quotient ring standard basis.

The **multiplication matrix** T_j of x_j is defined by the square matrix whose i-th vector is represented by the coefficients of the monomials of the basis B_I contained in the reduction of $x_j\epsilon_i$ modulo the Gröbner Basis G. Therefore, the complexity of computing a multiplication matrix depends on the number of reductions we need to perform. Due to ease of notation, let us fix a generic monomial ordering $\prec$ and define the following notations:

- $in_\prec(I) = \{LT(f) \mid f \in I\}$ denotes the initial ideal of I w.r.t. $\prec$
- $E_\prec(I) = \{LT(f) \mid f \in G\}$ denotes the stair of I w.r.t. $\prec$ (that is a minimal set of generators of $in_\prec(I)$)
- $NF_\prec(f) = \sum_{i=1}^{t} c_i\epsilon_i$, where $1 \leq t \leq |B_I|$, $c_i \in \mathbb{K}$ and $\epsilon_i \in B_I$, denotes the normal form of the polynomial f.

To reduce the number of reductions it is straightforward to define the following criterion.

Proposition 4. *Two simple checks can be done in order to avoid useless reductions:*

i. if $x_j\epsilon_i \in B_I$, then $\overline{x_j\epsilon_i}^G = x_j\epsilon_i$.

ii. if $x_j\epsilon_i \in E_\prec(I)$, then, $\overline{x_j\epsilon_i}^G = g - x_j\epsilon_i$ where $g \in G | LT(g) = x_j\epsilon_i$.

By doing so, we need to perform the reduction only for those monomials $x_j\epsilon_i$ which are not in the quotient ring basis or in the leading terms of the reduced Gröbner basis. Other improvements can be done by recycling previous computations.

Example 6. Let x, y, z variables and let $G = \{x^2 + y, y^2 + z, z^2\}$ be a reduced Gröbner basis. The quotient ring basis is then $1, x, y, z, xy, xz, yz, xyz$. If we are computing the multiplication matrix w.r.t. the variable y, we are required to reduce the polynomials $y, xy, y^2, yz, xy^2, xyz, y^2z, xy^2z$. As regards y, xy, yz, xyz, we can easily apply point *i.* of criterion 4. As regards y^2, we can apply point *ii.* of the same criterion.

The remaining monomials must be reduced. But, is it possible to reuse some previous reductions? Suppose we are computing the reduction of xy^2. We have already computed the reduction of y^2 by following point *ii.* of criterion 4. In particular, we obtained $y^2 - (y^2 + z) = -z$. Therefore, reducing xy^2 means reducing $-xz$ by substitution, and luckily xz is a member of the quotient ring basis for which we can apply point *i.* of the criterion. Of course, this is not always the case. Sometimes, additional reductions are needed. Anyway, combining substitutions and reductions is a good way to reduce the number of computations required by the algorithm.

As a consequence of the previous example, we can define a second criterion.

Proposition 5. *Let $F = \{x_j\epsilon_i | 1 \le i \le |B_I| \ \wedge \ 1 \le j \le n\} \setminus B_I$ be the border of the quotient ring basis. Recall the example above, xy^2 is an element of the border. Suppose we have to reduce a monomial $t = x_j\epsilon_i$. If Criterion 4 cannot be applied, we are in the following case:*

- *$u = x_k\tilde{u}$ where $\tilde{u} \in F$ and $NF_\prec(\tilde{u}) = \sum_{i=1}^{t} c_i\epsilon_i$ where $\tilde{u} >_\prec \epsilon_t$. This means that u is a multiple of a monomial $\tilde{u}$ which has been already reduced. Indeed,*

$$NF_\prec(u) = NF_\prec(x_k NF_\prec(\tilde{u})) = NF_\prec(x_k \sum_{i=1}^{t} c_i\epsilon_i) = NF_\prec(\sum_{i=1}^{t} x_k c_i\epsilon_i) = \sum_{i=1}^{t} c_i NF_\prec(x_k\epsilon_i).$$

However, applying the substitution technique we have seen in the previous example and formally defined in criterion 5 makes the computation not fully parallelizable. Hence, depending on the available computational power we can decide when it is better parallelizing the reductions or applying the substitution technique. Hybrid approaches can be used as well.

4

Gröbner basis cryptanalysis

Algebraic attacks are the most decisive in determining the security of Arithmetization-Oriented primitives. Moreover, different strategies can be applied depending on the context and on the structure of the analysed primitive. In this chapter we focus our attention on the application of the Gröbner basis methodology and how it is possible to improve the complexity of the attack and define the security of the primitives. However, for sake of completeness, we provide a brief description of other two algebraic cryptanalysis methodologies in Appendix B.

In Gröbner basis (GB) cryptanalysis a cryptographic primitive is represented as a system $\mathcal{F}$ of polynomial equations with a certain number of variables. Depending on the primitive description, different approaches can be taken towards constructing the polynomial system. For finding solutions to that system, Gröbner basis method is applied. Since cryptographic primitives are typically constructed over a finite field $\mathbb{K}$, the polynomial system representing such a primitive will always have a finite number of solutions in $\mathbb{K}$. In algebraic geometry language this result in zero-dimensional ideals. Solving system of polynomials with GB means finding at least one solution to the problem. This is usually divided into three main computations:

1) A GB G of the ideal $I = \langle \mathcal{F} \rangle$ is computed with respect to a chosen monomial ordering (usually Degree Reverse Lexicographic) using one of the known algorithms, e.g. F4 [22], F5 [19] etc. We will denote the complexity of this part $\mathcal{C}_{GB}$.

2) One or more univariate polynomials are obtained towards solving the system of polynomials in G. The complexity of this computation is denoted as $\mathcal{C}_{univar}$.

   Remark 1. A common way to achieve this is converting the computed GB G (from a graded monomial ordering) to G_{LEX} by using a conversion algorithm such as FGLM [20] for zero-dimensional ideals [1]. The FGLM conversion generates a *reduced* Gröbner Basis G_{LEX} which contains a (unique) univariate polynomial. Under the condition that I is a radical ideal or I follows Shape lemma, FGLM outputs G_{LEX} having the shape form (Definition 16).

3) Factorize that univariate polynomial by using an off-the-shelf polynomial factoring algorithm to

[1]Gröbner Walk [14] for general purposes

find at least one root (solution) of that representative variable. The complexity of this part is denoted as $\mathcal{C}_{root}$.

Remark 2. If the ideal is in Shape form, the output of the FGLM algorithm has the convenient structure given in Definition 16. If so happens, back-substitute that solution into the other equations to find the values for the remaining variables. Otherwise, we have no guarantees that we are able to find the other roots by simple back-substitution. For example, additional applications of the factorization algorithm could be required.

The complexity of a GB cryptanalysis depends on the 3 computations described above: finding a Gröbner basis $G_{\prec}$, obtaining a univariate polynomial in GB, and factorizing or root finding of a univariate polynomial from G_{LEX}.

4.1 Computing a Gröbner basis

Determining the complexity of GB computation, particularly providing a good theoretical bound is a non-trivial task in the context of GB cryptanalysis. Only a good theoretical bound on $\mathcal{C}_{GB}$ can be relied upon to compute the security parameter(s) based on this part of the GB cryptanalysis. By denoting as d_{solv} the solving degree, which is the maximum degree reached during the Gröbner basis computation, the runtime complexity of common Gröbner basis algorithms such as F4 or F5 is bounded [2] by

$$\mathcal{O}\left(n_e d_{solv}\binom{n_v + d_{solv}}{d_{solv}}^{\omega}\right)$$

operations in $\mathbb{K}$ (see Chapter 3). It has to be considered that the given bound is loose, meaning that it is too high compared to the experimental results. Due to this issue, from a defensive (or a designer's) point of view, we should not consider it as a metric to define the number of rounds of the analysed primitive. Moreover, this bound is generic and thus, it does not take any structure of the underlying polynomial into account. In Chapter 5 we provide a more concrete bound on the complexity of computing the GB corresponding to the ideal generated by the `Anemoi` polynomial system.

For many systems, the value of d_{solv} can be approximated by computing the degree of semi-regularity of the homogenization of the system [11]. Indeed, denote as F the system of polynomials and as F^H the corresponding homogenized system, [11] shows that

$$\texttt{solve.deg}(F^H) = \texttt{max.GB.deg}(F^H) \geq \texttt{solve.deg}(F) \geq \texttt{max.GB.deg}(F)$$

when DRL monomial order is used, where `solve.deg` returns the solving degree and `max.GB.deg` returns the maximum degree encountered during the DRL Gröbner basis computation (which is the correct value of d_{solv}). In addition, let $f_1, \ldots, f_s$ be the set of polynomials defining an ideal I and f_i^{top} denote the homogeneous component of f_i of largest degree, that is the sum of its terms whose total degree is the largest one, *i.e.* if $f_i = \sum_{j=0}^{n} c_j x_1^{\alpha_{j,1}} \ldots x_n^{\alpha_{j,n}}$,

$$f^{top} = \sum_{j \in \{i : (\alpha_{i,1}, \ldots, \alpha_{i,n}) = \mathsf{deg}(f)\}} c_j x_1^{\alpha_{j,1}} \ldots x_n^{\alpha_{j,n}}.$$

We can define

$$F^{top} = \{f_1^{top}, \ldots, f_s^{top}\}.$$

Commonly, a lower bound on the value of d_{solv} can be obtained by computing the degree of semi-regularity of the set F^{top} defined above.

However, for practical examples, we can check that bound by directly computing the value d_{solv} and obtain a relation on its growth with respect to the number of rounds of the primitive we are analysing.

4.2 Obtaining univariate polynomials

There are different ways to find the target univariate polynomial as we have seen in Chapter 3. Here, we briefly discuss the common ones namely, basis conversion and determinant computation. As already discussed, there are two main ways to obtain univariate polynomials e.g. *FGLM* and *Eigenvalue* methods.

4.2.1 FGLM

The common way of computing the univariate polynomial, which can be used to obtain the variety of the ideal, is to convert G_{DRL} to G_{LEX}. Owing to the zero-dimensional ideal, one of the equations in G_{LEX} is univariate. From it, we can perform the third step and obtain the variety of the original ideal. For zero dimensional ideals, the change of monomial order is typically done by using the FGLM algorithm (see Subsection 3.2.1) which has complexity

$$\mathcal{O}(n_v \cdot d_I^3)$$

operations in $\mathbb{K}$, where n_v is the number of variables in R and $d_I = dim_{\mathbb{K}}(R/I)$ is the dimension of the quotient ring R/I. By using fast linear algebra and taking into account that our polynomials are sparse, by using `SparseFGLM` the bound can be improved to a runtime complexity of

$$\mathcal{O}(n_v \cdot d_I^{\omega} \log(d_I)) \tag{4.1}$$

operation in $\mathbb{K}$ [21], where ω is the linear algebra constant which theoretically is bounded by $2 \leq \omega \leq 2.3727$ [44], but practically the best known procedure is the Strassen algorithm which sets the value of ω to 2.8074. By applying that algorithm we obtain a Gröbner basis G_{LEX} containing a univariate polynomial in the smallest variable. In addition, as seen in Subsection 3.2.2, the complexity of `SparseFGLM` can also be defined in terms of the sparsity of the involved matrices:

$$\mathcal{O}(d_I(\mathcal{Z} + n_v \log(d_I))) \tag{4.2}$$

where $\mathcal{Z}$ is the number of non-zero entries [21].

4.2.2 Eigenvalue method

However, the eigenvalue method, which directly exploits the multiplication matrices of the target variables, can be used instead of FGLM. Instead of using `SparseFGLM` we can compute the multiplication matrix of the l variables we are interested in and, for each of them, compute the characteristic polynomial. This operation can be performed by computing the determinant of a matrix or by computing the polynomial of a linearly recurring sequence, as previously described in Subsection 3.3.3.

Determinant computation Let x_i be the target variable, that is the variable which we want to obtain a univariate polynomial for, and let T_i be the associated multiplication matrix defined as in Definition 15, the corresponding univariate polynomial can be defined by computing $det(x_i\mathbf{I}_{d_I} - T_i)$, whose roots, by Corollary 2, are the possible values for x_i, that are the solutions we are interested in. To compute such determinant we can apply one of the methods presented in 3.3.3. Suppose we decide to use the Labahn et al. algorithm [30] whose complexity is given by

$$\mathcal{O}(d_I^{\omega}\lceil s \rceil)$$

operations in $\mathbb{K}$, where s is the mean of the column degrees. As a consequence, if we aim at finding l univariate polynomials, the overall complexity of $\mathcal{C}_{univar}$ becomes

$$\mathcal{O}(l \cdot d_I^{\omega}).$$

Linearly recurring sequence Let x_i be the target variable and let T_i be the associated multiplication matrix, we can compute a linearly recurring sequence to determine, thanks to the Berlekamp-Massey algorithm [5], its minimal polynomial [21, 25]. There exist two versions of the Wiedemann algorithm as we have seen in Subsection 3.2.2. Suppose we decide to use the probabilistic version whose complexity is

$$\mathcal{O}(d_I(\mathcal{Z} + \log(d_I))) \tag{4.3}$$

operation in $\mathbb{K}$, where $\mathcal{Z}$ is the number of non-zero entries in T_i. Due to the fact that we need to apply the method for all the l target variables, the overall complexity is given by:

$$ld_I(\mathcal{Z} + \log(d_I)). \tag{4.4}$$

SparseFGLM or Wiedemann algorithm For cryptanalysis purposes, applying FGLM algorithms is useful when the output is a basis in Shape Form, and we need to find the complete variety of the system. Therefore, proving that the ideal we are analysing has a basis in Shape form is necessary to fully exploit the "nice" structure that can be obtained and thus, to decide whether it is convenient to apply such algorithms or not. Moreover, we are dealing with sparse polynomials, meaning that it makes sense to apply `SparseFGLM` and Wiedemann algorithm. As we have studied in Subsection 3.2.2, Wiedemann algorithm is the starting step of the `SparseFGLM` algorithm. Therefore, an immediate question would be: when is it better to use directly Wiedemann algorithm or vice versa?

Generically, it is worth using Wiedemann algorithm when we don't need to compute the solutions for all the variables involved in a system. For example, given an algebraic model for a hash function, cryptographers are usually not interested in the values assumed by the inner variables, but only on those of the input or output variables. On the other hand, due to the fact that we can represent the complexity of `SparseFGLM` in terms of the number of non-zero entries as done for the Wiedemann algorithm, we can directly compare the two methods and define when one performs extremely better than the other. With Wiedemann algorithm we find a univariate polynomial for each of the variables we are interested in (say we want to find the solutions of k out of n variables). Therefore, the overall complexity is k times the complexity of computing the univariate polynomial: $\mathcal{O}(kd_I(\mathcal{Z} + \log(d_I)))$.

With `SparseFGLM` the complexity is given by the computation of **one** univariate polynomial (for the smaller variable with respect to a certain monomial order) plus the conversion of the system to a LEX Gröbner basis in Shape form: $\mathcal{O}(d_I(\mathcal{Z} + n\log(d_I)))$.

We want to discover when the Wiedemann algorithm performs better than `SparseFGLM`. We can temporarily remove the asymptotic notation and define the following inequality:

$$d_I(\mathcal{Z} + n\log(d_I)) - kd_I(\mathcal{Z} + \log(d_I)) \geq 0$$

Let define $\delta = n - k$ and rewrite the above equation accordingly.

$$\begin{aligned}
d_I(\mathcal{Z} + (\delta + k)\log(d_I)) - kd_I(\mathcal{Z} + \log(d_I)) &\geq 0 \\
d_I\mathcal{Z} + \delta d_I\log(d_I) + kd_I\log(d_I) - kd_I\mathcal{Z} - kd_I\log(d_I) &\geq 0 \\
\delta d_I\log(d_I) + (1-k)d_I\mathcal{Z} &\geq 0 \\
\delta\log(d_I) + (1-k)\mathcal{Z} &\geq 0 \\
(n-k)\log(d_I) &\geq (k-1)\mathcal{Z}
\end{aligned}$$

From that result we can conclude that:

- if $n = k = 1$ the complexities are the same
- if $\mathcal{Z} \leq \frac{n-k}{k-1}\log(d_I)$, repeating Wiedemann algorithm performs better that `SparseFGLM`.

Note that we assume the sparsity of the multiplication matrices to be almost the same for each variable. This is not always the case. Therefore, this is another check that has to be done to choose the best algorithm.

4.3 Factoring polynomials or root finding

As for the polynomial factorization, there are multiple choice, some of them are:

- Cantor-Zassenhaus [13] [18]
- Berlekamp Algorithm [4]
- Kaltofen-Shoup probabilistic algorithm [27].

Moreover, some criteria can be used to choose the best one. As an example, which often holds in the context of cryptanalysis, if we are dealing with finite fields the latter one gives the best complexity:

$$d^{1.815}\log(q) \tag{4.5}$$

where d is the degree of the univariate polynomial h we have obtained from the previous step and q is the field characteristic. Owing to zero-dimensional ideals, the degree of the univariate polynomial will be equal to the dimension of the quotient space R/I: $\mathsf{deg}(h) = d_I$. From an experimental point of view, the best algorithm can vary depending on the type of polynomial and on the implementation [36] [37].

5 Application to ANEMOI

Anemoi is a family of ZK-friendly permutations that can be used to construct efficient hash functions and compression functions [8]. This permutation operates on F_q^{2l} for any field size q which is either a prime number of a power of two and for any positive integer l. To define the structure of each `Anemoi` round we organize the internal state into two vectors $X \in F_q^l = (x_1, \ldots, x_l)$ and $Y \in F_q^l = (y_1, \ldots, y_l)$.

5.1 The primitive

We provide a brief description of the main components of `Anemoi` that are important to follow our GB cryptanalysis.

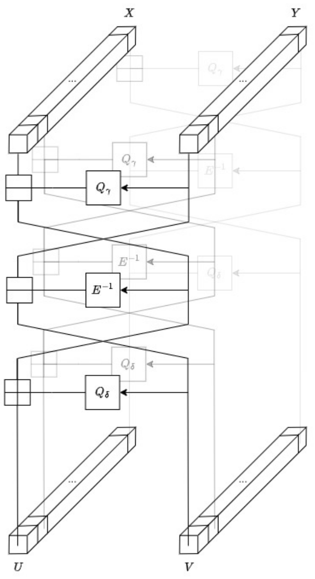


Figure 5.1: Non-linear layer of `Anemoi`$(l > 1)$

- **Linear Layer**: The `Anemoi` state consists of $2l$ elements $(X, Y) \in \mathbb{F}_q^l \times \mathbb{F}_q^l$ where $l \geq 1$. After constant addition to the state, linear transformations M_l and $M_l \circ \rho$ are applied to the two halves X and Y respectively. The permutation ρ is defined as a circular shift by one position, meaning that the first entry is moved to the last position while shifting the other ones to the previous position:

$$\rho(i) := (i-1) \mod l \quad \text{for } i \in \{0,\dots,l-1\}.$$

On the output of the linear transformations, a pseudo-hadamard transform denoted as P is applied to the vector (X,Y). Examples of matrices M_l are,

$$M_1 = \begin{bmatrix}1\end{bmatrix} \quad M_2 = \begin{bmatrix}1 & g \\ g & g^2+1\end{bmatrix} \quad M_3 = \begin{bmatrix}g+1 & 1 & g+1 \\ 1 & 1 & g \\ g & 1 & 1\end{bmatrix} \quad M_4 = \begin{bmatrix}1 & g^2 & g^2 & g+1 \\ g+1 & g^2+g & g^2 & 2g+1 \\ g & g+1 & 1 & g \\ g & 2g+1 & g+1 & g+1\end{bmatrix}$$

For our analysis, *we will denote as $\mathcal{L}$ the block of operations composed by the round constant addition and the linear transformations.*

- **Non-linear layer**: Let $Q_\gamma : \mathbb{F}_q \to \mathbb{F}_q$, $Q_\delta : \mathbb{F}_q \to \mathbb{F}_q$ be low-degree polynomials and let $E : \mathbb{F}_q \to \mathbb{F}_q$ be a power map inducing the permutation over $\mathbb{F}_q$, that is $E = x^\alpha$ with $\alpha \geq 3$ coprime with $\varphi(q)$. The non-linear layer is a so-called open Flystel, denoted as H, applied component-wise to X and Y (Figure 5.2). Q_γ and Q_δ are defined as follows:

$$Q_\gamma = \begin{cases}\beta x^2 + \gamma & \text{if } q > 2 \text{ is prime} \\ \beta x^3 + \gamma & \text{if } q = 2^n\end{cases} \qquad Q_\delta = \begin{cases}\beta x^2 + \delta & \text{if } q > 2 \text{ is prime} \\ \beta x^3 + \delta & \text{if } q = 2^n\end{cases}$$

where common values for β,γ and δ are respectively g (generator of the multiplicative subgroup of the field F_q), 0 and $g^{-1} \mod q$.

The `Anemoi` permutation on inputs $(x_{1,0},\dots,x_{l,0},y_{1,0},\dots,y_{l,0})$ (where $w_{i,j}$ means the i-th input w of the j-th round) can be summarized by the following function:

$$\begin{aligned}\texttt{Anemoi}_{q,\alpha}(x_{1,0},\dots,x_{l,0},y_{1,0},\dots,y_{l,0}) &= \\ L \circ R_N \circ R_{N-1} \circ \dots \circ R_1(x_{1,0},\dots,x_{l,0},y_{1,0},\dots,y_{l,0}) &= \\ (x_{1,N+1},\dots,x_{l,N+1},y_{1,N+1},\dots,y_{l,N+1})\end{aligned} \tag{5.1}$$

Flystel evaluation and verification in detail

Evaluation phase Let x_i, y_i the inputs to the open Flystel procedure that we denoted as H. The output of H is:

$$\begin{bmatrix}u_i \\ v_i\end{bmatrix} = H\left(\begin{bmatrix}x_i \\ y_i\end{bmatrix}\right) = \begin{bmatrix}x_i - Q_\gamma(y_i) + Q_\delta(y_i - E^{-1}(x_i - Q_\gamma(y_i))) \\ y_i - E^{-1}(x_i - Q_\gamma(y_i))\end{bmatrix}$$

Verification phase It is the corresponding counterpart of the open Flystel procedure, and it is called *closed* Flystel (denoted as V). It is defined in order to verify that $(u_i, v_i) = H(x_i, y_i)$. That operation

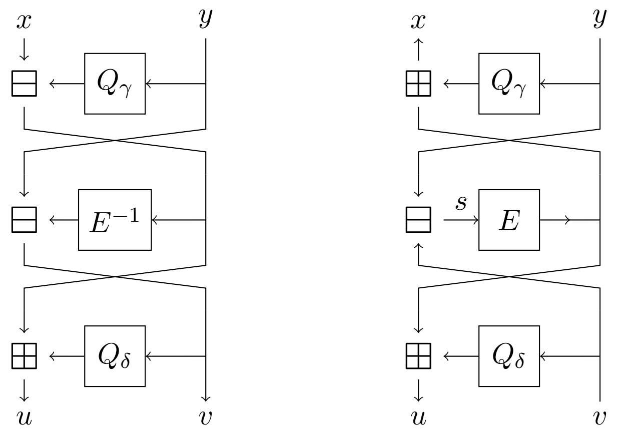


Figure 5.2: Flystel evaluation (left) and verification (right) circuit representations in `Anemoi` [29].

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	21	21	20	19	37	37	36	35
$l = 2$	14	14	13	13	22	22	21	21
$l = 3$	12	12	12	11	17	17	17	17

Table 5.1: Number of rounds for `Anemoi` [8]

is equivalent to verify that $(x_i, u_i) = V(y_i, v_i)$. The output of V is:

$$\begin{bmatrix} x_i \\ u_i \end{bmatrix} = V\left(\begin{bmatrix} y_i \\ v_i \end{bmatrix}\right) = \begin{bmatrix} Q_\gamma(y_i) + E(y_i - v_i) \\ Q_\delta(v_i) + E(y_i - v_i) \end{bmatrix}$$

Refer to Figure 5.2 for the meaning of these formulas.

5.2 `Anemoi` rounds

In the `Anemoi` proposal, the authors provide different number of rounds depending on the choice of α, the number of branches $(2l)$ and the target security levels. These values are summarized in Table 5.1.

5.3 Modes of operation

The designers proposed two modes of operation depending on the possible usage of the primitive:

- sponge mode: to turn the primitive into a hash function
- JIVE: to turn the primitive into a compression function

The sponge mode is convenient to emulate a random oracle. The structure is given in Figure 5.3. In our setting the permutation function f is exactly `Anemoi`,and it is defined over $\mathbb{K}^{r+c}$ where r stands for the *rate*, that is the size of the outer part, and c is the capacity, that is the size of the inner part.

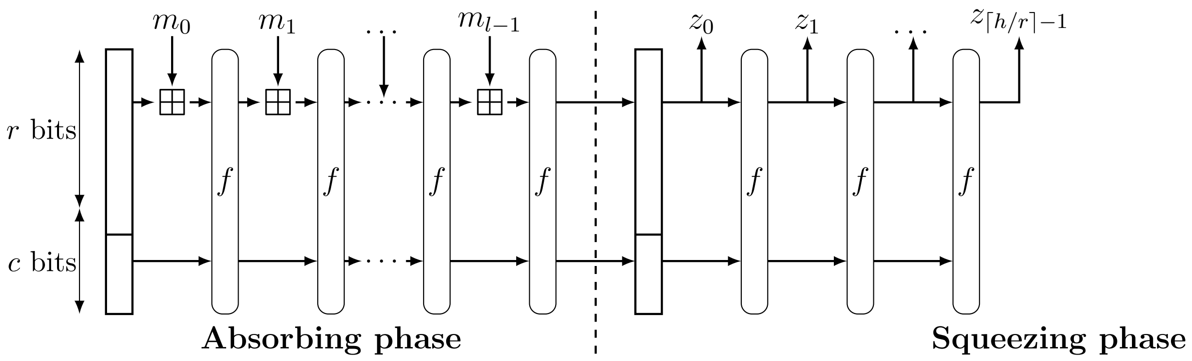


Figure 5.3: Sponge mode: f is the permutation function, r stands for the rate, c for the capacity. Image taken and modified from [26].

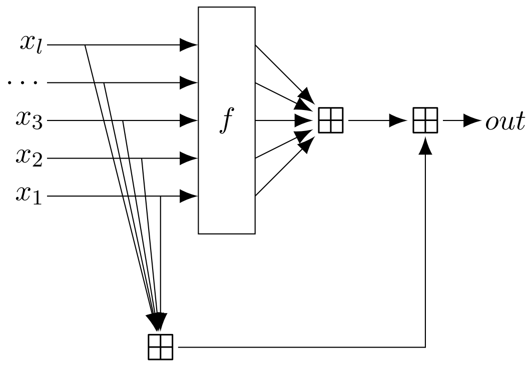


Figure 5.4: JIVE mode: f is the permutation function, $x_1, \ldots, x_l$ are the inputs and `out` is the output [8].

The JIVE mode seems a derivation of the Davis-Meyer mode, but it has to be still analysed from a security point-of-view. The reason under the JIVE mode is the usage of `Anemoi` as a compression function. The construction is given in Figure 5.4.

5.4 Modelling phase

In this section, we describe the polynomial representation of `Anemoi` for $l \geq 1$. This polynomial representation (or modelling) play important role in our GB cryptanalysis. More specifically, we describe how we at first adaptively extend the PCICO and FCICO modeling for more than two branches, and then propose a new ACICO modelling for our GB analysis.

A variable x involved in the modelling are indexed as $x_{i,j}$ where i and j correspond to branch number and round number respectively. For ease of notation we use $\mathbf{x}_a$ to denote all the variables $x_{i,a}$ for $1 \leq i \leq l$ from round a. Moreover, we use $\mathtt{deg}(\mathbf{x}_a) = \{\mathtt{deg}(x_{1,a}), \ldots, \mathtt{deg}(x_{l,a})\}$ to identify the degrees of the variables $\mathbf{x}_a$.

The so-called CICO (*Constrained Input Constrained Output*) problem in algebraic modelling and analysis of cryptographic primitives is given as following.

Definition 1 (CICO problem). *Let $F : \mathbb{F}_p^t \rightarrow \mathbb{F}_p^t$ be a function and let $u < t$ be an integer. The CICO problem consists of finding $x, y \in F_p^u$ such that*

$$F(x||0) = (y||0).$$

For Anemoi, [7] suggests fixing the first l inputs and outputs to zero. Therefore, the $CICO$ problem for `Anemoi` is represented by

$$\texttt{Anemoi}(0||\mathbf{y}_0) = (0||\mathbf{y}_{out}) \tag{5.2}$$

and we aim to find values for the variables $\mathbf{y}_0$.

Previous works introduced two main models to describe the `Anemoi` function. In particular, [7] proposed two models: F_{CICO} which represents each round with two equations whose unknowns are the inputs of the main function and P_{CICO} which represents each round with a single equation. [7] shows the analysis of F_{CICO}, whilst [29] recently shows the security analysis of P_{CICO} and its differences with respect to the first one. Although, those models were not extended to more than 2 branches. To improve the GB analysis to more than 2 branches, we propose a new model which we denote as A_{CICO}.

Let $(\mathbf{x}_0, \mathbf{y}_0) = (x_{1,0}, \dots, x_{l,0}, y_{1,0}, \dots, y_{l,0})$ be the inputs to the `Anemoi` permutation and $(\mathbf{x}_{r-1}, \mathbf{y}_{r-1}) = (x_{1,r-1}, \dots, x_{l,r-1}, y_{1,r-1}, \dots, y_{l,r-1})$ be the inputs to the r-th round for $1 \le r \le N$. Moreover, let $s_{i,r}$ for $1 \le i \le l$ model the input to the function E within the Flystel verification.

For ease of notation, we denote as $\mathcal{L}$ the application of the `Anemoi` linear layer at round r. In particular

$$\begin{bmatrix} \mathbf{L}_r^{(1)} \\ \mathbf{L}_r^{(2)} \end{bmatrix} = \mathcal{L}_r(\mathbf{x}_{r-1}, \mathbf{y}_{r-1}) := \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \left(M_l \begin{bmatrix} x_{1,r-1} + c_{1,r-1} & y_{2,r-1} + c_{2,r-1} \\ \cdots & \cdots \\ x_{l-1,r-1} + c_{l-1,r-1} & y_{l,r-1} + c_{l,r-1} \\ x_{l,r-1} + c_{l,r-1} & y_{1,r-1} + c_{1,r-1} \end{bmatrix} \right)^T \tag{5.3}$$

where c_i and d_i are the round constants. Note that the $\mathbf{y} + \mathbf{c}$ in eq. (5.3) is written as the output of $\rho(\mathbf{y} + \mathbf{c})$. Here ρ is a (linear) permutation as described in section 5.1. Furthermore, we use $L_{i,r}^{(j)}$ to denote the i-th entry of the j-th vector of the output matrix generated by the application of the linear layer at the round r. Sometimes, when we want to specify what are the inputs to the linear layer $\mathcal{L}$ and then, what are the variables which the output depends on, we use $\mathcal{L}_{i,r}^{(j)}(\mathbf{x}, \mathbf{y}) = L_{i,r}^{(j)}$.

5.4.1 PCICO Model

Polynomials representation

For every $1 \le r \le N$, let $\mathbf{L}_r^{(1)}$ and $\mathbf{L}_r^{(2)}$ be the inputs to the non-linear layer H in the r-th round. Notice that this operation is performed for each couple $(\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)})$ for $1 \le i \le l$. It outputs the functions $x_{i,r}, y_{i,r}$ where:

$$\begin{bmatrix} x_{i,r} \\ y_{i,r} \end{bmatrix} := H(\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)}) = \begin{bmatrix} \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) + Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) \\ \mathbf{L}_r^{(2)} - s_{i,r} \end{bmatrix}$$

Obviously, $\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)} \in \mathbb{K}[\mathbf{x}_0, \mathbf{y}_0, \mathbf{s}_1, \dots, \mathbf{s}_{r-1}]$ and $x_{i,r}, y_{i,r} \in \mathbb{K}[\mathbf{x}_0, \mathbf{y}_0, \mathbf{s}_1, \dots, \mathbf{s}_r]$ for $1 \le r \le N$. Due to CICO, i.e. fixing $x_{i,0} = 0$ for all $1 \le i \le l$, we get $\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)} \in \mathbb{K}[\mathbf{y}_0, \mathbf{s}_1, \dots \mathbf{s}_{r-1}]$ and $x_{i,r}, y_{i,r} \in \mathbb{K}[\mathbf{y}_0, \mathbf{s}_1, \dots, \mathbf{s}_r]$.

For each couple $(\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)})$, let

$$\begin{bmatrix} x_{i,r} \\ y_{i,r} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) + Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) \\ \mathbf{L}_r^{(2)} - s_{i,r} \end{bmatrix} \tag{5.4}$$

be the Flystel evaluation modelled following the above statements and let

$$\begin{bmatrix} L_{i,r}^{(1)} \\ x_{i,r} \end{bmatrix} := V(L_{i,r}^{(2)}, y_{i,r}) = \begin{bmatrix} Q_\gamma(L_{i,r}^{(2)}) + E(s_{i,r}) \\ Q_\delta(y_{i,r}) + E(s_{i,r}) \end{bmatrix} \tag{5.5}$$

be the corresponding Flystel verification. The $x_{i,r}$ output of both the procedures must be the same. Then, the following equality gives us the possibility to represent the Flystel evaluation of the i-th couple at the r-th round with a single equation.

$$\begin{aligned} \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) + Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) &= Q_\delta(y_{i,r}) + E(s_{i,r}) \\ \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) + Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) &= Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) + E(s_{i,r}) \\ \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) &= E(s_{i,r}) \\ E(s_{i,r}) - \mathbf{L}_r^{(1)} + Q_\gamma(\mathbf{L}_r^{(2)}) &= 0 \end{aligned} \tag{5.6}$$

Let $p_{i,r}$ be the equation 5.6 corresponding to the evaluation of the couple $(\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)})$. Using that definition we can model every round $1 \leq r \leq N$ with l equations $p_{1,r}, p_{2,r}, \dots, p_{l,r}$ where $p_{i,r} \in \mathbb{K}[\mathbf{y}_0, \mathbf{s}_1, \dots, \mathbf{s}_{r-1}, s_{i,r}]$. In this way we obtain a total of $N \times l$ equations.

At the end, after the last rounds, the linear layer L is applied again. Therefore, by applying the CICO output constraints, which set the first l outputs to 0, we can define the following equation:

$$x_{i,N+1} := L(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{y}_1, \dots, \mathbf{y}_N) = 0 \tag{5.7}$$

for all $1 \leq i \leq l$ where $x_{i,N+1} \in \mathbb{K}[\mathbf{y}_0, \mathbf{s}_1, \dots, \mathbf{s}_N]$. With the last linear layer application we obtain other l equations.

Thanks to that, we can define the P_{CICO} model for arbitrary values of l.

Definition 2 (**P_{CICO} model for arbitrary l**)**.** *An algebraic model of the* `Anemoi` *permutation $A_\pi : \mathbb{K}^{2l} \to \mathbb{K}^{2l}$ applied for N rounds, under the CICO constraints in Equation 5.2, is given by the system*

$$P_{CICO} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_N, \boldsymbol{x}_{N+1}\}$$

where $P_{CICO} \subset \mathbb{K}[\boldsymbol{y}_0, \boldsymbol{s}_1, \dots, \boldsymbol{s}_N]$ and $p_{i,r}$ defined as in Equation 5.6 and $x_{i,N+1}$ as in Equation 5.7. The generated system contains $lN + l$ equations in $lN + l$ variables.

Equations degree analysis for $q = p$ prime

To make a complete analysis of the P_{CICO} model, we give a detailed inspection of the degrees of the polynomials we defined in Subsection 5.4.1 with respect to a field of characteristic p prime for which $\mathsf{deg}(Q_\gamma) = \mathsf{deg}(Q_\delta) = 2$.

Let consider the equation $x_{i,r}$ given in Equation 5.4:

$$\begin{aligned} x_{i,r} &= \mathbf{L}_r^{(1)} - Q_\gamma(\mathbf{L}_r^{(2)}) + Q_\delta(\mathbf{L}_r^{(2)} - s_{i,r}) \\ x_{i,r} &= \mathbf{L}_r^{(1)} - \beta(\mathbf{L}_r^{(2)})^2 - \gamma + \beta(\mathbf{L}_r^{(2)})^2 + \beta s_{i,r}^2 - 2\beta\mathbf{L}_r^{(2)} s_{i,r} + \delta \\ x_{i,r} &= \mathbf{L}_r^{(1)} + \beta s_{i,r}^2 - 2\beta\mathbf{L}_r^{(2)} s_{i,r} - \gamma + \delta \end{aligned} \tag{5.8}$$

We can notice that the leading terms of Q_γ and Q_δ cancel out, meaning that, since $\mathbf{L}_r^{(1)}, \mathbf{L}_r^{(2)} \in \mathbb{K}[\mathbf{y}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{r-1}]$ and $\mathsf{deg}(\mathbf{L}_r^{(1)}) = \mathsf{deg}(\mathbf{L}_r^{(2)})$:

$$\begin{aligned} \mathsf{deg}(x_{i,r}) &= \max\{\mathsf{deg}(\mathbf{L}_r^{(1)}), \mathsf{deg}(\mathbf{L}_r^{(2)} s_{i,r}), \mathsf{deg}(s_{i,r}^2)\} = \mathsf{deg}(\mathbf{L}_r^{(2)}) + 1 \\ \mathsf{deg}(y_{i,r}) &= \max\{\mathsf{deg}(\mathbf{L}_r^{(2)}), \mathsf{deg}(s_{i,r})\} = \mathsf{deg}(\mathbf{L}_r^{(2)}) \end{aligned} \tag{5.9}$$

Recursively, $\mathsf{deg}(\mathbf{L}_r^{(1)})$ and $\mathsf{deg}(\mathbf{L}_r^{(2)})$ depend on $\mathsf{deg}(\mathbf{x}_{r-1})$ and $\mathsf{deg}(\mathbf{y}_{r-1})$:

$$\begin{aligned} \mathsf{deg}(\mathbf{L}_r^{(2)}) &= \mathsf{deg}(\mathbf{L}_r^{(1)}) \\ &= \max\{\mathsf{deg}(\mathbf{x}_{r-1}), \mathsf{deg}(\mathbf{y}_{r-1})\} \\ &= \max\{\mathsf{deg}(\mathbf{g}_{r-1})\} + 1 = r \end{aligned}$$

The last result follows from $\max\{\mathsf{deg}(\mathbf{g}_1)\} = 1$. Therefore, taking all these results into account, we can define the following degrees for the P_{CICO} model equations:

$$\begin{cases} \mathsf{deg}(p_{i,r}) & = \max\{\alpha, \mathsf{deg}(Q_\gamma) \cdot \mathsf{deg}(\mathbf{L}_r^{(2)}), \mathsf{deg}(\mathbf{L}_r^{(1)})\} = \max\{\alpha, 2r\} \\ \mathsf{deg}(x_{i,N+1}) & = \max\{\mathsf{deg}(x_{i,N}), \mathsf{deg}(y_{i,N})\} = \mathsf{deg}(g_{i,N}) + 1 = N + 1 \end{cases} \tag{5.10}$$

for $1 \le i \le l$ and $1 \le r \le N$.

5.4.2 FCICO Model

Differently from the P_{CICO} model, the F_{CICO} unknowns will be both $s_{i,r}$, both $y_{i,r}$ for each $1 \le i \le l$ and $1 \le r \le N + 1$.

Polynomials representation

For each $1 \le r \le N$, let $\mathbf{L}_r^{(2)}$ and $y_{i,r}$ be the inputs to the non-linear verification layer V in the r-th round. Notice that this operation is performed for each couple $(\mathbf{L}_r^{(2)}, y_{i,r})$ for $1 \le i \le l$. It outputs the functions $\mathbf{L}_r^{(1)}, x_{i,r}$ where:

$$\begin{bmatrix} \mathbf{L}_r^{(1)} \\ x_{i,r} \end{bmatrix} := V(\mathbf{L}_r^{(2)}, y_{i,r}) = \begin{bmatrix} Q_\gamma(\mathbf{L}_r^{(2)}) + E(s_{i,r}) \\ Q_\delta(y_{i,r}) + E(s_{i,r}) \end{bmatrix}$$

By that definition, $\mathbf{L}_r^{(2)} \in \mathbb{K}[\mathbf{x}_{r-1}, \mathbf{y}_{r-1}, \mathbf{s}_{r-1}]$, then $x_{i,r} \in \mathbb{K}[y_{i,r}, s_{i,r}]$ and $\mathbf{L}_r^{(1)} \in \mathbb{K}[\mathbf{x}_{r-1}, \mathbf{y}_{r-1}, \mathbf{s}_{r-1}, s_{i,r}]$ for $1 \le r \le N$ and $1 \le i \le l$. By applying the CICO input constraint we defined above, rather is fixing $x_{i,0} = 0$ for all $1 \le i \le l$, we get the first equation of our system.

$$q_{i,1}^{(1)} := \mathcal{L}_{i,1}^{(1)}\left(\mathbf{0}, \mathbf{y}_0\right) - Q_\gamma\left(\mathcal{L}_{i,1}^{(2)}\left(\mathbf{0}, \mathbf{y}_0\right)\right) - S_{i,1}^{\alpha} \tag{5.11}$$

for all $1 \leq i \leq l$ where $L_{i,j}^1$ and $L_{i,j}^2$ are respectively the i-th element of the first and the second row of the resulting matrix after the application of the linear layer at the round j.

Let $m_{i,r} := Q_\delta(y_{i,r-1}) + E(s_{i,r-1})$, then, the other equations are similarly defined, but they involve the previous variables as follows:

$$q_{i,r}^{(1)} := \mathcal{L}_{i,r}^{(1)}\left(\mathbf{m}_r, \mathbf{y}_{r-1}\right) - Q_\gamma\left(\mathcal{L}_{i,r}^{(2)}\left(\mathbf{m}_r, \mathbf{y}_{r-1}\right)\right) - S_{i,r}^{\alpha} \tag{5.12}$$

for all $1 \leq i \leq l$ and $1 < r \leq N$.

Additionally, for each round $1 \leq r \leq N$, we need to link the variables $\mathbf{y}_{r-1}$, $y_{i,r}$ and $s_{i,r}$ with the following equations which take care of the effect of the function E:

$$q_{i,r}^{(2)} := \left(\mathcal{L}_{i,r}^{(2)}\left(\mathbf{m}_r, \mathbf{y}_{r-1}\right) - y_{i,r}\right)^{\alpha} - s_{i,r}^{\alpha} \tag{5.13}$$

By doing so, we obtain $2l$ equations per round, meaning $2Nl$ equations. At the end, after the last rounds, the linear layer L is applied again. Then, the last l equations are of the form:

$$f_{i,N+1} = \mathcal{L}_{i,N}^{(1)}\left(\mathbf{m}_N, \mathbf{y}_N\right) \tag{5.14}$$

which are equal to zero with respect to the CICO constraints.

Definition 3 (**F_{CICO} model for arbitrary l**). *An algebraic model of the* Anemoi *permutation $A_\pi : \mathbb{K}^{2l} \to \mathbb{K}^{2l}$ applied for N rounds, under the CICO constraints in Equation 5.2, is given by the system*

$$F_{CICO} = \{\boldsymbol{q}_1^{(1)}, \ldots, \boldsymbol{q}_N^{(1)}, \boldsymbol{q}_1^{(2)}, \ldots, \boldsymbol{q}_N^{(2)}, \boldsymbol{f}_{N+1}\}$$

where $F_{CICO} \subset \mathbb{K}[\boldsymbol{y}_0, \ldots, \boldsymbol{y}_N, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_N]$, $q_{i,r}^{(1)}$ is defined as in Equation 5.12, $q_{i,r}^{(2)}$ is defined as in Equation 5.13 and $f_{i,N+1}$ as in Equation 5.14. The generated system contains $l(2N+1)$ equations in $l(2N+1)$ variables.

Equations degree analysis for q = p prime

To make a complete analysis of the F_{CICO} model, we give a detailed inspection of the degrees of the polynomials we defined in Subsection 5.4.3 with respect to a field of characteristic p prime for which $\mathsf{deg}(Q_\gamma) = \mathsf{deg}(Q_\delta) = 2$.

Let consider the equation $q_{i,1}^{(1)}$ given in Equation 5.11. $L_{i,1}^{(1)}$ and $L_{i,1}^{(2)}$ depend on the degree 1 variables $\mathbf{y}_0$. It means that

$$\mathsf{deg}(q_{i,1}) = \max\{\mathsf{deg}(\mathbf{y}_0), \mathsf{deg}(\mathbf{y}_0)^2, \mathsf{deg}(s_{i,1})^{\alpha}\} = \alpha \tag{5.15}$$

Now, let consider the equation $q_{i,r}^{(1)}$ with $1 < r \leq N$ given in Equation 5.12. The sub-equation $m_{i,r} := Q_\delta(y_{i,r-1}) + E(s_{i,r-1})$ depends on the degree 1 variables $y_{1,r-1}$ and $s_{1,r-1}$. Then $\mathsf{deg}(m_{i,r}) = \max\{\mathsf{deg}(y_{i,r-1})^2, \mathsf{deg}(s_{i,r-1})^{\alpha}\} = \alpha$.

As a consequence, the degree of $q_{i,r}$ is:

$$\mathsf{deg}(q^{(1)}_{i,r}) = \max\{\mathsf{deg}(\mathbf{m}_r), \mathsf{deg}(\mathbf{m}_r)^2, \mathsf{deg}(s_{i,r-1})^{\alpha}\} = 2\alpha \tag{5.16}$$

The same can be done for equation $q^{(2)}_{i,r}$ for $1 \leq r \leq N$ (Equation 5.13). Its degree is:

$$\mathsf{deg}(q^{(2)}_{i,r}) = \max\{\mathsf{deg}(\mathbf{m}_r)^{\alpha}, \mathsf{deg}(\mathbf{y}_{r-1})^{\alpha}, \mathsf{deg}(s_{i,r})^{\alpha}\} = \alpha^2 \tag{5.17}$$

Finally, the degree of the last equations, given by the additional application of the linear layer (Equation 5.14), is:

$$\mathsf{deg}(f_{i,N+1}) = \max\{\mathsf{deg}(\mathbf{m}_N), \mathsf{deg}(\mathbf{y}_N)\} = \alpha \tag{5.18}$$

Therefore, taking all these results into account, we can define the following degrees for the F_{CICO} model equations:

$$\begin{cases} \mathsf{deg}(q^{(1)}_{i,1}) = \alpha & \\ \mathsf{deg}(q^{(1)}_{i,r}) = 2\alpha & \text{for } 1 < r \leq N \\ \mathsf{deg}(q^{(2)}_{i,r}) = \alpha^2 & \text{for } 1 \leq r \leq N \\ \mathsf{deg}(f_{i,N+1}) = \alpha & \end{cases} \tag{5.19}$$

for $1 \leq i \leq l$.

5.4.3 ACICO model

Polynomials representation

We can define the following functions for every $1 \leq r \leq N$:

1) Let $L^{(2)}_{i,r}$ and $y_{i,r}$ be the inputs to the non-linear verification layer V in the r-th round. Notice that this operation is performed for each couple $(L^{(2)}_{i,r}, y_{i,r})$ for $1 \leq i \leq l$. It outputs $L^{(1)}_{i,r}, x_{i,r}$ where:

$$\begin{bmatrix} L^{(1)}_{i,r} \\ x_{i,r} \end{bmatrix} := V(L^{(2)}_{i,r}, y_{i,r}) = \begin{bmatrix} Q_{\gamma}(L^{(2)}_{i,r}) + E(s_{i,r}) \\ Q_{\delta}(y_{i,r}) + E(s_{i,r}) \end{bmatrix}$$

By that definition:

- for $r = 1$ and $1 \leq i \leq l$, $L^{(2)}_{i,1} \in \mathbb{K}[\mathbf{x}_0, \mathbf{y}_0]$, $x_{i,1} \in \mathbb{K}[y_{i,1}, s_{i,1}]$ and $L^{(1)}_{i,1} \in \mathbb{K}[\mathbf{x}_0, \mathbf{y}_0, s_{i,r}]$
- for $1 < r \leq N$ and $1 \leq i \leq l$, $L^{(2)}_{i,r} \in \mathbb{K}[\mathbf{x}_{r-1}, \mathbf{y}_{r-1}, \mathbf{s}_{r-1}]$, $x_{i,r} \in \mathbb{K}[y_{i,r}, s_{i,r}]$ and $L^{(1)}_{i,r} \in \mathbb{K}[\mathbf{x}_{r-1}, \mathbf{y}_{r-1}, \mathbf{s}_{r-1}, s_{i,r}]$

By applying the input constraint, that is setting $x_{i,0} = 0$ for all $1 \leq i \leq l$, we get the first family of polynomials

$$a_{i,1} := \mathcal{L}^{(1)}_{i,1}(\mathbf{0}, \mathbf{y}_0) - Q_{\gamma}\left(\mathcal{L}^{(2)}_{i,1}(\mathbf{0}, \mathbf{y}_0)\right) - s^{\alpha}_{i,1}, \text{ where } 1 \leq i \leq l. \tag{5.20}$$

Assuming $m_{i,r} := Q_\delta(y_{i,r-1}) + E(s_{i,r-1})$, we get

$$a_{i,r} := \mathcal{L}^{(1)}_{i,r}(\mathbf{m}_r, \mathbf{y}_{r-1}) - Q_\gamma\left(\mathcal{L}^{(2)}_{i,r}(\mathbf{m}_r, \mathbf{y}_{r-1})\right) - s^{\alpha}_{i,r} \tag{5.21}$$

where $1 \le i \le l$ and $1 < r \le N$. Thus we have $\mathsf{deg}(a_{i,1}) = \alpha$. Since $\mathsf{deg}(m_{i,r}) = \alpha$ and $\mathcal{L}_{i,r}$ is a linear function we have

$$\mathsf{deg}(a_{i,r}) = \max\{\mathsf{deg}(\mathbf{m}_r), \mathsf{deg}(\mathbf{m}_r)^2, \mathsf{deg}(s_{i,r-1})^{\alpha}\} = 2\alpha$$

Additionally, for each round, from the relations of the variables $\mathbf{y}_{r-1}$, $y_{i,r}$ and $s_{i,r}$ we get

$$b_{i,1} := \mathcal{L}^{(2)}_{i,1}(\mathbf{0}, \mathbf{y}_0) - y_{i,1} - s_{i,1}, \text{ for } r = 1 \tag{5.22}$$

$$b_{i,r} := \mathcal{L}^{(2)}_{i,r}(\mathbf{m}_r, \mathbf{y}_{r-1}) - y_{i,r} - s_{i,r}, \text{ for } 1 < r \le N \tag{5.23}$$

From the above two equations it is clear that $\mathsf{deg}(b_{i,1}) = 1$, and $\mathsf{deg}(b_{i,r}) = \max\{\alpha, 1\} = \alpha$.

Thus, we obtain $2l$ polynomials per round given by $a_{i,r}, b_{i,r}$ (for $1 \le i \le l$) i.e. $2Nl$ equations overall. At the end, after the last rounds, the linear transformation L is applied. This leads to l equations of the form:

$$f_{i,N+1} := \mathcal{L}^{(1)}_{i,N}(\mathbf{m}_{N+1}, \mathbf{y}_N) \tag{5.24}$$

which are equal to zero due to constrained output and $\mathsf{deg}(f_{i,N+1}) = \alpha$.

Definition 4 (ACICO model for $l \ge 1$). *An algebraic model of the* `Anemoi` *permutation $A_\pi : \mathbb{K}^{2l} \to \mathbb{K}^{2l}$ applied for N rounds, under the CICO constraints in Equation 5.2, is given by the system*

$$A_{CICO} = \{\boldsymbol{a}_1, \dots, \boldsymbol{a}_N, \boldsymbol{b}_1, \dots, \boldsymbol{b}_N, \boldsymbol{f}_{N+1}\}$$

where $A_{CICO} \subset \mathbb{K}[\boldsymbol{y}_0, \dots \boldsymbol{y}_N, \boldsymbol{s}_1, \dots, \boldsymbol{s}_N]$ and $a_{i,r}$, $b_{i,r}$, $f_{i,N+1}$ are as defined in eq. (5.21)*, eq.* (5.23)*, eq.* (5.24) *respectively. The generated system contains $l(2N+1)$ equations in $l(2N+1)$ variables.*

Degrees of polynomials

For GB analysis with ACICO model, we require the degrees of the polynomials in A_{CICO}, that are as follows

$$\mathsf{deg}(a_{i,1}) = \alpha, \mathsf{deg}(b_{i,1}) = 1, \mathsf{deg}(f_{i,N+1}) = \alpha \tag{5.25}$$

$$\mathsf{deg}(a_{i,r}) = 2\alpha, \mathsf{deg}(b_{i,r}) = \alpha \tag{5.26}$$

for $1 \le i \le l$ and $1 < r \le N$.

5.5 Gröbner basis cryptanalysis against Anemoi

In this section we describe the Gröbner basis cryptanalysis of `Anemoi`. In particular, we will show what is the particular structure of the DRL Gröbner basis generated by the A_{CICO} polynomials and how we can avoid converting it to a LEX Gröbner basis to obtain a univariate polynomial. At the end, we compare our results to previous ones.

5.5.1 Gröbner basis for `Anemoi`

Here, we show the structural properties of the polynomials in the GB of the ideal $\langle A_{CICO}\rangle$ w.r.t DRL monomial ordering. We use $LM_k(f)$ to denote the k^{th} (≥ 2) monomial of the polynomial f, as already done in Chapter 2.

Proposition 1. *DRL Basis structure for $\alpha = 3$ Let F be the system of polynomials generated by the A_{CICO} model and I the corresponding ideal. For $\alpha = 3$, the DRL Gröbner basis G_{DRL} of I contains a well-structured set of polynomials. In particular, they always have polynomials of the following form:*

- *l equations of degree 1 whose leading terms are $y_{i,0}$ for all $1 \leq i \leq l$:*

$$g_{i,0} := y_{i,0} + Q(\boldsymbol{y}_1, \boldsymbol{s}_1)$$

  for $1 \leq k \leq l$, where $\mathtt{LM}(Q) = y_{1,1}$ and the other monomials are linear terms.

- *Nl equations of degree α whose leading term is $y_{i,j}^{\alpha}$ for all $1 \leq i \leq l$ and $1 \leq j \leq r$:*

$$g'_{i,j} := y_{i,j}^{\alpha} + Q(s_{i,j}, \boldsymbol{s}_{j+1}, \boldsymbol{y}_{j-1}, \boldsymbol{y}_j, \boldsymbol{y}_{j+1}) \textit{for } 1 \leq k \leq l$$

  for $l < k \leq Nl + l$, where $\mathtt{LM}(Q) = y_{i,j-1}^{\lfloor \frac{\alpha}{2} \rfloor + 1} s_{i,j}^{\lfloor \frac{\alpha}{2} \rfloor}$ and the remaining monomials define a generic polynomial up to degree α.

- *Nl equations of degree α whose leading term is $s_{i,j}^{\alpha}$ for all $1 \leq i \leq l$ and $1 \leq j \leq r$:*

$$g_{i,j} := s_{i,j}^{\alpha} + Q(\boldsymbol{y}_j, \boldsymbol{y}_{j+1}, \boldsymbol{s}_{j+1})$$

  for $Nl + l < k \leq 2Nl + l$, where $\mathtt{LM}(Q) = y_{i,j}^2$ and the other monomials are linear terms.

- *Nl of degree 2 whose leading terms are $y_{i,j}s_{i,j}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$:*

$$g''_{i,j} := y_{i,j}s_{i,j} + Q(s_{i,j}, \boldsymbol{s}_{j+1}, \boldsymbol{y}_{j-1}, \boldsymbol{y}_j, \boldsymbol{y}_{j+1})$$

  for $2Nl + l < k \leq 3Nl + l$, where $\mathtt{LM}(Q) = s_{i,j}^2$ and the other monomials are linear terms.

Notice that when $j = N$, the variables $\boldsymbol{s}_{N+1}$ and $\boldsymbol{y}_{N+1}$ do not exist, meaning that the equation depend only on the remaining variables. Moreover, the complexity of computing the Gröbner basis is given by

$$\mathcal{O}(\frac{poly_1(l)N - poly_2(l)}{2})$$

field operations where $poly_1(l) = l(24l^4 + 40l^3 + 45l^2 + 16l - 1)$ and $poly_2(l) = l(17l^4 + 4l^3 + 26l^2 + 16l + 1)$. The given complexity does not take care of the final reductions to obtain a reduced Gröbner basis.

Remark 1. DRL Basis structure for $\alpha = 5, 7, 11$ Let F be the system of polynomials generated by the A_{CICO} model and I the corresponding ideal. For $\alpha = 5, 7, 11$, the DRL Gröbner basis G_{DRL} of I

contains the polynomials $g_{i,j}$ defined in Proposition 1 plus other polynomials which are useful to shorten the quotient ring basis and obtain the minimum univariate polynomial degree needed in 5.5.2.

As pointed out by Remark 1, due to the DRL monomial ordering, we are not able to perfectly describe the structure of the Gröbner basis for $\alpha = 5, 7, 11$. The issue comes from the fact that $s^\alpha \prec_{\mathsf{DRL}} y^2$. To circumvent this issue we can introduce a weighted monomial ordering, denoted as W_{DRL}, such that $y^2 \prec_{\mathsf{W}_{DRL}} s^\alpha$. In particular, we want to define weights such that $2w(y) \geq \alpha w(s)$. Therefore, $w(s) = 2k \rightarrow w(y) = \alpha k$.

Definition 5 (Weighted monomial ordering for `Anemoi`). *The following definition is an instantiation of Definition 11. Let $\boldsymbol{w} = (w_0, w_1) \in \mathbb{R}^2$, where $w_j \neq 0$ for $j = 0, 1$. We define $\prec_{\mathsf{W}_{DRL}}$ to be the weighted DRL monomial order defined by weights w_0, w_1, where w_0 is the weight associated with the variables $y_{i,r}$ for $1 \leq i \leq l$ and $0 \leq r \leq N$ and w_1 is the weight associated with the variables $s_{i,r}$ for $1 \leq i \leq l$ and $1 \leq r \leq N$. Moreover, we constrain the weights to satisfy the following inequality:*

$$2w_0 \geq \alpha w_1.$$

Given two monomials M_0 and M_1, let $k_y^{(0)}$ and $k_y^{(1)}$ be the number of variables $y_{i,r}$ in M_0 and M_1. Furthermore, $k_s^{(0)}$ and $k_s^{(1)}$ are the number of variables $s_{i,r}$ in M_0 and M_1. We say that:

$$M_0 := \prod_{i=1}^{k_y^{(0)}} y^{e_i^{(0)}} \prod_{i=1}^{k_s^{(0)}} s^{u_i^{(0)}} \prec_{\mathsf{W}_{DRL}} \prod_{i=1}^{k_y^{(1)}} y^{e_i^{(1)}} \prod_{i=1}^{k_s^{(1)}} s^{u_i^{(1)}}$$

$$\Updownarrow$$

$$\begin{cases} \sum_{i=1}^{k_y^{(0)}} w_0 e_i^{(0)} + \sum_{i=1}^{k_s^{(0)}} w_1 u_i^{(0)} > \sum_{i=1}^{k_y^{(1)}} w_0 e_i^{(1)} + \sum_{i=1}^{k_s^{(1)}} w_1 u_i^{(1)} \\ \sum_{i=1}^{k_y^{(0)}} w_0 e_i^{(0)} + \sum_{i=1}^{k_s^{(0)}} w_1 u_i^{(0)} = \sum_{i=1}^{k_y^{(1)}} w_0 e_i^{(1)} + \sum_{i=1}^{k_s^{(1)}} w_1 u_i^{(1)}, M_0 \prec_{\mathtt{revlex}} M_1 \end{cases}$$

Proposition 2. *W_{DRL} Basis structure for $\alpha = 5, 7, 11$ Let F be the system of polynomials generated by the A_{CICO} model and I the corresponding ideal. For $\alpha = 5, 7, 11$, the W_{DRL} Gröbner basis $G_{\mathsf{W}_{DRL}}$ of I contains a well-structured set of polynomials. In particular, they always have polynomials of the following form:*

- *l equations of degree 1 whose leading terms are $y_{i,0}$ for all $1 \leq i \leq l$:*

$$g_{i,0} := y_{i,0} + Q(\boldsymbol{y}_1, \boldsymbol{s}_1)$$

for $1 \leq k \leq l$, where $\mathtt{LM}(Q) = y_{1,1}$ and the other monomials are linear terms.

- *Nl equations of degree $\alpha + 1$ whose leading term is $s_{i,j}^{\alpha+1}$ for all $1 \leq i \leq l$ and $1 \leq j \leq r$:*

$$g'_{i,j} := s_{i,j}^{\alpha+1} + Q(s_{i,j}, \boldsymbol{s}_{j+1}, \boldsymbol{y}_{j-1}, \boldsymbol{y}_j, \boldsymbol{y}_{j+1}) \textit{for } 1 \leq k \leq l$$

for $l < k \leq Nl + l$, where $\mathtt{LM}(Q) = y_{1,j-1} y_{i,j}$ and the remaining monomials define a generic polynomial up to degree α.

- *Nl equations of degree 2 whose leading term is $y_{i,j}^2$ for all $1 \leq i \leq l$ and $1 \leq j \leq r$:*

$$g_{i,j} := y_{i,j}^2 + Q(s_{i,j}^\alpha, y_{l-i-1,j}, \boldsymbol{y}_{j+1}, \boldsymbol{s}_{j+1})$$

for $Nl + l < k \leq 2Nl + l$, where $\texttt{LM}(Q) = s_{i,j}^\alpha$ and the other monomials are linear terms.

- *Nl of degree 2 whose leading terms are $y_{i,j}s_{i,j}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$:*

$$g''_{i,j} := y_{i,j}s_{i,j} + Q(s_{i,j}, \boldsymbol{s}_{j+1}, \boldsymbol{y}_{j-1}, \boldsymbol{y}_j, \boldsymbol{y}_{j+1})$$

for $2Nl + l < k \leq 3Nl + l$, where $\texttt{LM}(Q) = y_{1,j-1}$ and the other monomials are linear terms.

Notice that when $j = N$, the variables $\boldsymbol{s}_{N+1}$ and $\boldsymbol{y}_{N+1}$ do not exist, meaning that the equation depend only on the remaining variables.

5.5.2 Univariate polynomial finding

For obtaining the univariate polynomial(s), we can mainly use two different approaches

- Basis conversion: converting G_{DRL} to G_{LEX} using `SparseFGLM`.
- Eigenvalue method: applying one of the following methods:
  - computing $det(x_i\mathbf{I}_{d_I} - T_i)$ that we refer as `polyDet` (following [3]).
  - applying Wiedemann algorithm that we call `linSeq` (since it generates a linearly recurring sequence from the multiplication matrix T_i)

Change of basis by `SparseFGLM` One common approach is to apply the FGLM [20] algorithm to convert G_{DRL} to G_{LEX}. In particular, due to the sparsity of the polynomials involved, we can apply `SparseFGLM` [21], whose complexity is:

$$\mathcal{O}(n_v \cdot d_I^\omega \log(d_I)) \tag{5.27}$$

operation in $\mathbb{K}$, where ω is the linear algebra constant which, in our case, is theoretically bounded by $2 \leq \omega \leq 2.3727$ [44]. $d_I = dim_{\mathbb{K}}(R/I)$ is the dimension of the quotient ring R/I and a good approximation of this value is necessary to precisely define the security of `Anemoi`. By applying that algorithm we obtain a Gröbner basis G_{LEX} containing a univariate polynomial in the smallest variable. As we already know, the complexity of `SparseFGLM` can also be defined in terms of the sparsity of the involved matrices:

$$\mathcal{O}(d_I(\mathcal{Z} + n_v \log(d_I))) \tag{5.28}$$

where $\mathcal{Z}$ is the number of non-zero entries. This way of representing the complexity of `SparseFGLM` permits to compare that method with the iterated application of the Wiedemann algorithm, as we discussed in Chapter 4.

Eigenvalue method with `polyDet` The `SparseFGLM` introduces a lot of unnecessary computations with respect to our objective namely, finding possible values for the inputs $y_{1,0}, \dots, y_{l,0}$. Instead of using `SparseFGLM` we can compute the multiplication matrix of the l variables we are interested in and, for each of them, compute $\det(y_{i,0}\mathbf{I}_{d_I} - T_i)$ for $1 \le i \le l$. To compute such determinant we can apply the Labahn et al. algorithm [30] whose complexity is given by

$$\mathcal{O}(d_I^{\omega}\lceil s \rceil)$$

where s is the mean of the column degrees. As a consequence, if we use that method to find l univariate polynomials, the overall complexity of `polyDet` becomes

$$\mathcal{O}(l \cdot d_I^{\omega}).$$

Further details will be given in Section 5.6.

Eigenvalue method with Wiedemann algorithm A third option which can save a lot of unnecessary computations is given by the Wiedemann algorithm. Indeed, we can compute the multiplication matrix of the l variables we are interested in and, for each of them, generate a linearly recurring sequence to determine, thanks to the Berlekamp-Massey algorithm [5], its minimal polynomial [21, 25]. In Chapter 3 we presented the two versions of the Wiedemann algorithm. In our analysis, we use the probabilistic version (Algorithm 8) whose runtime complexity is bounded by

$$\mathcal{O}(d_I(\mathcal{Z} + \log(d_I))) \tag{5.29}$$

operation in $\mathbb{K}$, where $\mathcal{Z}$ is the number of non-zero entries in T_i. The probability of success of the probabilistic algorithm is very high, close to 1 in our case, and it is given by:

$$P_q(n) = \begin{cases} (1-\frac{1}{q})^{2n} & \text{if } q \ge N \\ (1-\frac{1}{q})^{2q}(1-\frac{1}{q^2})^{n-q} & \text{if } \sqrt{n} \le q < n \end{cases} \tag{5.30}$$

where q is the field cardinality and n is the matrix dimension [25].

By applying that method for all the l input/target variables, the overall complexity becomes:

$$ld_I(\mathcal{Z} + \log(d_I)) \tag{5.31}$$

On the value of d_I The dimension d_I is crucial in determining the complexity of obtaining univariate polynomial. In [29] d_I for `Anemoi` is conjectured as $d_I = (\alpha+2)^N$ for $l = 1$ with experimental evidences. We prove and extend this conjecture.

Proposition 3. *Let N be the number of rounds of `Anemoi` for $l \ge 1$ and $\alpha = 3$ (resp. $\alpha = 3, 5, 7, 11$). The dimension of the quotient ring basis of the DRL Gröbner basis (resp. W_{DRL} Gröbner basis) with respect to the instance `Anemoi`(N, α, l) is:*

$$d_I = |B_{G_{DRL}}| = |B_{G_{W_{DRL}}}| := (\alpha+2)^{Nl}$$

Proof. We need to distinguish two cases:

DRL From Proposition 1, we know that G_{DRL} will be composed by:

1) l equations of degree 1 whose leading terms are $y_{i,0}$ for all $1 \leq i \leq l$.
2) $2Nl$ equations of degree α whose leading term is $y_{i,j}^{\alpha}$ or $s_{i,j}^{\alpha}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$.
3) Nl equations of degree 2 whose leading terms are $y_{i,j}s_{i,j}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$.

We are interested in defining the quotient ring basis $B_{G_{\mathsf{DRL}}}$. Due to the fact that the equations from point 1 have got degree 1, we can directly skip them. By considering the equations given at point 2 we would define a quotient ring basis made by α^{2Nl} monomials. But, by point 3, we have to remove all the monomials which are multiples of the leading monomials of the equations at point 3, which act as filter equations: *e.g.* if $y_{i,j}^2 s_{i,j}^2 \in B_{G_{\mathsf{DRL}}}$, due to the fact that $y_{i,j}s_{i,j}$ is the leading monomial of one of those equations, we must remove it from $B_{G_{\mathsf{DRL}}}$. In order to directly count the cardinality of the final quotient ring basis, we need to consider what are the possible monomials $y_{i,j}^{a_0} s_{i,j}^{a_1} Q$, where Q is a polynomial depending on other variables and $a_0, a_1 \in [0, \ldots, \alpha - 1]$, which are not deleted by equations at point 3. For each monomial of that kind:

- if $a_0 > 0$, then $a_1 = 0$ and the possible couples of degrees are: $[1, 0], [2, 0]$.
- if $a_1 > 0$, then $a_0 = 0$ and the possible couples of degrees are: $[0, 1], [0, 2]$.
- the last possibility is $a_0 = 0$ and $a_1 = 0$: $[0, 0]$.

For each couple $(y_{i,j}, s_{i,j})$, we have $2(\alpha - 1) + 1 = 2\alpha - 1$ possible combinations. Due to the fact that $\alpha = 3$, $2\alpha - 1 = \alpha + 2$. We have Nl couples, then the possible valid monomials within the quotient ring basis are in total $(\alpha + 2)^{Nl}$.

W$_{DRL}$ From Proposition 2, we know that $G_{\mathsf{W}_{DRL}}$ will be composed by:

1) l equations of degree 1 whose leading terms are $y_{i,0}$ for all $1 \leq i \leq l$.
2) Nl equations of degree 2 whose leading term is $y_{i,j}^2$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$.
3) Nl equations of degree $\alpha + 1$ whose leading term is $s_{i,j}^{\alpha+1}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$.
4) Nl of degree 2 whose leading terms are $y_{i,j}s_{i,j}$ for all $1 \leq i \leq l$ and $1 \leq j \leq N$.

We are interested in defining the quotient ring basis $B_{G_{\mathsf{W}_{DRL}}}$. As done for the previous case, we can directly skip equations at point 1. By considering only the equations at points 2 and 3, we would define a quotient ring basis made by $2^{Nl}\alpha^{Nl}$ monomials. But, by point 4, we have to remove all the monomials which are multiples of the leading monomials of the equations at point 4: *e.g.* if $y_{i,j}^2 s_{i,j}^2 \in B_{G_{\mathsf{W}_{DRL}}}$, due to the fact that $y_{i,j}s_{i,j}$ is the leading monomial of one of those equations, we must remove it from $B_{G_{\mathsf{W}_{DRL}}}$. In order to directly count the cardinality of the final quotient ring basis, we need to consider what are the possible monomials $y_{i,j}^{a_0} s_{i,j}^{a_1} Q$, where Q is a polynomial depending on other variables and $a_0 \in \{0, 1\}$ and $a_1 \in \{0, \ldots, \alpha\}$, which are not deleted by equations at point 4. For each monomial of that kind:

- if $a_0 > 0$, then $a_1 = 0$ and the possible couples of degrees are:

$$[1, 0] \text{ (1 couple).}$$

- if $a_1 > 0$, then $a_0 = 0$ and the possible couples of degrees are:

$$[0, 1], [0, 2], \ldots, [0, \alpha] \text{ (α couples).}$$

- the last possibility is $a_0 = 0$ and $a_1 = 0$:

$$[0, 0] \text{ (1 couple).}$$

For each couple $(y_{i,j}, s_{i,j})$, we have $\alpha + 2$ possible combinations. We have Nl couples, then the possible valid monomials within the quotient ring basis are in total $(\alpha + 2)^{Nl}$.

Due to the fact that our target variables are involved only in the degree 1 polynomials, their multiplication matrix dimension will represent the degree of the associated univariate polynomial, that is also the number of roots. The dimension of the multiplication matrix is the dimension of the quotient ring basis, that is composed by all the monomials which are not multiples or a composition of the elements in $LM(I)$, where I is, in our case, the DRL or the W_{DRL} Gröbner Basis. The obtained result validates and generalizes the conjecture given by [29] for the case $l = 1$.

5.5.3 Polynomial Factorization or Root Finding

For `Anemoi` cryptanalysis (or for GB cryptanalysis in general) this part corresponds to finding the collision or preimage (resp. finding solution(s) to the polynomial system). The input to this section is the univariate polynomial obtained in the previous one. To find its roots, we can simply apply one of the off-the-shelf factorization algorithms. As we have seen in Section 4.2, there exist several choices. However, since we are dealing with finite fields, the latter one gives the best complexity. Therefore, for our analysis, we choose the Kaltofen-Shoup probabilistic algorithm, whose probabilistic complexity is given by

$$d^{1.815} \log(q) \tag{5.32}$$

where d is the degree of the univariate polynomial and q is the field characteristic.

Root finding To determine the complexity of finding the solutions to the system of polynomials, we need to figure out if we are dealing with an ideal in Shape Position, if we have a particular structure to exploit or if we must apply the methods in 5.5.2 for each variable. The Shape form for `Anemoi` has been conjectured by experimental results (even if a formal proof would be preferred). As a result, it is possible to apply `SparseFGLM`, getting, in this way, the only univariate polynomial required to find the complete variety of the analysed system. Due to the fact that we don't need to determine the solutions for all the variables involved in the system, but only for the input variables, we can exploit other methods.

Instead of using `SparseFGLM` the eigenvalue methods can be applied for each variable, leading to a complexity of

$$\mathcal{O}(n_v \cdot d_I^{\omega})$$

for `polyDet` and

$$\mathcal{O}(n_v d_I(\mathcal{Z} + \log(d_I)))$$

for the Wiedemann algorithm, where n_v is the number of variables of the system. As a result of this process, we find n_v univariate polynomials whose roots can be computed with the factorization algorithms above. Fortunately, the system of polynomials defined by A_{CICO} has a particular structure which can be exploited to avoid the computation of many univariate polynomials. We can just compute l univariate polynomials with the eigenvalue methods, factorize them and find the values for the remaining variables by using gcd (resp. Gaussian elimination) if $l = 1$ (resp. $l > 1$). Theorem 5 and Corollary 1 from [12] work on LEX Gröbner basis, but the algorithm proposed in Corollary 1 can be easily extended to our case. The polynomials in the DRL Gröbner basis given in Proposition 1 can be sorted with respect to the number and the type of variables they depend on:

- l equations which depend only on $l + 1$ variables:

$$g_{i,N}(s_{i,N}, \mathbf{y}_N) \text{ where } LM(g_{i,N}) = s_{i,N}^{\alpha}$$

- $2l$ equations which depend on $2l + 1$ variables:

$$g'_{i,N}(s_{i,N}, \mathbf{y}_{N-1}, \mathbf{y}_N) \text{ where } LM(g'_{i,N}) = y_{i,N}^{\alpha}$$
$$g''_{i,N}(s_{i,N}, \mathbf{y}_{N-1}, \mathbf{y}_N) \text{ where } LM(g''_{i,N}) = y_{i,N}s_{i,N}$$

- l equations which depend on $3l + 1$ variables:

$$g_{i,N-1}(s_{i,N-1}, \mathbf{y}_{N-1}, \mathbf{y}_N, \mathbf{s}_N) \text{ where } LM(g_{i,N-1}) = s_{i,N-1}^{\alpha}$$

- $2l$ equations which depend on $4l + 1$ variables:

$$g'_{i,N-1}(s_{i,N-1}, \mathbf{s}_N, \mathbf{y}_{N-2}, \mathbf{y}_{N-1}, \mathbf{y}_N) \text{ where } LM(g_{i,N}) = y_{i,N-1}^{\alpha}$$
$$g''_{i,N-1}(s_{i,N-1}, \mathbf{s}_N, \mathbf{y}_{N-2}, \mathbf{y}_{N-1}, \mathbf{y}_N) \text{ where } LM(g''_{i,N}) = y_{i,N-1}s_{i,N-1}$$

- ...

- l equations which depend on $3l + 1$ variables:

$$g_{i,j}(s_{i,j}, \mathbf{y}_j, \mathbf{y}_{j+1}, \mathbf{s}_{j+1}) \text{ where } LM(g_{i,j}) = s_{i,j}^{\alpha}$$

- $2l$ equations which depend on $4l + 1$ variables:

$$g'_{i,j}(\mathbf{y}_{j-1}, \mathbf{y}_j, \mathbf{y}_{j+1}, \mathbf{s}_{j+1}, s_{i,j}) \text{ where } LM(g'_{i,j}) = y_{i,j}^{\alpha}$$
$$g''_{i,j}(\mathbf{y}_{j-1}, \mathbf{y}_j, \mathbf{y}_{j+1}, \mathbf{s}_{j+1}, s_{i,j}) \text{ where } LM(g''_{i,j}) = y_{i,j}s_{i,j}$$

- ...
- l equations which depend on $2l+1$ variables:

$$g_{i,0}(y_{i,0}, \mathbf{s}_1, \mathbf{y}_1) \text{ where } LM(g_{i,0}) = y_{i,0}$$

To apply a modified version of Corollary 1 from [12], we need to find at least l univariate polynomials that depend on the variables $y_{i,N}$ for $1 \leq i \leq l$. Those polynomials, which we will denote as $h_i(y_{i,N})$, can be computed by using the methods discussed in 5.5.2 (`Wiedemann algorithm`, `polyDet`). We add them to the equations above in order to obtain the desired form. The procedure to compute the variety of the ideal generated by the DRL basis of `Anemoi` with respect to the A_{CICO} modelling is given in Algorithm 14. The same can be done for the W_{DRL} basis given in Proposition 2 due to the similar structure of the generated equations.

Remark 2. For $l = 1$, the previous algorithm can be simplified by substituting the linear system solving procedure with the gcd computation as in Corollary 1 from [12]. On the other hand, due to the fact that the equations generated in Steps 6, 9, etc... are degree 1 polynomials, we can simply solve one of the equations to get the value of the unknown variable.

The complexity of this algorithm depends on the computation of the required l univariate polynomials for which we can choose one of the algorithms previously presented. The other operations are negligible with respect to that one. Indeed, the complexity of solving a linear system of l equations in l unknowns is in the worst case $\mathcal{O}(l^3)$. Moreover, if $l = 1$, we deal with 2 equations of degree 1 in 1 unknown, meaning that its solution comes for free.

5.6 Theoretical results

In this section we discuss the complexities for the first two steps of the Gröbner basis cryptanalysis methodology with respect to `Anemoi`.

5.6.1 Gröbner basis computation complexity

Propositions 1 and 2 show that the Gröbner basis computation complexity, with respect to the system generated by the A_{CICO} model, polynomially depends on the number of branches l and number of rounds N. Therefore, due to the fact that such result makes the complexity of computing the GB "negligible" with respect to the computation of the univariate polynomial, it is not a good choice to make the security of `Anemoi` relying on that complexity bound. Moreover, our experimental results heavily support our choice. As a consequence, we will consider a threat model where the attacker is always able to efficiently compute the GB, focusing our attention on the second step of the Gröbner basis attack, meaning the univariate polynomial finding.

5.6.2 Univariate polynomial finding complexities

We firstly present the results with $\omega = 2.8074$ and, in particular, we show how this complexity can be improved by using the Wiedemann algorithm. Secondly, from a designer point-of-view, we present the

Algorithm 14 Solutions of the polynomial system

Let $I_{\mathsf{DRL}} \subset R$ be the ideal generated by the DRL basis of `Anemoi` with respect to the A_{CICO} representation. $\mathbb{V}(I_{\mathsf{DRL}})$ can be computed as follows:

1. Compute the univariate polynomials $h_i(y_{i,N})$ for $1 \le i \le N$ by using `polyDet` or `Wiedemann algorithm`.
2. Find the roots of the univariate polynomials, that is finding a set of solutions for each of the variables $y_{i,N}$.
3. If $\exists\ h_i$ such that h_i is irreducible, $\mathbb{V}(I_{\mathsf{DRL}}) = \emptyset$ else, compute the Cartesian product between those sets.
4. For each tuple of roots $\tau = (\tau_1, \dots, \tau_l)$ in the Cartesian product, compute:
$$p_{i,N} = g_{i,N}(s_{i,N}, \tau)$$
for $1 \le i \le l$.
5. Compute the roots of $p_{i,N}$ for $1 \le i \le l$ to find the possible values of $s_{i,N}$. If $\exists\ p_{i,N}$ irreducible, $\mathbb{V}(I_{\mathsf{DRL}}) = \emptyset$ else compute the Cartesian product between the found sets of roots.
6. For each tuple of roots $\tau' = (\tau'_1, \dots, \tau'_l)$ in the Cartesian product, compute the system of $2l$ linear equations:
$$\{g'_{i,N}(\mathbf{y}_{N-1}, \tau'_i, \tau), \dots, g''_{i,N}(\mathbf{y}_{N-1}, \tau'_i, \tau)\}$$
where $1 \le i \le l$. Solve the system of $2l$ equations in l unknowns (we just need l equations) and get the new set of roots $\tau'' = (\tau''_1, \dots, \tau''_l)$.
7. Compute
$$p_{i,N-1} = g_{i,N-1}(s_{i,N-1}, \tau, \tau', \tau'')$$
for each $1 \le i \le l$.
8. Compute the roots $p_{i,N-1}$ for $1 \le i \le l$. If $\exists\ p_{i,N-1}$ irreducible, $\mathbb{V}(I_{\mathsf{DRL}}) = \emptyset$ else compute the Cartesian product between the found sets of roots.
9. For each tuple of roots $\tau''' = (\tau'''_1, \dots, \tau'''_l)$ in the Cartesian product, compute the system of $2l$ linear equations:
$$\{g'_{i,N-1}(\mathbf{y}_{N-2}, \tau'''_i, \tau', \tau), \dots, g''_{i,N-1}(\mathbf{y}_{N-2}, \tau'''_i, \tau', \tau)\}$$
where $1 \le i \le l$. Solve the system in l unknowns as done in step 6 and get the new set of roots $\tau'''' = (\tau''''_1, \dots, \tau''''_l)$.
10. Proceed similarly for the other polynomials $g_{i,j}, g'_{i,j}, g''_{i,j}$ for $1 \le i \le l$ and $1 \le j \le N-2$.
11. At the end, consider the polynomials $g_{i,0}$. If we reached this step, it means that we know values for each of the variables they depend on. Being equations of degree 1, the values of the variables $y_{i,0}$ are easy to find by substitution.

results for $\omega = 2$ and the required number of rounds to reach the claimed security level with respect to that conservative and designing choice. We will denote by C_{alg} the complexity of the considered algorithms. In particular, $C_{alg} \in \{C_{FGLM}, C_{polyDet}, C_{linSeq}\}$, where $C_{FGLM}, C_{polyDet}, C_{linSeq}$, denote respectively the FGLM (basis conversion), the determinant computation and the Wiedemann algorithm complexities.

In the attack of Koschatko et al. [29] the dominating complexity is represented by the Gröbner basis computation C_{GB}, even if their experimental results showed the opposite. Table 5.2 presents their results showing the security level reached by the standard number of rounds of `Anemoi`. On the other hand, Bariant et al. [3] present the attack based on the construction of the FreeLunch systems (see Appendix A). In this case, the dominating complexity is represented by the matrix determinant computation $C_{polyDet}$. Table 5.2 summarizes their results showing the achieved security level with respect to the standard number of rounds of `Anemoi`. From a designer perspective it is important to notice that, whilst Koschatko et al. used $\omega = 2$ as a conservative choice, Bariant et al. gave a more practical result by using $\omega = 2.8074$, that is the algebra constant for the currently best known algorithm for matrix multiplication.

	Koschatko et al. [29] ($\omega = 2$)				Bariant et al. [3] ($\omega = 2.81$)			
Security claim	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 11$	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 11$
128	117 (21)	144 (21)	152 (20)	152 (19)	118 (21)	156 (21)	174 (20)	198 (19)
256	210 (37)	257 (37)	277 (36)	280 (35)	203 (37)	270 (37)	307 (36)	358 (35)

Table 5.2: Complexity of the existing Gröbner Basis attacks for $l = 1$. The number of rounds is given in brackets. The two articles used different values of ω.

As discussed above, in our setting, the dominating complexity is represented by one of the methodologies (C_{FGLM}, $C_{polyDet}$, C_{linSeq}) we applied for the second step of the Gröbner basis attack. Due to the A_{CICO} model, the improvements to C_{GB} make that solution **applicable to more than 2 branches** ($l \geq 1$). Moreover, the usage of the Wiedemann algorithm gives us the possibility to fully exploit the sparsity of the involved equations and multiplication matrices, leading to a significant improvement to the currently known attacks.

Indeed, from Conjecture 1, we can define the complexity given in Equation 5.31 with respect to the number of rounds of `Anemoi`. The complexity of the probabilistic algorithm, applied for l variables, is

$$\mathcal{O}(l d_I(\mathcal{Z} + \log(d_I))) \tag{5.33}$$

where $d_I \cdot \mathcal{Z}$ represents the complexity given by the generation of the linearly recurring sequence and $d_I \log(d_I)$ is the complexity of the Berlekamp-Massey algorithm which finds the minimal polynomial [25] [43]. The probability of success of the Wiedemann algorithm, given in Equation 5.30, depends on the field size, but, if the conditions of Equation 5.30 are met, it is extremely high, close to 1. With this method we can give a correct value for the complexity of the Wiedemann algorithm without depending on the linear algebra constant, and we have seen that $\log_{d_I}(C_{\texttt{linSeq}})$ get closer to the conservative choice of $\omega = 2$ when the number of rounds grow. From an experimental point of view, that result is more

Security	128		256	
α	3	5	3	5
$l = 1$	111 (21)	143 (21)	196 (37)	253 (37)
	111 (21)	122 (<u>18</u>)	196 (37)	253 (37)
$l = 2$	171 (14)	212 (14)	269 (22)	335 (22)
	122 (<u>10</u>)	120 (<u>8</u>)	245 (<u>20</u>)	243 (<u>16</u>)
$l = 3$	230 (12)	283 (12)	326 (17)	401 (17)
	115 (<u>6</u>)	117 (<u>5</u>)	249 (<u>13</u>)	235 (<u>10</u>)

Table 5.3: Complexity given by `linSeq`, $ld_I(\mathcal{Z} + \log(d_I))$, with the Conjecture 1 on the sparsity level of the matrices. The first line shows the complexity of the algorithm with respect to the number of rounds of `Anemoi` and the second line shows the number of rounds whose complexity is below the target security level.

applicable than `polyDet` and `SparseFGLM` (the sparsity of the matrices satisfies the condition given in Section 4.2). Moreover, sparse matrices can be represented in the CSR format [32] which maintains the amount of memory usage low with respect to keeping in memory the entire matrix. This, together with the Keller-Gehrig algorithm [28] and the possibility of using the GPU to speed up the matrix-vector multiplication [24] needed to determine the linearly recurring sequence, leads to the possibility of experimentally applying that method also for larger matrices and more `Anemoi` rounds. Although, it must be kept in mind that having more RAM would give the possibility to refine the results given in Conjecture 1, leading to more accurate outcomes. Table 5.3 shows the complexity of the Wiedemann algorithm, with the Conjecture 1, applied to the standard number of rounds of `Anemoi` for $\alpha = 3, 5$.

For sake of completeness, we discuss also the complexities given by the application of the other two methodologies for the univariate polynomial finding: `SparseFGLM` and `polyDet`.

Main differences with the `polyDet` method applied by [3] The main difference between the complexity of the `polyDet` method applied in [3] and ours derives from the generated square multiplication matrices. The one generated by the FreeLunch attack is bigger than the proved value of d_I leading to univariate polynomials of higher degree. Nevertheless, their method to compute the determinant works on the top-right matrix of dimension α^N (notice that $l = 1$ due to the fact that the FreeLunch method is not extending to more than 2 branches) whose columns have maximum degree a value u_0 defined as in [3, Definition 15]. As a consequence, even if the computation of the determinant is slightly better, its degree is higher than d_I (as shown by [3] itself), making the last step of the Gröbner basis cryptanalysis worst. In contrast, in our case, the maximum degree of each column of the matrix $y_{i,0}\mathbf{I}_{d_I} - T_i$ is 1, then the mean is $\frac{1 \cdot d_I}{d_I} = 1$, leading to a complexity of d_I^ω for each input variable $y_{i,0}$ where $1 \leq i \leq l$ and to univariate polynomials of degree d_I. As a result, for a generic value of l, that is we can apply that method to more than 2 branches, the overall complexity of `polyDet` is $\mathcal{O}(l \cdot d_I^\omega)$. Table 5.4 shows the complexities given by `polyDet` when applied to the standard number of rounds of `Anemoi`.

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	136 (21)	165 (21)	177 (20)	197 (19)	241 (37)	291 (37)	320 (36)	363 (35)
	123 ($\underline{19}$)	126 ($\underline{16}$)	124 ($\underline{14}$)	124 ($\underline{12}$)	241 (37)	252 ($\underline{32}$)	249 ($\underline{28}$)	249 ($\underline{24}$)
$l = 2$	183 (14)	221 (14)	232 (13)	271 (13)	287 (22)	347 (22)	374 (21)	437 (21)
	118 ($\underline{9}$)	127 ($\underline{8}$)	125 ($\underline{7}$)	125 ($\underline{6}$)	248 ($\underline{19}$)	253 ($\underline{16}$)	250 ($\underline{14}$)	250 ($\underline{12}$)
$l = 3$	236 (12)	285 (12)	321 (12)	344 (11)	334 (17)	403 (17)	455 (17)	531 (17)
	118 ($\underline{6}$)	119 ($\underline{5}$)	108 ($\underline{4}$)	126 ($\underline{4}$)	255 ($\underline{13}$)	238 ($\underline{10}$)	241 ($\underline{9}$)	250 ($\underline{8}$)

Table 5.4: Complexity given by `polyDet`: d_I^ω, for $\omega = 2.8074$. The first line shows the complexity of the algorithm with respect to the number of rounds of `Anemoi`, the second line shows the number of rounds whose complexity is below the target security level.

On the usage of `SparseFGLM` From experimental results we conjectured that our ideal has Shape form, therefore it makes sense to use the FGLM algorithm. However, due to the fact that the polynomials in the computed GB (Propositions 1 and 2) and the multiplication matrices generated by those equations are extremely sparse, we can apply the sparse version of the $FGLM$ algorithm: `SparseFGLM` [21] (see Section 4.2). Table 5.5 shows the complexities given by `SparseFGLM` when applied to the standard number of rounds of `Anemoi`. Moreover, it shows also the number of rounds whose `SparseFGLM` complexity is below the target security level.

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	147 (21)	176 (21)	189 (20)	208 (19)	253 (37)	304 (37)	333 (36)	376 (35)
	127 ($\underline{18}$)	120 ($\underline{14}$)	125 ($\underline{13}$)	124 ($\underline{11}$)	253 (37)	248 ($\underline{30}$)	252 ($\underline{27}$)	250 ($\underline{23}$)
$l = 2$	194 (14)	232 (14)	243 (13)	282 (13)	299 (22)	360 (22)	387 (21)	450 (21)
	127 ($\underline{9}$)	120 ($\underline{7}$)	116 ($\underline{6}$)	113 ($\underline{5}$)	247 ($\underline{18}$)	248 ($\underline{15}$)	243 ($\underline{13}$)	240 ($\underline{11}$)
$l = 3$	247 (12)	296 (12)	333 (12)	355 (11)	346 (17)	415 (17)	467 (17)	544 (17)
	107 ($\underline{5}$)	104 ($\underline{4}$)	116 ($\underline{4}$)	102 ($\underline{3}$)	247 ($\underline{12}$)	248 ($\underline{10}$)	252 ($\underline{9}$)	229 ($\underline{7}$)

Table 5.5: Complexity given by deterministic SparseFGLM: $n_v \cdot d_I^\omega \cdot \log(d_I)$, for $\omega = 2.8074$. The first line shows the complexity of the algorithm with respect to the number of rounds of `Anemoi`, the second line shows the number of rounds whose complexity is below the target security level.

5.6.3 Design choices (conservative approach)

Usually, from a designer point-of-view, it is better to apply a conservative approach while defining the security of a proposed primitive. Therefore, in Tables 5.6 and 5.7, we show the complexities of `SparseFGLM` and `polyDet` with respect to $\omega = 2$.

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	108 (21)	129 (21)	138 (20)	152 (19)	184 (37)	220 (37)	241 (36)	272 (35)
	108 (21)	123 ($\underline{20}$)	125 ($\underline{18}$)	121 ($\underline{15}$)	184 (37)	220 (37)	241 (36)	249 ($\underline{32}$)
$l = 2$	141 (14)	169 (14)	176 (13)	204 (13)	217 (22)	260 (22)	279 (21)	324 (21)
	122 ($\underline{12}$)	123 ($\underline{10}$)	125 ($\underline{9}$)	114 ($\underline{7}$)	217 (22)	249 ($\underline{21}$)	254 ($\underline{19}$)	249 ($\underline{16}$)
$l = 3$	179 (12)	215 (12)	241 (12)	257 (11)	250 (17)	300 (17)	337 (17)	391 (17)
	122 ($\underline{8}$)	112 ($\underline{6}$)	125 ($\underline{6}$)	121 ($\underline{5}$)	250 (17)	249 ($\underline{14}$)	241 ($\underline{12}$)	234 ($\underline{10}$)

Table 5.6: Complexity given by deterministic SparseFGLM: $n_v \cdot d_I^{\omega} \cdot \log(d_I)$, for $\omega = 2$. The first line shows the complexity of the algorithm with respect to the number of rounds of `Anemoi`, the second line shows the number of rounds whose complexity is below the target security level.

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	97 (21)	117 (21)	126 (20)	140 (19)	171 (37)	207 (37)	228 (36)	259 (35)
	97 (21)	117 (21)	126 (20)	125 ($\underline{17}$)	171 (37)	207 (37)	228 (36)	251 ($\underline{34}$)
$l = 2$	131 (14)	158 (14)	165 (13)	193 (13)	205 (22)	248 (22)	267 (21)	311 (21)
	121 ($\underline{13}$)	124 ($\underline{11}$)	127 ($\underline{10}$)	119 ($\underline{8}$)	205 (22)	248 (22)	254 ($\underline{20}$)	252 ($\underline{17}$)
$l = 3$	168 (12)	203 (12)	229 (12)	245 (11)	238 (17)	287 (17)	324 (17)	379 (17)
	126 ($\underline{9}$)	119 ($\underline{7}$)	115 ($\underline{6}$)	112 ($\underline{5}$)	238 (17)	254 ($\underline{15}$)	248 ($\underline{13}$)	245 ($\underline{11}$)

Table 5.7: Complexity given by `polyDet`: d_I^{ω}, for $\omega = 2$. The first line shows the complexity of the algorithm with respect to the number of rounds of `Anemoi`, the second line shows the number of rounds whose complexity is below the target security level.

Security	128				256			
α	3	5	7	11	3	5	7	11
$l = 1$	26	21	19	16	53	44	39	33
	28	23	21	18	56	46	41	35
	27	20	-	-	54	40	-	-
$l = 2$	13	11	10	8	27	22	20	17
	14	12	11	9	28	23	21	18
	11	9	-	-	22	18	-	-
$l = 3$	9	7	7	6	18	15	13	11
	10	8	7	6	19	16	14	12
	7	6	-	-	14	11	-	-

Table 5.8: Minimum number of rounds required for `Anemoi`: `SparseFGLM` (1st line), `polyDet` (2nd line) and `linSeq` (3rd line)

5.6.4 Suggested number of rounds

Thanks to all those results, we can suggest the minimum number of rounds of `Anemoi` to achieve the claimed security of 128 and 256 bits. In particular, that value will be computed as follows:

$$min\{N \in \mathbb{N} : C_{alg(N)} \geq 2^s\} \qquad with \;\; s \in \{128, 256\}$$

for $C_{alg} \in \{C_{FGLM}, C_{polyDet}, C_{linSeq}\}$. As pointed out in Subsection 5.6.3, we believe it is better to use $\omega = 2$ to define the security of the primitive, as confirmed by the high sparsity of the multiplication matrices involved in the computations of the univariate polynomials. Table 5.8 shows the minimum number of rounds to reach the claimed security level towards the application of the deterministic `SparseFGLM` algorithm, the `polyDet` and the `linSeq` method.

5.7 Experimental results

We experimentally test the methodologies in Section 5.5 on the A_{CICO} Model applied to a small version of `Anemoi`. In particular, those results where achieved for `Anemoi` over $\mathbb{K}_p$ with a 31-bits prime $p = 1481823929$ on a machine with `Intel Xeon(R) Gold 6342 CPU @ 2.80GHz` (32 cores), 48GB RAM and `NVIDIA 48GB L40 GPU` under `Ubuntu 22.04` using `SageMath` [41], `MSOLVE` [6], `Singular` [17], `Macaulay2` [23], `NTL` [35] and `FLINT` [40]. Here, we present the experimental results and the derived conjectures.

Comparison with previous works Previous works [29] [3] were able to apply their model to the case $l = 1$ (`Anemoi` with 2 branches) without mentioning the possibility to extend the attack to more branches. Due to their construction, P_{CICO} and F_{CICO} models are not extendible to more than 2 branches, giving

		$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 11$
$l = 1$	$T_{GB}[ms]$	2.16 (21)	2.33 (21)	2.12 (20)	1.94 (19)
	d_{solv}	9	30	42	66
$l = 2$	$T_{GB}[ms]$	4.57 (14)	4.39 (14)	3.83 (13)	4.02 (13)
	d_{solv}	9	30	42	66
$l = 3$	$T_{GB}[ms]$	11.0 (12)	8.79 (12)	8.74 (12)	8.49 (11)
	d_{solv}	9	30	42	66

Table 5.9: For $l = 1, 2, 3$, the G_{DRL} (if $\alpha = 3$) and $G_{\mathsf{W}_{DRL}}$ (if $\alpha = 5, 7, 11$) computation time in millisecond and the solving degree d_{solv}. We present the GB results with respect to the number of rounds of `Anemoi` (in brackets), but we were able to compute the Gröbner basis for more than 100 rounds.

too high execution timings and complexities. Moreover, neither the Bariant et al. approach [3] is possible to extend to more than two branches due to the impossibility of finding valid (positive) monomial weights for the generation of the FreeLunch system. Therefore, thanks to our methodology we were able to *(a.)* precisely define the complexity of computing the GB for `Anemoi` *(b.)* reduce the complexity of the first two steps of the attack (see Section 5.5) and, then, *(c.)* apply it to more than two branches.

GB computation One of the main advantages of the A_{CICO} model is the possibility to efficiently compute the Gröbner basis. As shown by Propositions 1 and 2, we precisely defined the complexity of computing the GB for `Anemoi`. Table 5.9 shows the computation time (in milliseconds) of the GB with respect to the standard number of rounds of `Anemoi` for $l = 1, 2, 3$ and $\alpha = 3, 5, 7, 11$. It is straightforward to notice how those results are confirming our complexity definition. Moreover, we were able to compute the GB for more than 100 rounds. As a consequence, both theoretical results both experimental results support our choice of making the security of `Anemoi` relying only on the second step of the GB cryptanalysis methodology.

Univariate polynomial finding Tables 5.10, 5.11 and 5.12 show the experimental results for A_{CICO}, $l = 1, 2, 3$ and $\alpha = 3, 5, 7, 11$. As pointed out, the generation of the univariate polynomial is the most involving part of the attack, but it is interesting to notice the differences between the three approaches: `SparseFGLM` , `polyDet` and Wiedemann algorithm. Due to the fact that our polynomials are extremely sparse as well as the multiplication matrices, `SparseFGLM` and Wiedemann algorithm are performing better than `polyDet`. As well explained in Section 4.2, both `SparseFGLM` both Wiedemann algorithm rely on the sparsity of the involved multiplication matrices. Hence, to precisely define their complexity we derived the following conjecture for $\alpha = 3, 5$:

Conjecture 1. Sparsity of the multiplication matrices The level of sparsity of the multiplication matrices of the input variables $y_{i,0}$ for $1 \leq i \leq l$, computed as $\mathcal{Z}/d_I^2$ where $\mathcal{Z}$ is the number of non-zero entries, is approximately given by:

$$Sparsity(T_i) = \begin{cases} 0.9e^{-1.148N}/l & \text{for } \alpha = 3 \\ 0.32e^{-1.06N}/l & \text{for } \alpha = 5 \end{cases} \tag{5.34}$$

Computing the multiplication matrix is one of the most involving parts. Being able to reduce the complexity of this step would give us the possibility to refine our results, extending them to other values of α. For cryptanalysis purposes we are not interested in finding the solutions of all the variables involved in the system. As suggested by the CICO problem, we are interested in finding solutions for the input variables. Therefore, applying the Wiedemann algorithm for each variable results in the possibility of applying the attack to more rounds.

α	N	**Gröbner basis**		**Univariate Poly.**				**Factorization**
		d_{solv}	$T_{GB}[s]$	d_I	$T_{SparseFGLM}[s]$	$T_{polyDet}[s]$	$T_{Wiedemann}[s]$	$T_{Fact}[s]$
3	2	9	0.0010	5^2	0.0237	0.0020	1.0502	0.0004
	3	9	0.0005	5^3	0.0588	8.3710	3.3447	0.0235
	4	9	0.0006	5^4	0.2866	1045.4280	6.8501	0.4097
	5	9	0.00063	5^5	7.7925		20.7408	6.0211
	6	9	0.00066	5^6	1008.2052		110.5109	128.1806
	7	9	0.00076	5^7			750.8691	6443.1831
	8	9	0.00078	5^8			1212.9214	14567.52
5	2	30	0.0012	7^2	2.6680	0.2355	0.2368	0.0012
	3	30	0.0009	7^3	11.5976	239.9753	1.6871	0.4430
	4	30	0.0010	7^4	132.9562	7256.5462	16.0431	28.0277
	5	30	0.0011	7^5	-	-	131.0137	278.0756
	6	30	0.0012	7^6	-	-	1104.3812	3273.5400
7	2	42	0.0013	9^2	0.0541	1.1463	0.4012	0.0033
	3	42	0.0008	9^3	2.021	2348.871	4.7974	2.2697
	4	42	0.0010	9^4	-	-	47.0143	61.6649
	5	42	0.0011	9^5	-	-	1307.0560	903.4305
11	2	66	0.0012	13^2	0.0896	15.8898	0.8268	0.0741
	3	66	0.0008	13^3	29.5011	5491.1795	14.9939	11.9790
	4	66	0.0010	13^4	-	-	214.0266	138.7288

Table 5.10: For $p = 1481823929$, $l = 1$, the computation times for the 3 steps in GB cryptanalysis are listed. As d_I grows, the SparseFGLM, polyDet and Wiedemann algorithm fails due to *memory constraint*. For $\alpha = 3$ the structure of the GB was obtained with DRL monomial ordering, whilst for $\alpha = 5, 7, 11$ the structure of the GB was obtained with W_{DRL}.

α	N	**Gröbner basis**		**Univ. Poly**				**Factorization**
		d_{solv}	$T_{GB}[s]$	d_I	$T_{SparseFGLM}[s]$	$T_{polyDet}[s]$	$T_{Wiedemann}[s]$	$T_{Fact}[s]$
3	2	9	0.00071	5^4	0.6711	1567.6331	8.6146	1.4307
	3	9	0.00082	5^6	1258.0808	-	283.0214	269.6457
5	2	30	0.0013	7^4	185.3975	7389.7569	32.4342	23.8519
	3	30	0.0014	7^6	-	-	5349.7558	3104.4810
7	2	42	0.0013	9^4	722.2398	-	97.7862	65.0040
11	2	66	0.0013	13^4	-	-	486.0422	141.8831

Table 5.11: For $p = 1481823929$, $l = 2$ and $\alpha = 3$, the computation times for the 3 steps in GB cryptanalysis are listed. For $\alpha = 3$ the structure of the GB was obtained with DRL monomial ordering, whilst for $\alpha = 5, 7, 11$ the structure of the GB was obtained with W_{DRL}.

α	N	**Gröbner basis**		**Univ. Poly**				**Factorization**
		d_{solv}	$T_{GB}[s]$	d_I	$T_{SparseFGLM}[s]$	$T_{polyDet}[s]$	$T_{Wiedemann}[s]$	$T_{Fact}[s]$
3	2	9	0.00106	5^6	2072.0338		789.9852	422.6369

Table 5.12: For $p = 1481823929$, $l = 3$ and $\alpha = 3$, the computation times for the 3 steps in GB cryptanalysis are listed. For $\alpha = 3$ the structure of the GB was obtained with DRL monomial ordering.

6 Conclusions

In this thesis we presented the foundations of algebraic cryptanalysis, both from a mathematical both from a computational point of view, then, we provided an in-depth Gröbner basis cryptanalysis of `Anemoi`, one of the most recent Arithmetization-Oriented primitives. The computational part is of high relevance towards the application in the cryptanalysis context. Polynomials represent one of the main tools and, due to the fact that we currently don't know everything about them, they could represent a dangerous attack vector for newer, and also traditional, ciphers. After a detailed walkthrough about the Gröbner basis cryptanalysis methodology and the algorithms that can be used, we tested `Anemoi` a recent permutation primitive that can be used both as a compression function both as a hash function. By using our polynomial representation of `Anemoi` we construct a polynomial system A_{CICO} corresponding to the permutation function. Thanks to the structural properties of the polynomials in the Gröbner basis we were able to show that this modelling choice makes the GB computation step "negligible" with respect to the other steps of the attack, no matter the number of branches ($l \geq 1$).

Additionally, the structural properties of the polynomials constituting the Gröbner basis (for `Anemoi`) allowed us to prove the dimension of the quotient space $\mathbb{K}[x_1, x_2, \ldots, x_n]/I$ which was conjectured in [29]. As a result, we had the possibility to define the complexity of obtaining univariate polynomials which is the other main step in the GB cryptanalysis methodology.

Towards obtaining univariate polynomials, we used efficient eigenvalue method with the aim of solving the system of (polynomial) equations corresponding to the GB of `Anemoi`. More specifically, we used Wiedemann algorithm to obtain the univariate polynomials, improving the existing attacks. Moreover, we experimentally tested our methodology, showing how it led us to break more rounds with respect to previous works. As a result, we defined the number of rounds required by the primitive to guarantee a security of 128 and 256 bits.

The Gröbner basis cryptanalysis methodology that we presented can be applied to other Arithmetization-Oriented primitives, such as Griffin and Poseidon. In particular, the application of the Wiedemann algorithm, instead of using a naive application of basis conversion algorithms and determinant computation, can be extended to other primitives that show the same characteristic w.r.t the sparsity of the multiplication matrices. Moreover, as done for `Anemoi` finding a good model can lead to the complete bypass of the GB computation step of the methodology, and this works in general.

During the last years, algebraic attacks have gained more prominence in the cryptographic community due to their application to AO primitives, as shown by this thesis. Owing to the fact that these strategies did not pose any security threat against traditional block ciphers, the research on this topic was proceeding slowly. However, because this new interest towards the application of algebraic geometry in the context of cryptanalysis could provide newer insights, believing traditional ciphers to be secure against this threat could be a huge mistake. In addition, FHE, MPC, ZK-proofs are becoming more popular both from an academical point of view, both from an industrial point of view, due to the new attention for privacy preserving technologies and zero-trust mechanisms.

A

FreeLunch attack against Anemoi

Definition 1 (FreeLunch System). *Let $P = \{p_1, \dots, p_n\}$ a sequence of polynomials in the ring $\mathbb{K}[x_1, \dots, x_n]$. We say that P is a* ***FreeLunch System*** *if there exist a valid monomial ordering $\prec$ and integers $(\alpha_1, \alpha_2, \dots, \alpha_n)$ such that for all $i \in \{1, n\}$, $LM_{\prec}(p_i) = x_i^{\alpha_i}$. Any monomial order that verifies this property is called FreeLunch order.*

A FreeLunch system P is a Gröbner Basis for the ideal $I = \langle P \rangle$ with respect to any of its FreeLunch orders. Moreover, I is zero-dimensional and the value of d_I is simply the product of the integers $(\alpha_1, \alpha_2, \dots, \alpha_n)$. Moreover, permutation ciphers often show special structures which give the possibility to convert them to Triangular systems and, then, to FreeLunch system.

Definition 2 (Triangular System). *Let $P = \{p_1, \dots, p_n\}$ a sequence of polynomials in the ring $\mathbb{K}[x_1, \dots, x_n]$. We say that P is a* ***triangular system*** *if there exist polynomials $q_1, \dots, q_n$, integers $(\alpha_1, \alpha_2, \dots, \alpha_n)$ and $(c_1, c_2, \dots, c_n) \in F \setminus \{0\}$ such that*

$$\begin{cases} p_i = c_i x_i^{\alpha_i} + q_i(x_1, \dots, x_{i-1}) & \textit{for } 2 \leq i \leq n \\ g = c_1 x_1^{\alpha_1} + q_1(x_1, \dots, x_n) \end{cases}$$

A triangular system is, by definition, a FreeLunch system, Therefore, finding a valid monomial ordering for that triangular system corresponds to find a suitable FreeLunch order.

General inequalities for triangular orders From definition 2, we define the following inequalities which guarantee that the generated monomial ordering is a triangular order $\prec_T$.

$$wt(x_i) > \frac{wt(LM_{\prec_T}(q_i))}{\alpha_i} \qquad for \quad 1 \leq i \leq n$$

A.1 Freelunch system attack methodology

[3] shows how to exploit this structure to define a competitive method for finding solutions to x_1. The proof of the following theorem which will be used to define the attack methodology, is given in their article.

Theorem 1. *Given a FreeLunch system P, a FreeLunch order $\prec$ and the multiplication matrix T_1 of the variable x_1, there exists an algorithm to compute a solution for x_1 with time complexity:*

$$\mathcal{O}\left(\alpha_1\left(\prod_{i=1}^{n-1}\alpha_i\right)^{\omega}\right)$$

Then, the methodology is defined as follows:

1) Compute the multiplication matrix T_1 of the variable x_1
2) Compute the determinant of the matrix $x_1\mathbf{I} - T_1$
3) Factorize the obtained univariate polynomial

Complexity analysis As for the first step, it is hard to define its complexity. As shown by the experiments, it can be costlier than the other two steps. The procedure returning the univariate polynomial (the computation of the determinant) has a complexity of $\mathcal{O}(d_I d_1^{\omega-1})$ with the algorithm of Labahn et al. [30]. The last step deals with the factorization of a univariate polynomial of degree d_I which costs $\mathcal{O}(d_I)$ (negligible w.r.t. the previous step).

A.2 `Anemoi` algebraic cryptanalysis through Freelunch systems

Following [3], as a starting point we will use the equations given by the P_{CICO} defined in Subsection 5.4.1.

$$p_{i,r} = E(s_{i,r}) - f_{i,r} + Q_\gamma(g_{i,r}) = s_{i,r}^{\alpha} + \beta g_{i,r}^2 + \gamma - f_{i,r} \tag{A.1}$$

where $p_{i,r} \in F_p[y_{1,0}, \dots, y_{l,0}, s_{1,1}, \dots, s_{l,1}, \dots, s_{1,r-1}, \dots, s_{l,r-1}, s_{i,r}]$.

$$x_{i,N+1} := L(x_{1,N}, x_{2,N}, \dots, x_{l,N}, y_{1,N}, \dots, y_{l,N}) = 0 \tag{A.2}$$

where $x_{i,N+1} \in F_p[y_{1,0}, \dots, y_{l,0}, s_{1,1}, \dots, s_{l,1}, \dots, s_{1,N}, \dots, s_{l,N}]$.

To make the system composed by those equations a FreeLunch system, we need to find suitable monomial weights such that:

$$LM(p_{i,r}) = s_{i,r}^{\alpha} \qquad\qquad LM(x_{i,N+1}) = y_{i,0}^{u}$$

for each $1 \le i \le l$ and $1 \le r \le N$, where u is an exponent as defined in a while.

A.2.1 Weights for $p_{i,r}$

From Subsection 5.4.1 we know that $\mathsf{deg}(p_{i,r}) = \max\{\alpha, 2r\}$. In particular, we can define $r_\alpha := min\{r \in \mathbb{N} \mid 2r \ge \alpha\}$ to denote the first round such that the maximum degree of $p_{i,r}$ moves from α to $2r$. For each $p_{i,r}$, we can say that:

$$deg(p_{i,r}) = \begin{cases} \alpha & \text{for } 1 \le r < r_\alpha \\ 2r & \text{for } r_\alpha \le r \le N \end{cases}$$

Therefore, we can notice that:

$$LM(p_{i,r}) = \begin{cases} s_{i,r}^{\alpha} & \text{for } 1 \le r < r_\alpha \\ LM(Q_\gamma(g_{i,r})) & \text{for } r_\alpha \le r \le N \end{cases}$$

To discover what is the leading monomial of $Q_\gamma(g_{i,r})$ we need to understand what is the leading monomial of $x_{i,r-1}$ for each $1 \le i \le l$. From Equation 5.8 we can say that the leading monomial of $x_{i,r}$ is either $s_{i,r}^2$ or $LM(g_{i,r}s_{i,r})$. In particular, taking into account the consideration given in Subsection 5.4.1:

$$LM(x_{i,r}) = \begin{cases} s_{i,1}^2 \quad or \quad s_{i,1}LM(g_{i,1}) & \text{for } r = 1 \\ LM(g_{i,r}s_{i,r}) = s_{i,r}LM(g_{i,r}) & \text{for } 2 \le r \le N \end{cases} \tag{A.3}$$

As regards $g_{i,r}$, we can say that its leading monomial depends on the leading monomials of $x_{1,r-1}, \dots, x_{l,r-1}, y_{1,r}$ due to the application of the linear layer before the Flystel evaluation. Moreover, thanks to Equation 5.9 we know that $\mathsf{deg}(LM(x_{i,r-1})) > \mathsf{deg}(LM(y_{i,r-1}))$. Hence,

$$LM(g_{i,r}) = LM\left(\sum_{i=1}^{l} k_i x_{i,r-1} + \sum_{i=1}^{l} b_i y_{i,r-1}\right)$$

where k_i and b_i are coefficients in F_p depending on the application of the linear layer L. There will be more than one monomial with the maximum degree, then the leading monomial will depend on the chosen monomial ordering. In general, by following the considerations above from the beginning to the r-th round, the leading monomial of $g_{i,r}$ follows the given structure:

$$y_{t,in} \cdot \prod_{j=1}^{r-1} s_{c,j}$$

where t, c can be whatever value from 1 to l. To determine what is the leading monomial among those l^r possibilities, we would need to define a variable and a monomial ordering. Thanks to the statement above, we are finally able to define $LM(p_{i,1})$. For sake of simplicity, we will use $LM(g_{i,r}) = y_{i,0} \cdot \prod_{j=1}^{r-1} s_{i,j}$.

$$LM(p_{i,r}) = \begin{cases} s_{i,r}^{\alpha} & \text{for } 1 \le r < r_\alpha \\ Q_\gamma(y_{i,0} \cdot \prod_{j=1}^{r-1} s_{i,j}) = y_{i,0}^2 \cdot \prod_{j=1}^{r-1} s_{i,j}^2 & \text{for } r_\alpha \le r \le N \end{cases}$$

By considering that result, when we deal with a number of rounds greater or equal to r_α, the equations $p_{i,r}$ are not already in FreeLunch form. Than, we need to find positive variable weights such that the weighted monomial ordering so defined makes $s_{i,r}^{\alpha}$ the leading monomial of each equation $p_{i,r}$. Therefore our weights must satisfy the following inequality:

$$\alpha \cdot wt(s_{i,r}) > 2 \cdot (wt(y_{i,in}) + wt(s_{i,1}) + \dots + wt(s_{i,r-1})) \tag{A.4}$$

for all $1 \le i \le l$ and for all $1 \le r \le N$.

A.2.2 Weights for $x_{i,N+1}$

By Equation A.3 and the considerations on the leading monomial of $g_{i,r}$ we can say that:

$$LM(x_{i,r}) = \begin{cases} s_{i,1}^2 \quad or \quad s_{i,1}y_{i,0} & \text{for } r = 1 \\ LM(g_{i,r}s_{i,r}) = y_{i,0} \cdot \prod_{j=1}^{r} s_{i,j} & \text{for } 2 \le r \le N \end{cases} \tag{A.5}$$

Now, we are considering $x_{i,N+1}$ which, thanks to the linear layer, is a linear combination of all the $x_{i,N}$ and $y_{i,N}$ for $1 \le i \le l$. Therefore, its leading monomial is defined as in Equation A.5.

As a consequence, there is no possible choice of monomial ordering and weights where $x_{i,r}$ will have a leading monomial in only $y_{i,0}$. Moreover, by fixing a variable order and a monomial ordering, and by considering that all the l last equations $x_{i,N+1}$ will have the same monomials, their leading monomial will always be the same. Bariant et al. [3] proposed a way to force the last equation to be in FreeLunch form in the case $l = 1$. Unfortunately, that is not possible with $l > 1$ due to the impossibility of cancelling high order terms. We will briefly present their technique, then we will show why it is not exploitable for more than 2 branches.

A.2.3 Forcing the last equation to be in FreeLunch form

We will refer to the case of $l = 1$. In order to force the last equation to be in FreeLunch form, we will multiply $x_{1,N+1}$ by suitable monomials in $s_{1,1}, s_{1,2}, \dots, s_{1,N}$ that will lead to a reduction by the polynomials $p_{1,1}, \dots, p_{1,N}$. This process will generate a new polynomial $x^*_{1,N+1}$ whose leading monomial will be only in $y_{1,in}$. The new polynomial will have more solutions than the original one, but it will share the ones of the original equation. From a mathematical point of view, given the ideal I generated by the set of polynomials $\{p_{1,1}, \dots, p_{1,N}, x_{1,N+1}\}$ we will generate a new ideal J defined by the set of polynomials $\{p_{1,1}, \dots, p_{1,N}, x^*_{1,N+1}\}$ such that $\mathbb{V}(I) \subset \mathbb{V}(J)$. To do what we have said so far, we need to define a way to predict the powers of $s_{1,r}$ we will use in the multiplication of $x_{1,N+1}$ prior to the reduction by $p_{1,1}, \dots, p_{1,N}$.

Definition 3. *[3] We define two integer sequences $\{u_i\}_{0 \le i \le r}$ and $\{k_j\}_{1 \le j \le r}$, where $u_r = 1$ and the remaining sequences are recursively defines as follows:*

- *k_i is the unique integer $0 \le k_i < \alpha$ such that $k_i \equiv -u_i \mod \alpha$*
- *$u_i = u_{i+1} + 2\frac{(u_{i+1}+k_{i+1})}{\alpha}$*

Whilst k_j will represent the power of $s_{1,j}$ that we will use in the multiplication of $x_{1,N+1}$ for all $\{k_j\}_{1 \le j \le N}$, u_i will represent the power of the variable $y_{1,in}$ prior to the multiplication by $s_{1,i}^{k_i}$ and the following reduction. Moreover, notice that k_0 is not defined, whilst u_0 will represent the higher degree reachable by the target variable $y_{1,in}$ after all the reductions. That is important because we need to use u_0 to define the last inequality and then the weights that gives the possibility to set $y_{1,in}^{u_0}$ as the leading monomial of $x^*_{1,N+1}$ after all the reductions. But, to define that inequality, we need to figure out what is the $LM(x^*_{1,N+1})$ and, in particular, what is $LM_2(x^*_{1,N+1})$, that is the second monomial after the leading one w.r.t the monomial ordering we are going to define and that we already partially defined in the previous subsections with Equation A.4.

A.2.4 Impossibility to extend the technique to more than 2 branches

The polynomials $p_{i,r}$ can always be made in FreeLunch form, which means that we are always able to solve the corresponding inequalities. Unfortunately, the issue comes from the attempt to force the last l equations to be in FreeLunch form. Due to the presence of the diffusion layer and the pseudo-hadamart transform, there is no possible reduction which is able to remove monomials of the type $s_{i,r}^{\alpha-1}s_{i+1,r}^{2}$ or $s_{i,r}^{\alpha-1}s_{i-1,r}^{2}$ which are created from the attempt to force the system to be in Freelunch form. Another way to see the impossibility to apply such construction, is trying to define an inequality for each variable $y_{i,0}$ such that:

$$\alpha \cdot u_0 \cdot wt(y_{i,0}) > wt(y_{i,in}) + 2 \cdot (wt(s_{i,1}) + \cdots + wt(s_{i,r-1})) \tag{A.6}$$

for all $1 \leq i \leq l$ and for all $1 \leq r \leq N$.

The so-defined system of inequalities has no solutions in $\mathbb{R}_{\geq 0}$ and, by Definition 11, a weighted monomial order is valid only for values greater or equal to 0.

B

Additional Algebraic cryptanalysis methods

B.1 Interpolation attack

Each symmetric cipher involves a non-linear operation which algebraically increases the degree of the generated polynomials. If the round function, and in particular the non-linear part, can be expressed as a low-degree polynomial, the interpolation attacks could act as an important security threat for the analyzed primitive. As an example, if the ciphertexts generated by the primitive can be expressed as evaluations of low-degree polynomials of the plaintext, then an attacker could obtain several plaintext-ciphertext couples and interpolate the polynomial (e.g. through Lagrange interpolation). That polynomial would be the algebraic representation of the primitive, meaning that all new plaintext-ciphertext couples that are intercepted should satisfy the relation.

Let us assume to have multiple plaintext-ciphertext couples (p, c). Let $p_1, p_2, \ldots, p_n$ be the binary representation of the plaintext p, where $n = \log_2(p)$. Hence, we can represent p as a multivariate polynomial $q(x) \in GF(2)[x_1, \ldots, x_n]$ where x_i is the i-th plaintext bit. The interpolation attack is applicable if the non-linear order d of the ciphertext polynomial (from the reduced cipher) as a function of the plaintext is low. Since d is low, the distinguisher can be used to recover the last round-key.

The interpolation attack is usually applied by using univariate polynomials. In this way p is expressed as a polynomial $q(x) \in GF(2^n)$ of the plaintext bits. If the polynomial is sparse, or it has low degree, given a collection of plaintext-ciphertext couples, one can construct that polynomial by interpolation. This would generate a polynomial based encryption algorithm that resembles the original cipher without knowing the secret key. As a consequence, the attacker can exploit the polynomial to recover the last round key.

The attack can be generalized to deal with block of bits, meaning that each block of s bits would be identified by a different variable of a multivariate polynomial. The higher-order differential attack is a special case where $s = m$, the polynomials involved are univariate and the way of generating such polynomial is slightly different.

B.2 Higher-order differential attack

As previously stated, the higher-order differential attack is a special case of the interpolation attack. If the algebraic degree d of a function f is sufficiently low, then, if the sum of the evaluation of f on all the elements of a certain affine vector space is 0, with high probability it is possible to recover the correct value of the secret key. Finding such a distinguisher is possible if the non-linear transformation has very low algebraic degree. For example, most of the proposed Arithmetization-Oriented primitives generate very high degree, making the attack infeasible. To show how this attack works, we propose the following example.

Example 1. In this example we are going to define the steps of the higher-order differential attack on a toy SPN cipher. Consider a simple substitution-permutation network cipher P with block size of $2n$ bits. Assume the first half of those input bits (denoted as I_{left}) to be constant and consider the remaining n bits (denoted as $I_{right} = (x_1, x_2, \ldots, x_n)$) as the variables of the polynomials we are going to consider. Each bit of the ciphertext $C = (c_1, c_2, \ldots, c_{2n})$, which we already know, can be expressed as a multivariate polynomial $p_i \in \mathbb{F}_2[x_1, x_2, \ldots, x_n]$ for each $1 \le i \le 2n$.

If $\mathsf{deg}(p_i) \le d$ for all $1 \le i \le 2n$, then, we can define $\mathcal{L}_d$ as the $(d+1)$-dimensional subspace of $\mathbb{F}_2^n$ such that $\sum_{I_{right} \in \mathcal{L}_{d+1}} p(I_{right}) = c$, where c is a constant for any space parallel to $\mathcal{L}_{d+1}$ and p is the function which represents the cipher encryption up to the last but one round.

The vectors space $\mathcal{L}_{d+1}$ can be represented as a full rank matrix M of dimension $(d+1) \times n$ over $\mathbb{F}_2$. Therefore, if and only if $\mathsf{deg}(p) \le d$ we can define

$$\sigma(w) = \sum_{I_{right} \in \mathcal{L}_{d+1}} p(I_{right} + w) = (0, 0, \ldots, 0) \text{ for all } w \in \mathbb{F}_2^n.$$

If we are able to find such a zero distinguisher, we are able to find the correct key with high probability. Let f be the round function and f^{-1} the corresponding inverse, a sketch of the process is given in Algorithm 15.

Algorithm 15 Sketch of the higher-order differential attack. The attack can be performed by running the algorithm with multiple values of w and then, compute the intersection between the found key sets.

```
procedure HIGHER-ORDER-ATTACK($I_{left}, w \in \mathbb{F}_2^n, M \in \mathbb{F}_2^{(d+1)\times n}$)    ▷
    ciphertexts = []
    for $a \in \mathbb{F}_2^{d+1}$ do
        $I_{left} = aM + w$
        ciphertexts = ciphertexts + [(P(I_{left}||I_{right})]
    possibleKeys = []
    for $k$ do                                                    ▷ All the possible keys
        $\sigma_w = 0$
        for $c \in$ ciphertexts do
            $\tilde{c} = f^{-1}(k, c)$
            $\sigma_w = \sigma_w + c$
        if $\sigma_w = 0$ then
            possibleKeys = possibleKeys + [$k$]
    return possibleKeys
```

Bibliography

[1] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic cryptanalysis of stark-friendly designs: Application to marvellous and mimc. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.

[2] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the f5 gröbner basis algorithm. *Journal of Symbolic Computation*, 70:49–70, 2015.

[3] Augustin Bariant, Aurélien Boeuf, Axel Lemoine, Irati Manterola Ayala, Morten Øygarden, Léo Perrin, and Håvard Raddum. The algebraic freelunch efficient gröbner basis attacks against arithmetization-oriented primitives. Cryptology ePrint Archive, Paper 2024/347, 2024. `https://eprint.iacr.org/2024/347`.

[4] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970.

[5] Elwyn R Berlekamp. *Algebraic coding theory (revised edition).* World Scientific, 2015.

[6] Jérémy Berthomieu, Christian Eder, and Mohab Safey El Din. msolve: A Library for Solving Polynomial Systems. In *2021 International Symposium on Symbolic and Algebraic Computation*, 46th International Symposium on Symbolic and Algebraic Computation, pages 51–58, Saint Petersburg, Russia, July 2021. ACM.

[7] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 507–539, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

[8] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions:anemoi permutations and jive compression mode. Cryptology ePrint Archive, Paper 2022/840, 2022.

[9] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. A zero-dimensional gröbner basis for aes-128. In *Fast Software Encryption: 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers 13*, pages 78–88. Springer, 2006.

[10] Stanislav Bulygin and Michael Brickenstein. Obtaining and solving systems of equations in key variables only for the small variants of aes. *Mathematics in Computer Science*, 3:185–200, 2010.

[11] Alessio Caminata and Elisa Gorla. Solving degree, last fall degree, and related invariants. Cryptology ePrint Archive, Report 2021/1611, 2021. `https://eprint.iacr.org/2021/1611`.

[12] Alessio Caminata and Elisa Gorla. *Solving Multivariate Polynomial Systems and an Invariant from Commutative Algebra*, page 3–36. Springer International Publishing, 2021.

[13] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[14] Stéphane Collart, Michael Kalkbrener, and Daniel Mall. Converting bases with the gröbner walk. *Journal of Symbolic Computation*, 24(3-4):465–469, 1997.

[15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms.* Undergraduate Texts in Mathematics. Springer, fourth edition, 2015.

[16] David A. Cox, John B. Little, and Donal O'Shea. *Using Algebraic Geometry*, volume 185 of *Graduate Texts in Mathematics.* Springer, first edition, 1998.

[17] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. Singular 4-3-0 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2022.

[18] Michele Elia and Davide Schipani. Improvements on cantor-zassenhaus factorization algorithm. *arXiv preprint arXiv:1012.5322*, 2010.

[19] Jean Charles Faugere. A new efficient algorithm for computing gröbner bases without reduction to zero (f 5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[20] Jean-Charles Faugere, 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.

[21] Jean-Charles Faugère and Chenqi Mou. Sparse fglm algorithms. *Journal of Symbolic Computation*, 80:538–569, May 2017.

[22] Jean-Charles Faugére. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[23] Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `http://www2.macaulay2.com`.

[24] Joseph L. Greathouse and Mayank Daga. Efficient sparse matrix-vector multiplication on gpus using the csr storage format. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 769–780, 2014.

[25] Gavin Harrison, Jeremy Johnson, and B. David Saunders. Probabilistic analysis of wiedemann's algorithm for minimal polynomial computation. *Journal of Symbolic Computation*, 74:55–69, 2016.

[26] Jérémy Jean. TikZ for Cryptographers. `https://www.iacr.org/authors/tikz/`, 2016.

[27] Erich Kaltofen and Victor Shoup. Subquadratic-time factoring of polynomials over finite fields. In *27th ACM STOC*, pages 398–406, Las Vegas, NV, USA, May 29 – June 1, 1995. ACM Press.

[28] Walter Keller-Gehrig. Fast algorithms for the characteristics polynomial. *Theoretical Computer Science*, 36:309–317, 1985.

[29] Katharina Koschatko, Reinhard Lüftenegger, and Christian Rechberger. Exploring the six worlds of gröbner basis cryptanalysis: Application to anemoi. Cryptology ePrint Archive, Paper 2024/250, 2024. `https://eprint.iacr.org/2024/250`.

[30] George Labahn, Vincent Neiger, and Wei Zhou. Fast, deterministic computation of the hermite normal form and determinant of a polynomial matrix, 2017.

[31] Vincent Neiger and Clément Pernet. Deterministic computation of the characteristic polynomial in the time of matrix multiplication. *Journal of Complexity*, 67:101572, 2021.

[32] Tomáš Oberhuber, Atsushi Suzuki, and Jan Vacata. New row-grouped csr format for storing the sparse matrices on gpu with implementation in cuda, 2010.

[33] Shojiro Sakata. *The BMS Algorithm*, pages 143–163. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[34] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949. See, in particular, page 704.

[35] Victor Shoup. Ntl: A library for doing number theory. `https://libntl.org/`.

[36] Victor Shoup. Factoring polynomials over finite fields: Asymptotic complexity vs. reality. 1993.

[37] Victor Shoup. Ntl vs flint. `https://libntl.org/benchmarks.pdf`, 2021.

[38] Matthias Johann Steiner. Solving degree bounds for iterated polynomial systems. *IACR Trans. Symmetric Cryptol.*, 2024(1):357–411, 2024.

[39] Arne Storjohann. High-order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3-4):613–648, 2003.

[40] The FLINT team. *FLINT: Fast Library for Number Theory*, 2023. Version 3.0.0, `https://flintlib.org`.

[41] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, YYYY. `https://www.sagemath.org`.

[42] Ralf-Philipp Weinmann. Evaluating algebraic attacks on the aes. *Diplom thesis, Technische Universität Darmstadt*, 2003.

[43] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.

[44] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 887–898, New York, NY, USA, May 19–22, 2012. ACM Press.