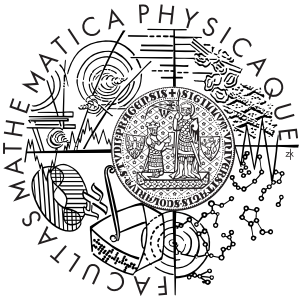
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Pavel Hubáček

Gröbnerovy báze v kryptografii

Katedra algebry

Vedoucí diplomové práce: RNDr. David Stanovský, Ph.D.
Studijní program: matematické metody informační bezpečnosti

2010

Chtěl bych poděkovat svým rodičům za neustálou podporu a zázemí, které mi poskytují při studiu. Rád bych také poděkoval vedoucímu mé práce za odborné vedení.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze 5. srpna 2010 Pavel Hubáček

Contents

Název práce: Gröbnerovy báze v kryptografii
Autor: Pavel Hubáček
Katedra (ústav): Katedra algebry
Vedoucí diplomové práce: RNDr. David Stanovský, Ph.D.
e-mail vedoucího: stanovsk@karlin.mff.cuni.cz

Abstrakt: Předložená práce studuje využití Gröbnerových bází v kryptografii, a to speciálně při kryptoanalýze blokových šifer. Nejprve seznamujeme se základními pojmy teorie Gröbnerových bází a metodou pro jejich nalezení, kterou je Buchbergerův algoritmus. Je vysvětlen princip řešení soustav polynomiálních rovnic pomocí vhodných Gröbnerových bází. Následně je věnována pozornost moderním algoritmům pro nalezení Gröbnerovy báze, jež Buchbergerův algoritmus vylepšují. V poslední části jsou shrnuty dosavadní výsledky dosažené v kryptografii pomocí metod založených na Gröbnerových bázích a je představen pojem algebraické kryptoanalýzy. Ta převádí problém prolomení kryptosystému na problém nalezení řešení soustavy polynomiálních rovnic nad konečným tělesem. Na příkladech je vysvětleno jak konstruovat soustavy polynomů více proměnných popisující blokové šifry a jsou prezentovány výsledky praktických pokusů s takovými soustavami.

Klíčová slova: algebraická kryptoanalýza, řešení soustav polynomiálních rovnic, počítačová algebra

Title: Gröbner Bases in Cryptography
Author: Pavel Hubáček
Department: Department of Algebra
Supervisor: RNDr. David Stanovský, Ph.D.
Supervisor's e-mail address: stanovsk@karlin.mff.cuni.cz

Abstract: The thesis focuses on the use of Gröbner bases in cryptography and especially on applications in cryptanalysis of block ciphers. Some elementary concepts from the theory of Gröbner bases are introduced together with Buchberger's algorithm, a method for constructing such bases. The principle of solving of polynomial systems using suitable Gröbner bases is explained. This is followed by presentation of modern algorithms that improve the Buchberger's algorithm. In the last part of the thesis present results achieved by Gröbner bases are summarised and the notion of algebraic cryptanalysis is introduced. In algebraic cryptanalysis we transform breaking of given cryptosystem into a problem of solving polynomial equations over some finite field. Examples of polynomial descriptions of block ciphers are provided together with some experimental result on arising polynomial systems.

Keywords: algebraic cryptanalysis, solving systems of polynomial equations, computer algebra

Chapter 1

Introduction

The concept of Gröbner bases was introduced by Bruno Buchberger [12] in 1965 and soon became one of the main tools for solving problems connected to polynomial ideals. There are many applications of Gröbner bases in algebraic geometry, elimination theory, coding theory, automated theorem proving, or robotics. The applicability of the concept might be quite astonishing. Still, it is merely connected to one's ability to transform given problem into a polynomial setting.

Gröbner bases are especially practical when trying to solve some system of polynomial equations. We will focus on applications of those properties in cryptography. The idea of use of solving polynomial equations in cryptography is not at all new. In Shannon's paper about secrecy systems [30] from 1949, the author states that:

> *Thus, if we could show that solving a certain system requires at least as much work as solving a system of simultaneous equations in a large number of unknowns, of a complex type, then we would have a lower bound of sorts for the work characteristic.*

However, this was impractical for a long time. In the last decades, remarkable advancements were achieved both in availability of computing power and in research of algorithms for computer algebra. This gave rise to the so called *algebraic cryptanalysis*, i.e. representing the task of breaking given cryptosystem as a problem of solving system of multivariate polynomial equations over a finite field.

The purpose of this thesis is to provide a theoretical background for algebraic cryptanalysis using Gröbner bases and equip prospective reader with sufficient understanding of the involved advanced Gröbner basis techniques. Thesis is therefore divided into two parts. The first half is a survey of theory of Gröbner bases and presentation of recent algorithms; the second half explains the ideas of algebraic cryptanalysis and illustrates them on examples of educational and real-life block ciphers. The contribution of the thesis should be a comprehensive overview of subject that is fragmented into journal papers and conference proceedings at the present time.

Chapter 2

Gröbner Bases and Solving Polynomial Equations

It should be stated that the original ideas of Buchberger for commutative polynomial rings were soon modified to suit different non-commutative structures. There are some applications of such structures in cryptography; for examples see Ackermann and Kreuzer [1]. However, the theory for non-commutative structures is omitted, because it goes beyond the scope of this thesis. In this chapter, we will present some basic definitions and try to explain the concept of solving polynomial equations using Gröbner bases.

2.1 Gröbner Bases Fundamentals

In this section, we would like to give a minimal amount of theory necessary in later sections to develop and understand the advanced algorithms for finding Gröbner bases. One of so far best introductory texts about Gröbner bases can be found in Cox, Little and O'Shea [22]. Depending on desired applications, one might consider other sources such as Adams and Loustaunau [2], Becker and Weispfenning [8] or Eisenbud [23]. The following section uses notation and theory from both Cox, Little and O'Shea [22] and Winkler [34].

The theory gives us a better insight into the structure and properties of polynomial rings and their ideals. We will denote X a set of variables (e.g. $\{x_1, x_2, x_3\}$). By $[X]$, we will understand the set of all *monomials* over X, i.e. $[x_1, \ldots, x_n] = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} : \alpha_1, \ldots, \alpha_n \in \mathbb{Z}_{\geq 0}^+\}$. For k, a commutative field, the *polynomial ring* $k[x_1, \ldots, x_n]$ over k will be sometimes abbreviated as $k[X]$ when the precise number of variables is not important. The elements of polynomial ring $k[x_1, \ldots, x_n]$ are finite k-linear combinations of monomials.

Definition 2.1. *Let R be ring. We call $I \subseteq R$ an* ideal *of R iff*

1. $\forall f, g \in I : f + g \in I$,
2. $\forall f \in I, \forall a \in R : a \cdot f \in I$.

Next definition introduces very important property of some rings, which will be crucial in later theorems.

Definition 2.2. *Let R be a ring. We say that R is* Noetherian *iff there exists no strictly ascending chain of ideals in R.*

There are some other equivalent definitions of ring to be Noetherian:

- Every ideal I in R is finitely generated, i.e. there exist elements $a_1, \ldots, a_n$ in I such that $I = Ra_1 + \ldots + Ra_n$.
- Every non-empty set of ideals of R, partially ordered by inclusion, has a maximal element with respect to set inclusion.

Especially for a commutative ring to be Noetherian it suffices that every prime ideal I of the ring (i.e. for any $a, b \in R$, if ab is in I, then at least one of a and b is in I) is finitely generated. We will denote an ideal generated by $a, b \in R$ by $\langle a, b \rangle \subset R$.

The following theorem is known as *Hilbert basis theorem* and it is a proposition about structure within polynomial rings.

Theorem 2.3 (Hilbert basis theorem)**.** *Let R be a Noetherian ring. Then $R[x]$ is also Noetherian.*

Proof. The theorem with proof can be found in Cox, Little and O'Shea [22] Chapter 2, §5. □

As a consequence, we can see that every ideal in polynomial ring is finitely generated. More geometrically, the theorem asserts that any infinite set of polynomial equations may be associated to a finite set of polynomial equations with precisely the same solution set (algebraic variety; denoted $\mathrm{V}(I)$ for an ideal $I \subseteq k[X]$). This introduces questions about properties of generating sets of polynomial ideals. It is a well accepted fact that Gröbner bases might be one of the optimal choices of such generating sets.

To develop the theory, we need to introduce some ordering on the set of monomials in the polynomial ring. Following definition gives us so called *admissible orderings*, which have the desired properties.

Definition 2.4. *Let $>$ be a total ordering on $[X]$ that is compatible with the monoid structure, i.e.,*

1. *$t > x_1^0 \ldots x_n^0 = 1$ for all $t \in [X] \setminus \{1\}$, and*
2. *$s > t \Rightarrow su > tu$ for all $s, t, u \in [X]$.*

We call such an ordering $>$ on $[X]$ an admissible ordering.

We will later see that choice of suitable ordering plays a central role in the theory of Gröbner bases. Therefore, we would like to show some examples of common admissible orderings.

Definition 2.5 (Lexicographic ordering)**.** *Let* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *and* $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$*. We say* $\alpha >_{\text{lex}} \beta$ *if, in the vector difference* $\alpha - \beta \in \mathbb{Z}^n$*, the left-most nonzero entry is positive. We will write* $x^\alpha >_{\text{lex}} x^\beta$ *if* $\alpha >_{\text{lex}} \beta$*.*

Some practical examples follow:
$(1, 2, 0) >_{\text{lex}} (0, 3, 4)$ since $\alpha - \beta = (1, -1, -4)$.
$(3, 2, 4) >_{\text{lex}} (3, 2, 1)$ since $\alpha - \beta = (0, 0, 3)$.
The variables $x_1, \ldots, x_n$ are ordered in the usual way by the lexicographic ordering:

$$(1, 0, \ldots, 0) >_{\text{lex}} (0, 1, 0, \ldots, 0) >_{\text{lex}} \cdots >_{\text{lex}} (0, \ldots, 0, 1),$$

so $x_1 >_{\text{lex}} x_2 >_{\text{lex}} \cdots >_{\text{lex}} x_n$. In the two variable case with variables x and y, we have:

$$\cdots > x^2 > \cdots > y^2x > yx > x > \cdots > y^3 > y^2 > y > 1.$$

The lexicographic ordering might have some desirable properties, but sometimes we would like to take into account also the degree of monomials. Following ordering is so called *graded lexicographic ordering.*

Definition 2.6 (Graded lexicographic ordering)**.** *Let* $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$*. We say* $\alpha >_{\text{grlex}} \beta$ *if*

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

We see that $>_{\text{grlex}}$ orders by total degree first, then "breaks ties" using the lexicographic ordering. Here are some examples:
$(1, 2, 3) >_{\text{grlex}} (3, 2, 0)$ since $|(1, 2, 3)| = 6 > |(3, 2, 0)| = 5$.
$(1, 2, 4) >_{\text{grlex}} (1, 1, 5)$ since $|(1, 2, 4)| = |(1, 1, 5)|$ and $(1, 2, 4) >_{\text{lex}} (1, 1, 5)$.

In the two variable case with variables x and y, we get:

$$\cdots > x^3 > x^2y > xy^2 > y^3 > x^2 > xy > y^2 > x > y > 1.$$

The $>_{\text{grlex}}$ ordering is obviously an admissible ordering.

Another ordering on monomials is the *graded reverse lexicographic ordering.* It might be less intuitive than the two previous orderings; nevertheless, it is very useful for computations.

Definition 2.7 (Graded reverse lexicographic ordering)**.** *Let* $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$*. We say* $\alpha >_{\text{grevlex}} \beta$ *if*

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

and, in $\alpha - \beta \in \mathbb{Z}^n$*, the right-most nonzero entry is negative.*

Like $>_{\text{grlex}}$, $>_{\text{grevlex}}$ ordes by total degree, but it "breaks ties" in a different way. For example:
$(4, 7, 1) >_{\text{grevlex}} (4, 2, 3)$ since $|(4, 7, 1)| = 12 > |(4, 2, 3)| = 9$.

$(1,5,2) >_{\text{grevlex}} (4,1,3)$ since $|(1,5,2)| = |(4,1,3)|$ and $\alpha - \beta = (-3,4,-1)$.
Note that $>_{\text{lex}}$, $>_{\text{grlex}}$ and $>_{\text{grevlex}}$ give the same ordering on the variables. That is,

$$x_1 > x_2 > \cdots > x_n.$$

Given an admissible ordering, we can examine polynomials in more detail.

Definition 2.8. *Let $s \in [X]$ be a monomial, $f \in k[X]$ a non-zero polynomial and F a subset of $k[X]$.*
Given an admissible ordering $>$ we define:

1. $\text{coeff}(f,s)$*denotes the coefficient of s in f.*
2. $\text{LM}(f) = \max_>\{t \in [X] : \text{coeff}(f,t) \neq 0\}$, leading monomial *of f,*
3. $\text{LC}(f) = \text{coeff}(f, \text{LM}(f))$, leading coefficient *of f,*
4. $\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f)$, leading term *of f,*
5. $\text{LM}(F) = \{\text{LM}(f) : f \in F \setminus \{0\}\}$,
6. $\text{LT}(F) = \{\text{LT}(f) : f \in F \setminus \{0\}\}$,
7. $\text{LC}(F) = \{\text{LC}(f) : f \in F \setminus \{0\}\}$.

One can now proceed with polynomial reduction.

Definition 2.9. *Let $f, g, h \in k[X], F \subseteq k[X]$. We say that g* reduces to *h* w.r.t. *f (and denote $g \to_f h$) iff there are monomials $s, t \in [X]$ such that s has a non-vanishing coefficient c in g, $s = \text{LT}(f) \cdot t$ and*

$$h = g - \frac{c}{\text{LC}(f)} \cdot t \cdot f.$$

We say that g reduces to *h* w.r.t. *F ($g \to_F h$) iff there is $f \in F$ such that $g \to_f h$.*

With this polynomial reduction, we can define a relation of reduction, that is compatible with the operations in the polynomial ring. Moreover, the reflexive-transitive-symmetric closure of the reduction relation $\to_F$ is equal to the congruence modulo the ideal generated by F(i.e., $\equiv_{\langle F \rangle} = \leftrightarrow_F^*$).
We are able to define the Gröbner basis now.

Definition 2.10. *A subset F of $k[X]$ is a* Gröbner basis *of an ideal I in $k[X]$ w.r.t. some admissible ordering $>$ iff*

$$\langle \text{LT}(F) \rangle = \langle \text{LT}(I) \rangle.$$

In order to find a Gröbner basis of an ideal given by a set of its generators, Buchberger introduced a criterion for deciding whether given set is a Gröbner basis. It is based on notion of so called *critical pairs* that represents the essential branching occurring during the polynomial reduction.

Definition 2.11. *Let* $f, g \in k[X]\,, t = \mathrm{lcm}(\mathrm{LM}(f)\,, \mathrm{LM}(g))$. *Then*

$$\mathrm{cp}(f,g) = (t - \frac{t}{\mathrm{LT}(f)} \cdot f, t - \frac{t}{\mathrm{LT}(g)} \cdot g)$$

is the critical pair *of* f *and* g. *The difference of the elements of* $\mathrm{cp}(f,g)$ *is the* S-polynomial *of* f *and* g, *denoted by* $\mathrm{spol}(f,g)$.

Theorem 2.12 (Buchberger's theorem)**.** *Let* F *be a subset of* $k[X]$. F *is a Gröbner basis if an only if* $\mathrm{spol}(f,g) \to_F^* 0$ *for all* $f, g \in F$.

Proof. The proof of the Buchberger's theorem is rather technical and can be found in Cox, Little and O'Shea [22] Chapter 2, §6. □

Previous theorem allows us to construct a Gröbner basis of an ideal from a given set of its generators. The *Buchberger's algorithm* is described in Algorithm 2.1.

Algorithm 2.1 Buchberger's Algorithm

Input: $F = \{f_1, \ldots, f_s\} \subseteq k[X]$, where $f_i \neq 0$ for $1 \leq i \leq s$ and some admissible ordering $>$.
Output: $G = \{g_1, \ldots, g_t\}$, a Gröbner basis of $\langle f_1, \ldots, f_s \rangle$ w.r.t. $>$.

```
G = F, 𝒢 = {{f_i, f_j} | f_i ≠ f_j ∈ G}
while 𝒢 ≠ ∅ do
    Choose some {f, g} ∈ 𝒢.
    𝒢 = 𝒢 \ {f, g}
    Find h such that spol(f, g) --G-->+ h.
    if h ≠ 0 then
        𝒢 = 𝒢 ∪ {{u, h} | u ∈ G}
        G = G ∪ {h}
return G
```

The biggest disadvantage of Buchberger's algorithm is that the algorithm performs many unnecessary reductions of S-polynomials. Further issues, such as the order in which S-polynomials are processed or the choice of monomial ordering, also have a strong influence on the efficiency of the algorithm. For example, the $>_{\mathrm{grevlex}}$ is often the most efficient admissible ordering.

The complexity of Buchberger's algorithm is closely related to the total degree of the intermediate polynomials generated by the algorithm. There are examples where the computation of a Gröbner basis of an ideal generated by polynomials of degree at most d involves polynomials of degree proportional to 2^{2^d} [22]. In fact, Buchberger's algorithm can have double exponential complexity. There are also examples for which the computation of a Gröbner basis using Buchberger's algorithm requires an enormous amount of time and memory. Still, these examples tend to be somewhat artificial and, in general, the running time and memory requirements seem to be much more manageable for generic cases, see Mayr [29].

Example 2.1.1. Lets observe the run of Buchberger's algorithm on an example. Let $f_1 = xy - y, f_2 = -x + y^2 \in \mathbb{Q}[x, y]$ and choose $>_{\text{lex}}$ with $y > x$ as the admissible ordering.

At first we have $G = \{f_1, f_2\}$ and $\mathcal{G} = \{\{f_1, f_2\}\}$. We enter the while loop and see that $\text{spol}(f_1, f_2) \xrightarrow{G} x^3 - x = h$. Because $h \neq 0$, we add $f_3 = x^3 - x$ to our generating set G. The new pairs $\{f_1, f_3\}, \{f_2, f_3\}$ must be added to the set $\mathcal{G}$.

In the next two iterations of the while loop we determine that both $\text{spol}(f_1, f_3)$ and $\text{spol}(f_2, f_3)$ reduce to zero and the Buchberger's algorithm terminates returning $G = \{xy - y, -x + y^2, x^3 - x\}$, a Gröbner basis of $\langle xy - y, -x + y^2\rangle$ w.r.t. $>_{\text{lex}}$.

Example 2.1.2. Now, lets consider $f_1 = x^2 + y^2 + 1, f_2 = x^2y + 2xy + x \in \mathbb{F}_5[x, y]$ and $>_{\text{lex}}$ with $x > y$. We see that $f_3 = \text{spol}(f_1, f_2) = 3xy + 4x + y^3 + y$ is already reduced w.r.t. G and add it to our basis. Our next choice is f_1, f_3, for which $\text{spol}(f_1, f_3) \xrightarrow{G} 4y^5 + 3y^4 + y^2 + y + 3$ and it is also added to the basis.

During the remaining four iterations of the while loop, all of the S-polynomials are reduced to zero. Thus, $G = \{f_1, f_2, f_3, f_4\}$ is a Gröbner bases for $I = \langle f_1, f_2\rangle$ w.r.t. the selected ordering.

One can see that the algorithm adds polynomials to the starting set, ending up with slightly more polynomials. Some of them are in fact redundant in the sense of the next definition.

Definition 2.13. *A* reduced Gröbner basis *for an ideal $I \subset k[X]$ is a Gröbner basis G for I such that for all distinct $p, q \in G$, no monomial appearing in p is a multiple of* $\text{LT}(q)$. *A* monic Gröbner basis *is a reduced Gröbner basis in which the leading coefficient of every polynomial is* 1, *or* 0 *if $I = \langle 0\rangle$.*

Important fact is that if we fix an admissible ordering $>$ on $k[X]$, each ideal I in $k[X]$ has a unique monic Gröbner basis with respect to $>$.

2.2 Equation Solving

One of the most desired properties of Gröbner bases is so called *elimination property*. We will show now, how to use the Gröbner bases to solve a system of polynomial equations. We need to somehow formalize the idea of elimination of variables.

Definition 2.14. *Given $I = \langle f_1, \ldots, f_s\rangle \subseteq k[x_1, \ldots, x_n]$, the* l-th elimination ideal *I_l is the ideal of $k[x_{l+1}, \ldots, x_n]$ defined by:*

$$I_l = I \cap k[x_{l+1}, \ldots, x_n].$$

Next theorem shows that a lexicographic Gröbner basis actually emulates Gaussian elimination for polynomial equations and the lexicographic basis has polynomials with some variables eliminated.

Theorem 2.15 (Elimination Theorem). *Let $I = \langle f_1, \ldots, f_s \rangle \subset k[x_1, \ldots, x_n]$ be an ideal and let G be a Gröbner basis of I with respect to the lexicographic ordering where $x_1 > x_2 > \cdots > x_n$. Then, for every $0 \leq l \leq n$, the set*

$$G_l = G \cap k[x_{l+1}, \ldots, x_n]$$

is a Gröbner basis of the l-th elimination ideal I_l.

Proof. Fix l between 0 and n. Since $G_l \subset I_l$ by construction, it suffices to show that

$$\langle \mathrm{LT}(I_l) \rangle = \langle \mathrm{LT}(G_l) \rangle$$

by the definition of Gröbner basis. One inclusion is obvious, and to prove the other inclusion $\langle \mathrm{LT}(I_l) \rangle \subset \langle \mathrm{LT}(G_l) \rangle$, we need only to show that the leading term $\mathrm{LT}(f)$, for an arbitrary $f \in I_l$, is divisible by $\mathrm{LT}(g)$ for some $g \in G_l$.

To prove this, note that f also lies in I, which tells us that $\mathrm{LT}(f)$ is divisible by $\mathrm{LT}(g)$ for some $g \in G$ since G is a Gröbner basis of I. Since $f \in I_l$, this means that $\mathrm{LT}(g)$ involves only variables $x_{l+1}, \ldots, x_n$. Now comes the crucial observation: since we are using the lexicographic ordering with $x_l > \cdots > x_n$, any monomial involving $x_1, \ldots, x_l$ is greater than all monomials in $k[x_{l+1}, \ldots, x_n]$, so that $\mathrm{LT}(g) \in k[x_{l+1}, \ldots, x_n]$ implies $g \in k[x_{l+1}, \ldots, x_n]$. This shows that $g \in G_l$. Thus, proving the theorem. □

Example 2.2.1. Consider the system of equations $f_1 = f_2 = f_3 = 0$, where

$$\begin{aligned} f_1 &= 4xz - 4xy^2 - 16x^2 - 1, \\ f_2 &= 2y^2z + 4x + 1, \\ f_3 &= 2x^2z + 2y^2 + x, \end{aligned}$$

are polynomials in $\mathbb{Q}[x, y, z]$. We are looking for solutions of this system of algebraic equations in $\overline{\mathbb{Q}}^3$, where $\overline{\mathbb{Q}}$ is the field of algebraic numbers.

Choose the lexicographic ordering with $x < y < z$. Then the reduced Gröbner basis of $\langle f_1, f_2, f_3 \rangle$ is $G = \{g_1, g_2, g_3\}$, where

$$\begin{aligned} g1 &= 65z + 64x^4 - 432x^3 + 168x^2 - 354x + 104, \\ g2 &= 26y^2 - 16x^4 + 108x^3 - 16x^2 + 17x, \\ g3 &= 32x^5 - 216x^4 + 64x^3 - 42x^2 + 32x + 5. \end{aligned}$$

We see that g_3 is a univariate polynomial in x. So we can get solutions by solving g_3 and propagating the partial solutions upwards to solutions of the full system. The univariate polynomial g3 is irreducible over Q, and the solutions are

$$(\alpha, \pm\frac{1}{\sqrt{26}}\sqrt{\alpha}\sqrt{16\alpha^3 - 108\alpha^2 + 16\alpha - 17}, -\frac{1}{65}(64\alpha^4 - 432\alpha^3 + 168\alpha^2 - 354\alpha + 104)),$$

where α is a root of g_3.

Chapter 3

Advanced Algorithms for Finding Gröbner Bases

We already mentioned that the complexity of Buchberger's algorithm is double exponential in the worst case. However, we might encounter many instances on which the performance of the algorithm is better. On the other hand, we can see even from the Example 2.1.2 in the previous chapter that the Buchberger's algorithm is not at all optimized. In fact, many S-polynomials tend to reduce to zero during the run of the algorithm and only one reduction is done at every step. Such observations motivated improvements to the Buchberger's algorithm proposed during last two decades.

Implementations of the Gröbner basis finding algorithms are available in most of today's computer algebra systems such as MATHEMATICA, MAPLE, MAGMA, SAGE, etc. Such implementations are done by teams of programmers and are highly optimised, in some cases even including optimizations of compiled code. Therefore, it is not very reasonable to try to come up with new implementation of such algorithms and rather use the available software. However, good understanding of the algorithms might be useful for determining how to prepare input to that algorithm. The details of algorithm can dictate strategies for pre-conditioning. In this chapter, we explain some of the advanced algorithms, that will be later used in cryptographic applications.

3.1 Linear Algebra and $F4$

Lots of new ideas are due to Faugère. One of the most obvious weaknesses of the Buchberger's algorithm is that there is done only one reduction of S-polynomial at every stage. This is the motivation for so called $F4$ algorithm introduced in Faugère [24]. The algorithm employs methods of linear algebra to increase the number of reductions done at every step.

Suppose we need to reduce polynomials $f_1, \ldots, f_n \in k[x_1, \ldots, x_n]$. The idea is to represent those polynomials as a matrix A over k and then use some efficient method from linear algebra to transform it into row echelon form. This transfor-

mation of A will reduce all of $f_1, \ldots, f_n$ simultaneously. If we include this new reduction into the Buchberger's algorithm, we get more efficient algorithm.

Following definition shows how to represent a set of polynomials as a matrix.

Definition 3.1. *Let $F = (f_1, \ldots, f_m)$ be a tuple of polynomials in $k[x_1, \ldots, x_n]$ and let $T(F)$ and $T_>(F)$ denote the set of pairwise distinct monomials in F and its tuple ordered w.r.t. some admissible ordering $>$, respectively. The number of distinct monomials in F, $|T(F)|$, is denoted by s.*

Let a general polynomial $f \in k[x_1, \ldots, x_n] = R$ be written as

$$f = \sum_{i=1}^{s} c_i x^{\alpha_i},$$

with $\alpha_i \in \mathbb{Z}_{\geq 0}^n$ and $c_i \in k$.

Define the vector representation *map*

$$\psi_{T_>(F)} : k[x_1, \ldots, x_n] \to k^s$$

of f w.r.t. $T_>(F)$ as follows

$$\psi_{T_>(F)}(f) = (c_1, \ldots, c_s)$$

and the matrix representation *of a tuple of polynomials F*

$$\Psi_{T_>(F)} : R^m \to Mat_{m,s}(k), (f_1, \ldots, f_m) \mapsto \begin{pmatrix} \psi_{T_>(F)}(f_1) \\ \vdots \\ \psi_{T_>(F)}(f_m) \end{pmatrix}$$

If it is clear from the context with respect to which ordering and support the polynomials are represented, the subscripts are omitted.

The following definition of the matrix $\tilde{F}^+$ is the $F4$-equivalent of reduced S-polynomials in the Buchberger's algorithm.

Definition 3.2. *Let F be a subset of $k[x_1, \ldots, x_n]$. We denote $\tilde{F}$, the set of polynomials corresponding to the row echelon form of $\Psi(F)$. Let $\tilde{F}^+$ denote $\left\{ f \in \tilde{F} : \mathrm{LT}(f) \notin \mathrm{LT}(F) \right\}$.*

We use $\tilde{F}^+$ to form a basis of $\langle F \rangle$. In order to get the basis, the elements of $\tilde{F}^+$ are joined with a subset, H, of the original F, such that

$$\mathrm{LT}(H) = \mathrm{LT}(F) \text{ and } |H| = |\mathrm{LT}(F)|$$

hold. As a consequence of the following theorem, the ideal $\langle F \rangle$ is spanned by $H \cup \tilde{F}^+$.

Theorem 3.3. *Let k be a field, F a finite subset of $k[x_1, \ldots, x_n]$ and let s denote the cardinality of the support $T_>(F)$. For any subset $H \subseteq F$ such that $|H| = |\mathrm{LT}(F)|$ and $\mathrm{LT}(H) = \mathrm{LT}(F)$, the vectors*

$$\psi(g) \in k^s, \text{ for } g \in \tilde{F}^+ \cup H,$$

form a triangular basis of the subspace of vector space k^s spanned by the vectors $\{\psi(f) : f \in F\}$.

Proof. Write $G = \tilde{F}^+ \cup H$. All elements g of G have distinct leading terms and are linear combinations of elements of F. Hence, the set $\{\psi(g) : g \in G\}$ is linearly independent and is included in the subspace spanned by the vectors corresponding to elements of F. Furthermore, let r denote the rank of the subspace spanned by $\psi(f)$ for $f \in F$. Also,

$$\mathrm{LT}(G) = \mathrm{LT}\left(\tilde{F}^+\right) \cup \mathrm{LT}(H) = \mathrm{LT}\left(\tilde{F}\right)$$

holds, which implies $|\mathrm{LT}(G)| = |\mathrm{LT}\left(\tilde{F}\right)| = r$ and the theorem follows. □

The main idea of $F4$ is rooted in the reduction of S-polynomials. Instead of computing the reduction of every S-polynomial individually, it creates a selection of critical pairs $b = (b_1, b_2)$ for b_1, b_2 in the intermediate basis G' and passes the two polynomials

$$\frac{\mathrm{lcm}(\mathrm{LT}(b_1), \mathrm{LT}(b_2))}{\mathrm{LT}(b_1)} b_1, \frac{\mathrm{lcm}(\mathrm{LT}(b_1), \mathrm{LT}(b_2))}{\mathrm{LT}(b_2)} b_2$$

to the reduction function. We can have different strategies for selecting the critical pairs, e.g. we can take all critical pairs available at the time of computation. Let us assume the *normal selection strategy* is adopted, i.e. we select critical pairs B_d such that the total degree of least common multiples of $\mathrm{LT}(b_1)$ and $\mathrm{LT}(b_2)$ for $(b_1, b_2) \in B$ is minimal. The critical pairs corresponding to degree d are

$$B_d = \{(b_1, b_2) : b_1, b_2 \in G' \text{ where } \deg(\mathrm{lcm}(\mathrm{LT}(b_1), \mathrm{LT}(b_2))) = d, b_1 \neq b_2\}.$$

Hence, the following set is passed to the simultaneous reduction routine $F4$,

$$L_d = \bigcup_{(b_1,b_2) \in B_d} \left\{ \frac{\mathrm{lcm}(\mathrm{LT}(b_1), \mathrm{LT}(b_2))}{\mathrm{LT}(b_1)} b_1, \frac{\mathrm{lcm}(\mathrm{LT}(b_1), \mathrm{LT}(b_2))}{\mathrm{LT}(b_2)} b_2 \right\}.$$

The reduction in $F4$ uses preprocessed reductors of an intermediate basis G'. The addition of reductors is done by a routine called *SymbolicPreprocessing*.

Definition 3.4. *During the execution of an algorithm to compute Gröbner basis, a* reductor *r of the set F is a polynomial satisfying*

$$\mathrm{LT}(r) \in T(F) \setminus \mathrm{LT}(F).$$

Algorithm 3.1 SymbolicPreprocessing

Input: A set $F \subset k[x_1, \ldots, x_n]$ and an intermediate basis G'.
Output: The set $F \cup R$ for a set of reductors R w.r.t. G'.

```
D = LT(F), R = ∅
while T(F ∪ R) ≠ D do
    Select m ∈ T(F ∪ R) \ D
    D = D ∪ {m}
    if m is divisible by an element g ∈ LT(G') then
        m' = m/LT(g)
        R = R ∪ {gm'}
return F ∪ R
```

Algorithm 3.2 Reduction$F4$

Input: A finite set $L \subset k[x_1, \ldots, x_n]$ and an intermediate basis G'.
Output: The set $\tilde{F}^+$ reduced w.r.t. G'.

```
F = SymbolicPreprocessing(L, G')
F̃ = polynomials corresponding to the row echelon form of Ψ(F)
F̃+ = { g ∈ F̃ : LT(g) ∉ LT(F) }
return F̃+
```

The algorithm SymbolicPreprocessing 3.1 appends reductors to the set F with respect to an intermediate basis G'.

Now, we are ready to formulate the function Reduction$F4$ that simultaneously reduces polynomials corresponding to several critical pairs. It is given in Algorithm 3.2.

S-polynomials that do not reduce to zero in Buchberger's algorithm extend the ideal spanned by the leading terms of the intermediate basis. This way, an ascending chain of leading term ideals is obtained. Similarly, the leading terms of the elements of $\tilde{F}^+$ contribute to the ideal spanned by the leading terms of the intermediate basis. This is formalized in the following lemma.

Lemma 3.5. *Let $\tilde{F}^+$ denote the output of* Reduction$F4$ *on L_d w.r.t. G'. For all $f \in \tilde{F}^+$, $\mathrm{LT}(f)$ is not an element of $\langle \mathrm{LT}(G') \rangle$.*

Proof. Let f be in $\tilde{F}^+$ and F the output of SymbolicPreprocessing of L_d with respect to G'. Suppose, to achieve a contradiction, that $\mathrm{LT}(f) \in \langle \mathrm{LT}(G') \rangle$. This assumption and $\mathrm{LT}(f) \in T(\tilde{F}^+) \subset T(F)$ implies that Symbolic Preprocessing must have added a reductor $\frac{\mathrm{LT}(f)}{\mathrm{LT}(g)} g$ to F for a suitable $g \in G'$. This would mean $\mathrm{LT}(f) \in \mathrm{LT}(F)$, a contradiction to the definition of $\tilde{F}^+$. Hence, $\mathrm{LT}(f)$ is not an element of $\langle \mathrm{LT}(G') \rangle$. □

The next lemma assures that the elements that are added to the intermediate basis are members of the ideal $\langle G' \rangle$.

Lemma 3.6. *Let $\tilde{F}^+$ be as in Lemma 3.5. Then $\tilde{F}^+ \subset \langle G' \rangle$.*

Proof. Every $f \in \tilde{F}^+$ is a linear combination of elements of L_d and reductors R, which are both subsets of $\langle G' \rangle$. □

The following lemma states that all S-polynomials in the set of possible k-linear combinations of L_d reduce to zero by a subset of $\tilde{F}^+ \cup G'$. This is used to prove the correctness of the algorithm by criterion stated in Buchberger's theorem.

Lemma 3.7. *Let $\tilde{F}^+$ be as in Lemma 3.5. For all k-linear combinations, f, of elements of L_d, the normal form equals to zero w.r.t. $\tilde{F}^+ \cup G'$.*

Proof. Let f be a linear combination of elements of L_d. Suppose F is the output of the SymbolicPreprocessing of L_d with respect to G'. By construction, L_d is a subset of F and, therefore due to Theorem 3.3, these elements are a linear combination of the triangular basis $\tilde{F}^+ \cup H$ for a suitable subset $H \subset F$. Elements of H are either elements of L_d or of the form $x^\alpha g$, for $g \in G'$ and $\alpha \in \mathbb{Z}^n_{\geq 0}$, and f can thus be written as

$$f = \sum_i a_i f_i + \sum_j a_j x^{\alpha_j} g_j,$$

for $f_i \in \tilde{F}^+$ and $g_j \in G'; a_i, a_j \in k$ and $\mathbb{Z}^n_{\geq 0}$. Thus the Division Algorithm gives a remainder equal to 0 for a suitable tuple of elements in $\tilde{F}^+ \cup G'$, hence there exists a reduction chain to zero. □

Now we are ready to describe the Algorithm $F4$ and prove its correctness. The selection strategy done in subroutine Select() is kept unspecified. One can choose from different strategies, e.g. to select all critical pairs available at that time, or use the normal selection strategy.

Algorithm 3.3 Algorithm $F4$

Input: A finite set $F \subset k[x_1, \ldots, x_n]$.
Output: A Gröbner basis G for $\langle F \rangle$ w.r.t. some admissible ordering $>$.

$G' = F, \tilde{F_0}^+ = F, d = 0$
$B = \{(b_1, b_2) : b_1, b_2 \in G' \text{ and } b_1 \neq b_2\}$
while $B \neq \emptyset$ **do**
  $d = d + 1$
  $B_d = \text{Select}(B)$
  $B = B \setminus B_d$
  $L_d = \bigcup_{(b_1,b_2) \in B_d} \left\{ \frac{\text{lcm}(\text{LT}(b_1), \text{LT}(b_2))}{\text{LT}(b_1)} b_1, \frac{\text{lcm}(\text{LT}(b_1), \text{LT}(b_2))}{\text{LT}(b_2)} b_2 \right\}$
  $\tilde{F_d}^+ = \text{Reduction}F4(L_d, G')$
  for $f \in \tilde{F_d}^+$ **do**
    $B = B \cup \{(f, g) : g \in G'\}$
    $G' = G' \cup \{f\}$
return G'

Theorem 3.8. *The algorithm $F4$ computes a Gröbner basis G of an ideal spanned by F, such that $F \subseteq G$, in a finite number of steps.*

Proof. Correctness and termination are proven by following observations

1. Lemma 3.6 implies that during stage $d = d'$ of the algorithm, the intermediate basis satisfies
$$G' = \bigcup_{d=1}^{d'} \tilde{F_d}^+ \subset \langle F \rangle;$$

2. Lemma 3.5 shows that
$$\langle \mathrm{LT}\left(\tilde{F_1}^+\right)\rangle \subset \langle \mathrm{LT}\left(\tilde{F_1}^+ \cup \tilde{F_2}^+\right)\rangle \subset \cdots,$$
is an ascending chain of monomial ideals. The ascending chain condition from the definition of Noetherian ring, states that it can not be infinite. This implies that the while-loop should terminate as we run out of critical pairs eventually;

3. Suppose the algorithm terminates at $d = d_{F4}$. Since every pair (g_1, g_2) for $g_1, g_2 \in G = \bigcup_{d=1}^{d_{F4}} \tilde{F_d}^+$ is considered, $\mathrm{spol}(g_1, g_2)$ is in the linear span of elements of G. Lemma 3.7 states that its normal form equals zero. Hence, G is a Gröbner basis by Buchberger's theorem.

□

Example 3.1.1. For a better understanding of the $F4$ algorithm, an example run follows. However, one should be aware that it is impossible to fully appreciate the efficiency of the algorithm for a small example.

We will consider the Cyclic-4 problem (for more detail see Björck and Fröberg [10]). The ideal I is generated by the following polynomials:

$$\begin{aligned} f_1 &= a + b + c + d, \\ f_2 &= ab + bc + cd + ad, \\ f_3 &= abc + abd + acd + bcd, \\ f_4 &= abcd - 1. \end{aligned}$$

We will adapt the normal strategy and use the degree reverse lexicographical ordering. At the beginning, $G' = \{f_1, f_2, f_3, f_4\}$. Because of the normal selection strategy, we choose $B_1 = \{(f_1, f_2)\}$ so that $L_1 = \{bf_1, f_2\} = \{ab+b^2+bc+bd, ab+bc+cd+ad\}$.

We enter the SymbolicPreprocessing(L_1, G') and want to add the reductors to L_1. The only monomial appearing in $T(L_1) \setminus \mathrm{LT}(L_1)$ divisible by some $g \in \mathrm{LT}(G)$ is ad. It is divisible by $\mathrm{LT}(f_1) = a$, so we add $df_1 = ad + bd + cd + d^2$ as a reductor. Therefore, the reduction will be done on $F_1 = \{f_2, df_1, bf_1\} = \{ab + bc + cd + ad, ad + bd + cd + d^2, ab + b^2 + bc + bd\}$, or in matrix form:

$$F_1 = \begin{array}{c} \begin{array}{ccccccc} ab & b^2 & bc & ad & bd & cd & d^2 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array},$$

where the columns of the matrix are indexed by the terms appearing in F_1, ordered by the chosen ordering.

The row echelon form of F_1 is:

$$F_1 = \begin{pmatrix} 1 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$

that is to say $\tilde{F}_1 = \{f_5 = b^2+2bd+d^2, df_1 = ad+bd+cd+d^2, f_6 = ab+bc-bd-d^2\}$ and since $ab, ad \in \mathrm{LT}(F_1)$ we have $\tilde{F}_1^+ = \{f_5\}$. We also add f_5 to the intermediate basis G'.

In the next iteration of the while loop follows: we choose $B_2 = \{(f_1, f_3), (f_1, f_5), (f_2, f_3), (f_2, f_5)\}$ so that $L_2 = \{bcf_1, f_3, b^2f_1, f_5, cf_2, f_3, bf_2, af_5\}$.

As we enter the SymbolicPreprocessing(L_2, G'), the reductors that we add are:

$$\begin{aligned} bf_5 &= b^3 + 2b^2d + bd^2, \\ df_5 &= b^2d + 2bd^2 + d^3, \\ cf_5 &= b^2c + 2bcd + cd^2, \\ cdf_1 &= acd + bcd + c^2d + cd^2, \\ d^2f_1 &= ad^2 + bd^2 + cd^2 + d^3, \\ bdf_1 &= abd + b^2d + bcd + bd^2. \end{aligned}$$

The matrix representation of F_2 is:

$$A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

where the columns are indexed by $ab^2, b^3, abc, b^2c, bc^2, abd, b^2d, acd, bcd, c^2d, ad^2$,

bd^2, cd^2, d^3. Again, we transform this matrix into the row echelon form and get:

$$\tilde{A}_2 = \left(\begin{array}{cccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 1 & -1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & -2 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & -1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right).$$

The only polynomial f in $\tilde{F}_2$ such that $\mathrm{LT}(f) \notin \mathrm{LT}(F_1)$ is $f_6 = bc^2 + c^2d - bd^2 - d^3$ and we add it to G'.

After another five iterations of the while loop and interreduction of G' we end up with the following reduced Gröbner basis of I w.r.t. degree reverse lexicographic ordering:

$$\begin{aligned} G \;=\; & \{c^2d^4 + bc - bd + cd - 2d^2, c^3d^2 + c^2d^3 - c - d, \\ & bd^4 + d^5 - b - d, bcd^2 + c^2d^2 - bd^3 + cd^3 - d^4 - 1, \\ & bc^2 + c^2d - bd^2 - d^3, b^2 + 2bd + d^2, a + b + c + d\}. \end{aligned}$$

Note that both f_5 and f_6 are in the basis.

The efficiency of $F4$ depends on the techniques from linear algebra for finding the row echelon form of a matrix. Such techniques were studied for a long time and many highly optimised algorithms are available. There are, however, no good results on complexity of $F4$ in general. Unfortunately, there is also one trade-off we have to make. The polynomial reduction is done using matrices and polynomials with many monomials introduce large matrices. This sometimes leads to huge memory requirements. The $F4$ is still a very effective algorithm that makes it possible to compute previously intractable Gröbner bases. Actually, the Cyclic-9 challenge was first cracked using $F4$.

To avoid unnecessary reductions to zero, new criterion was introduced in Faugère [25]. The criterion is also used to create a new algorithm called $F5$. The ideas behind the new criterion are closely related to regular sequences and syzygies of polynomials. However, we will not give a description of the $F5$ criterion and the algorithm because the necessary theory is too extensive. One can see Stegers [32] for thorough description of $F5$. Important fact is that if we omit the unnecessary reductions to zero, the efficiency of the algorithm increases significantly. In fact, The $F5$ algorithm was later successfully used to attack the Hidden Field Equation cryptosystem by Faugère [27]. There are also some attempts to combine the $F4$-style reduction with the $F5$ criterion; in Albrecht and Perry [5], such an algorithm called $F4/5$ is given.

3.2 Zero-Dimensional Ideals and FGLM

Thanks to Faugère, Gianni, Lazard, and Mora [26], there is an algorithm for transforming given Gröbner basis with respect to some ordering to a Gröbner basis w.r.t. a different ordering. The name of the algorithm, *FGLM*, is abbreviation of names of authors of the original paper. The motivation for this algorithm is obvious; finding a lexicographic Gröbner basis is computationally very demanding task, but for another ordering, such as $>_{\text{grevlex}}$, it might be much more easier.

We will compute in residue class ring $k[X]/I$ to reduce algebraic computations to linear algebra problems. By the residue class ring we consider the set of all equivalence classes in $k[X]$ such that f and g are in the same class iff the difference $f - g$ is in the ideal I. We can consider the residue class ring as a vector space over the ground field k. More over, if I is a zero-dimensional ideal (i.e. variety corresponding to I is finite), the residue class ring is a finite-dimensional vector space.

We should also remark that Gröbner basis G of I gives us a natural basis of $k[X]/I$.

Theorem 3.9. *The irreducible monomials modulo a Gröbner basis G w.r.t. some admissible ordering $>$, viewed as polynomials with coefficient 1, form a basis for vector space $k[X]/I$ over k.*

Proof. Let B be the set of all irreducible monomials modulo G, viewed as polynomials with coefficient 1. Clearly B generates $k[X]/I$, since every polynomial can be reduced to a linear combination of elements of B with coefficients in k. Moreover, B is linearly independent, because any nontrivial linear combination of elements of B is irreducible modulo G and therefore different from 0 in $k[X]/I$. □

We should remark that the previous theorem says nothing about the dimension of the ideal. However, if the ideal is zero-dimensional(i.e. the variety $\mathrm{V}(I)$ of the ideal I is finite), then the set of irreducible polynomials is finite. Dimension of the vector space is an invariant for a given ideal.

Next theorem gives a characterization for zero-dimensional ideals.

Theorem 3.10 (Finiteness Theorem)**.** *Let k be a field and $I \subset k[x_1, \ldots, x_n]$ be an ideal. The following conditions are equivalent:*

(i) The algebra $A = k[x_1, \ldots, x_n]/I$ is finite-dimensional over k.

(ii) The variety $\mathrm{V}(I) \subset \mathbb{C}^n$ is a finite set.

(iii) If G is a Gröbner basis for I, then for each $i, 1 \leq i \leq n$, there is an $m_i \geq 0$ such that $x_i^{m_i} = \mathrm{LT}(g_i)$ for some $g_i \in G$.

Proof. For proof see Theorem 6 of chapter 5, §3 in Cox, Little and O'Shea [22]. □

It is an easy corollary that the dimension of $k[X]/I$ as a vector space over k can be computed using the numbers $m_1, \ldots, m_n$ from Theorem 3.10:

$$\dim(k[X]/I) = m_1 \cdot \cdots \cdot m_n.$$

Algorithm 3.4 FGLM

Input: A Gröbner basis G of a zero-dimensional ideal $I \subset k[x_1, \ldots, x_n]$ with respect to some admissible term ordering $>$.

Output: A lexicographic Gröbner basis G' of I.

Let $G' = L = U = U' = \emptyset$ and let $u = 1$.
while u $\neq$ *nil* **do**
  Determine $u' = \mathrm{NF}_>(u, G)$ and test if it is a linear combination of the elements in U'.
  if u' is not a linear combination of the elements in U' **then**
    $U = U \cup \{u\}$
    $U' = U' \cup \{u'\}$
  else
    Let $u' = \sum_{j=1}^{m} a_j u'_j$ for some $u'_j \in U'$.
    $G' = G' \cup \{u - \sum_{j=1}^{m} a_j u_j\}$
    $L = L \cup \{u\}$
  $u = \mathrm{NEXT}(u, L)$
return G'

The importance of next lemma is that it allows us to test ideal membership incrementally using linear dependence tests.

Lemma 3.11. *Let $I \subset k[X]$ be an ideal and G a Gröbner basis of I w.r.t. some admissible ordering $>$. Let $u_j \in k[X]$ and $a_j \in k, 1 \leq j \leq m$. Then*

$$\sum_{j=1}^{m} a_j u_j \in I \Leftrightarrow \sum_{j=1}^{m} a_j \mathrm{NF}_>(u_j, G) = 0.$$

Proof. Since for a Gröbner basis G w.r.t. $>$, polynomials $u, v \in k[X]$ and $a, b \in k$ we have $\mathrm{NF}_>(au + bv, G) = a\ \mathrm{NF}_>(u, G) + b\ \mathrm{NF}_>(v, G)$, the claim follows from basic property of Gröbner bases: for all $p \in I$ iff $\mathrm{NF}_>(p, G) = 0$. □

FGLM Gröbner basis conversion is described in Algorithm 3.4. We also provide an example description of the procedure NEXT to generate appropriate next monomial w.r.t. the lexicographic ordering in Algorithm 3.5.

Algorithm 3.5 NEXT

Input: A power product u, a set of power products L and a list of indeterminates $\{x_1, \ldots, x_n\}$.

Output: The power product that is next in lexicographic ordering with $x_1 < x_2 < \cdots < x_n$ and that is not a multiple of any power product in L.

$u_1 = x_1 u$.
if u_1 is not a multiple of any power product in L **then**
  return $v = u_1$
else
  Let $u_1 = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$.
  return $\mathrm{NEXT}(x_2^{\alpha_2} \cdots x_n^{\alpha_n}, L, \{x_2, \ldots, x_n\})$

Theorem 3.12. *The FGLM algorithm is correct and terminates.*

Proof. We will show termination first. By induction on the while-loop, U is linearly independent modulo I. Since dimension of $k[x_1, \ldots, x_n]/I$ is finite by theorem 3.10, the then-branch can not be executed infinitely often. Also the else-branch can not be executed infinitely often, because if $p_1, \ldots, p_k \in k[x_1, \ldots, x_n]$ are the power products in L, and u is added to L then $u \notin \langle p_1, \ldots, p_k \rangle$ by construction of the algorithm next. This cannot happen infinitely often because of the ascending chain condition for noetherian rings.

As for the correctness, we want to show that G' is a Gröbner basis w.r.t. lexicographic ordering. By lemma 3.11 we add to G' only such polynomials that are in I, so $G' \subseteq I$. Now let p be an arbitrary but fixed polynomial in I. We want to show that its normal form w.r.t. G' and $<_{\text{lex}}$ is equal to zero. Aiming for a contradiction, lets assume that r is nonzero. We know that r lies in I and also that it is a linear combination of elements of U. Using again the lemma 3.11, we get:

$$r = \sum_{j=1}^{m} a_j u_j \Leftrightarrow 0 = \mathrm{NF}_{>}(r, G) = \sum_{j=1}^{m} a_j u'_j.$$

Since U was constructed such that there is no linear dependence among elements of U', only the trivial linear combination of u'_j is equal to zero. Therefore, r must be equal to zero giving the contradiction with our assumption. Hence p is an element of $\langle G' \rangle$ and G' is a lexicographic Gröbner basis. □

Example 3.2.1. We will illustrate basis conversion with a simple example. Consider the ideal in $k[x, y]$ generated by

$$G = \{x^3 + 2xy - x + 1, y^2 + x - 3\}.$$

G is already a Gröbner basis with respect to the total degree ordering, and the ideal $I\langle G \rangle$ can be shown to be zero-dimensional. We generate the monomials $1, x, x^2, \ldots$ with the following normal forms.

$$\begin{aligned}
\mathrm{NF}(1, G) &= 1 \\
\mathrm{NF}(x, G) &= x \\
\mathrm{NF}\left(x^2, G\right) &= x^2 \\
\mathrm{NF}\left(x^3, G\right) &= -2xy + x - 1 \\
\mathrm{NF}\left(x^4, G\right) &= -2x^2y + x^2 - x \\
\mathrm{NF}\left(x^5, G\right) &= -4xy - 5x^2 + 2y + 13x - 1 \\
\mathrm{NF}\left(x^6, G\right) &= -4x^2y + 12xy + 13x^2 - 6x + 5
\end{aligned}$$

At this point, we discover a linear dependence among the normal forms:

$$\begin{aligned}
&\mathrm{NF}(1, G) - 2\mathrm{NF}(x, g) - 11\mathrm{NF}\left(x^2, G\right) \\
&+6\mathrm{NF}\left(x^3, G\right) - 2\mathrm{NF}\left(x^4\right) + \mathrm{NF}\left(x^6, G\right) \;=\; 0.
\end{aligned}$$

Hence, we have found a polynomial in G'; namely:

$$1 - 2x - 11x^2 + 6x^3 - 2x^4 + x^6.$$

The set U is, at this point, $\{1, x, x^2, x^3, x^4, x^5\}$. We add the leading power product x^6 of p to L. Since x^7 is a multiple of x^6, the next power product generated is y. The monomial y cannot be simplified and is already in normal form. $\mathrm{NF}(y, G) = y$ is not linearly independent because

$$\begin{aligned} &-\mathrm{NF}(1, G) - 11\mathrm{NF}(x, G) + 5\mathrm{NF}\left(x^2, G\right) \\ &-2\mathrm{NF}\left(x^3, G\right) + \mathrm{NF}\left(x^5, G\right) - 2\mathrm{NF}(y, G) \quad = \quad 0. \end{aligned}$$

Therefore, the polynomial

$$-1 - 11x + 5x^2 - 2x^3 + x^5 - 2y$$

is also in G'. So, we add the polynomial to G', and add the leading term, y, to L.

At this point, no lexicographic successor to y can be found that is not a multiple of y or of x^6. In consequence, the algorithm terminates, having found the lexicographic basis

$$G' = \{1 - 2x - 11x^2 + 6x^3 - 2x^4 + x^6, -1 - 11x + 5x^2 - 2x^3 + x^5 - 2y\}.$$

One may be concerned about the cost of the basis conversion. Following theorem gives us a bound on complexity of the FGLM algorithm.

Theorem 3.13. *Let $R = k[x_1, \ldots, x_n]$. Furthermore $G_1 \subset R$ is the Gröbner basis relative to a term order $>_1$ of an ideal I, and $D = \dim(R/I)$. We can then convert G_1 into a Gröbner basis G_2 relative to a term order $>_2$ in $\mathcal{O}(nD^3)$ field operations.*

Proof. The complexity depends on optimizations involved in finding normal forms and linear algebraic methods; the proof can be found in original paper by Faugère et al. [26]. □

3.3 General Basis Conversion with Gröbner Walk

In this section, we will describe a general Gröbner basis conversion algorithm. To do so, we will need a notion of *Gröbner fan*, a structure containing all reduced Gröbner bases of a given ideal. The ideas connected to this topic are merely geometric. The algorithm itself was introduced by Collart, Kalkbrener and Mall [18]. We will build the theory and describe the algorithm as in Cox, Little and O'Shea [21].

Definition 3.14. *Let $I \subset k[X]$ be an ideal. We will denote the collection of all monomial ideals $\langle \mathrm{LT}_>(I)\rangle$, where $>$ ranges over all possible admissible orderings,*

$$\mathrm{Mon}(I) = \{\langle \mathrm{LT}_>(I)\rangle : > \text{ an admissible ordering}\}.$$

Following theorem is surprising and rather not obvious.

Theorem 3.15. *For an ideal $I \subset k[X]$, the set $\mathrm{Mon}(I)$ is finite.*

Proof. For a contradiction, let us suppose that $\mathrm{Mon}(I)$ is an infinite set. For each monomial ideal N in $\mathrm{Mon}(I)$, we will denote $>_N$ the particular admissible ordering such that $N = \{\mathrm{LT}(I)\}$. Let Σ be the collection of admissible orders $\{>_N : N \in \mathrm{Mon}(I)\}$. Our assumption implies that Σ is also infinite.

By the Hilbert Basis Theorem 2.3 we have $I = \langle f_1, \ldots, f_s\rangle$ for polynomials $f_i \in k[X]$. Since each f_i contains only a finite number of terms, by a pigeonhole principle argument, there exists an infinite subset $\Sigma_1 \subset \Sigma$ such that the leading terms $\mathrm{LT}_>(f_i)$ agree for $>$ in Σ_1 and all $i, 1 \leq i \leq s$. We write N_1 for the monomial ideal $\langle \mathrm{LT}_>(f_1), \ldots \mathrm{LT}_>(f_s)\rangle$, where $>$ in Σ_1.

If $F = \{f_1, \ldots, f_s\}$ were a Gröbner basis for I with respect to some $>_1$ in Σ_1, then we claim that F would be a Gröbner basis for I with respect to every $>$ in Σ_1. To see this, let $>$ be any element of Σ_1 other than $>_1$, and let $f \in I$ be arbitrary. Dividing f by F using $>$, we obtain

$$f = a_1 f_1 + \cdots + a_s f_s + r, \tag{3.1}$$

where no term in r is divisible by any of the $\mathrm{LT}_>(f_i)$. However, both $>$ and $>_1$ are in Σ_1, so $\mathrm{LT}_>(f_i) = \mathrm{LT}_{>_1}(f_i)$ for all i. Since $r = f - a_1 f_1 - \cdots - a_s f_s \in I$, and F is assumed to be a Gröbner basis for I w.r.t. $>_1$, this implies that $r = 0$. Since (3.1) was obtained using the division algorithm, $\mathrm{LT}_>(f) = \mathrm{LT}_>(a_i f_i)$ for some i, so $\mathrm{LT}_>(f)$ is divisible by $\mathrm{LT}_>(f_i)$. This shows that F is also a Gröbner basis for I with respect to $>$.

However, this cannot be the case since the original set of admissible orderings $\Sigma \supset \Sigma_1$ was chosen so that the monomial ideals $\langle \mathrm{LT}_>(I)\rangle$ for $>$ in Σ were all distinct. Hence, given any $>_1$ in Σ_1, there must be some $f_{s+1} \in I$ such that $\mathrm{LT}_{>_1}(f_{s+1}) \notin \langle \mathrm{LT}_{>_1}(f_1), \ldots, \mathrm{LT}_{>1}(f_s)\rangle = N_1$. Replacing f_{s+1} by its remainder on division by $f_1, \ldots, f_s$, we may assume in fact that no term in f_{s+1} is divisible by any of the monomial generators for N_1.

Now we apply the pigeonhole principle again to find an infinite subset $\Sigma_2 \subset \Sigma_1$ such that the leading terms of $f_1, \ldots, f_{s+1}$ are the same for all $>$ in Σ_2. Let

$N_2 = \langle \mathrm{LT}_>(f_1), \ldots, \mathrm{LT}_>(f_{s+1}) \rangle$ for all $>$ in Σ_2, and note that $N_1 \subset N_2$. The argument given in the preceding paragraph shows that $\{f_1, \ldots, f_{s+1}\}$ cannot be a Gröbner basis with respect to any of the admissible orderings in Σ_2, so fixing $>_2 \in \Sigma_2$, we find an $f_{s+2} \in I$ such that no term in f_{s+2} is divisible by any of the monomial generators for $N_2 = \langle \mathrm{LT}_{>_2}(f_1), \ldots, \mathrm{LT}_{>_2}(f_{s+1}) \rangle$.

Continuing in the same way, we produce a descending chain of infinite subsets $\Sigma \supset \Sigma_1 \supset \Sigma_2 \supset \Sigma_3 \supset \cdots$, and an infinite strictly ascending chain of monomial ideals $N_1 \subset N_2 \subset N_3 \subset \cdots$. This contradicts the ascending chain condition in $k[x_1, \ldots, x_n]$, so the proof is complete. □

We would like to use a somehow different notation in this section.

Definition 3.16. *A* marked Gröbner basis *for I is set GM of ordered pairs (g, m) such that $G = \{g : (g, m) \in GM\}$ is a monic Gröbner basis with respect to some admissible ordering $>$ and $m = \mathrm{LT}_>(g)$ for each $(g, m) \in GM$.*

The idea behind the definition is that we do not want to include a specific ordering into the definition of GM. From theorem 3.15 we can see that each ideal in $k[X]$ has only finitely many marked Gröbner bases.

We will also need a general method for specifying an admissible orderings on $k[X]$.

Definition 3.17. *Let M be an $m \times n$ real matrix M with rows $\mathbf{w}_1, \ldots, \mathbf{w}_m$. Then we define an order relation $>_M$ on $k[x_1, \ldots, x_n]$ in following way. For $x^\alpha, x^\beta \in [X]$ we say that $x^\alpha >_M x^\beta$ if there is an $l \leq m$ such that $\alpha \cdot \mathbf{w}_i = \beta \cdot \mathbf{w}_i$, for $i = 1, \ldots, l-1$, but $\alpha \cdot \mathbf{w}_l > \beta \cdot \mathbf{w}_l$.*

One can see that we compare terms x^α and x^β by first comparing their $\mathbf{w}_1$-weights $\alpha \cdot \mathbf{w}_1$ and $\beta \cdot \mathbf{w}_1$. If $\alpha \cdot \mathbf{w}_1 > \beta \cdot \mathbf{w}_1$ or $\beta \cdot \mathbf{w}_1 > \alpha \cdot \mathbf{w}_1$, then we order the terms accordingly. If $\alpha \cdot \mathbf{w}_1 = \beta \cdot \mathbf{w}_1$, then we continue to the later rows, breaking ties successively with the $\mathbf{w}_2$-weights, the $\mathbf{w}_3$-weights, and so on through to the $\mathbf{w}_m$-weights. To obtain a total order by this construction, it must be true that $\ker(M) \cap \mathbb{Z}^n = \{0\}$.

It is known that this construction gives every admissible ordering on $k[X]$. Therefore, all of the admissible orderings from our examples can be seen as $>_M$ orders for appropriate matrices.

1. The lexicographic ordering with $x > y > z$ is defined by the identity matrix
$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

2. The graded lexicographic ordering compares first by total degree and then breaks ties by the lexicographic ordering. Therefore, the appropriate matrix is
$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

3. The graded reverse lexicographic ordering can be defined for example by the matrix

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

For a matrix ordering $>_M$, the first row $\mathbf{w}$ of M plays a special role. We will assume that $\mathbf{w} \neq 0$.

Lemma 3.18. *Let $>_M$ be a matrix order with first row $\mathbf{w}$. Then*

(i) $\mathbf{w} \in (\mathbb{R}^n)^+ = \{(a_1, \ldots, a_n) : a_i \geq 0, 1 \leq i \leq n\}$,

(ii) Every nonzero $\mathbf{w} \in (\mathbb{R}^n)^+$ is the first row of some matrix M such that $>_M$ is an admissible ordering.

(iii) Let M' be a matrix such that the matrix orders $>_M$ and $>_{M'}$ are equal. Then their first rows satisfy $\mathbf{w} = \lambda \mathbf{w}'$ for some $0 < \lambda \in \mathbb{R}$.

Proof. The lemma easily follows from the definitions of admissible ordering and matrix order. □

We call $(\mathbb{R}^n)^+$ the *positive orthant* in $\mathbb{R}^n$. Previous lemma implies that each admissible ordering determines a well-defined ray in the positive orthant $(\mathbb{R}^n)^+$. Even different orderings might give the same ray (e.g., all of the graded orderings give the ray of positive multiples of $(1, \ldots, 1)$). Therefore, this leads to problems involving cones in the positive orthant.

Let I be an ideal, $G = \{g_1, \ldots, g_t\}$ one of the finitely many marked Gröbner bases of I, with $\mathrm{LT}(g_i) = x^{\alpha(i)}$, and $N = \langle x^{\alpha(1)}, \ldots, x^{\alpha(t)} \rangle$ the corresponding element of $\mathrm{Mon}(I)$. We will now concentrate on finding the set C_G of all admissible orderings for which G is the corresponding marked Gröbner basis of I. We write

$$g_i = x^{\alpha(i)} + \sum_{\beta} c_{i,\beta} x^{\beta},$$

where $x^{\alpha(i)} > x^{\beta}$ for all $c_{i,\beta} \neq 0$. By the lemma 3.18, each order $>$ in C_G comes from a matrix M. So in particular, to find the leading terms we compare monomials first according to the first row $\mathbf{w}$ of the matrix.

If $\alpha(i) \cdot \mathbf{w} > \beta \cdot \mathbf{w}$ for all β with $c_{i,\beta} \neq 0$, the single weight vector $\mathbf{w}$ selects the correct leading term in g_i as the term of highest weight. As we know, however, we may have a tie in the first comparison, in which case we would have to make further comparisons using the other rows of M. This suggests that we should consider the following set of vectors:

$$\begin{aligned} C_G &= \{\mathbf{w} \in (\mathbb{R}^n)^+ : \alpha(i) \cdot \mathbf{w} \geq \beta \cdot \mathbf{w} \text{ whenever } c_{i,\beta} \neq 0\} \\ &= \{\mathbf{w} \in (\mathbb{R}^n)^+ : (\alpha(i) - \beta) \cdot \mathbf{w} \geq 0 \text{ whenever } c_{i,\beta} \neq 0\} \end{aligned} \tag{3.2}$$

The cone C_G has the property that if $>_M$ is a matrix ordering such that G is the marked Gröbner basis of I with respect to $>_M$, then the first row $\mathbf{w}$ of M lies in C_G.

Example 3.3.1. Consider the ideal $I = \langle x^2 - y^3, x^3 - y^2 + x\rangle \subset \mathbb{Q}[x, y]$. The marked Gröbner basis w.r.t. $>_{\text{grevlex}}$ with $x > y$ is

$$G = \{\underline{y^3} - x^2, \underline{x^3} - y^2 + x\},$$

where the leading terms are underlined. Let $\mathbf{w} = (a, b)$ be a vector in positive orthant of $\mathbb{R}^2$. Then $\mathbf{w}$ lies in C_G if and only if the following inequalities hold:

$$\begin{aligned}(0,3)\cdot(a,b) &\geq (2,0)\cdot(a,b)\\ (3,0)\cdot(a,b) &\geq (0,2)\cdot(a,b)\\ (3,0)\cdot(a,b) &\geq (1,0)\cdot(a,b).\end{aligned}$$

This can be restated as $3a \geq 2b$ and $3b \geq 2a$. Those two inequalities define a ray in positive orthant of $\mathbb{R}^2$ bordered by the lines $3a = 2b$ and $3b = 2a$.

The collection of all cones C_G where G ranges over all marked Gröbner bases of I is a *Gröbner fan*. Fan in general consists of finitely many closed convex polyhedral cones with vertex at the origin with the following properties:

- Any face of a cone in the fan is also in the fan.
- The intersection of two cones in the fan is a face of each.

The Gröbner fan encodes information about all marked Gröbner bases of an ideal.

Example 3.3.2. The Gröbner fan of $I = \langle x^2 - y^3, x^3 - y^2 + x\rangle$ has seven cones corresponding to the marked Gröbner bases:

$$\begin{aligned}G_1 &= \{\underline{y^9} + 2y^6 - y^4 + y^3, \underline{x} - y^7 + y^4 - y^2\},\\ G_2 &= \{\underline{y^7} + y^4 - y^2 + x, \underline{xy^2} - y^6 - y^3, \underline{x^2} - y^3\},\\ G_3 &= \{\underline{y^6} - xy^2 + y^3, \underline{xy^3} - y^2 + x, \underline{x^2} - y^3\},\\ G_4 &= \{\underline{y^3} - x^2, \underline{x^3} - y^2 + x\},\\ G_5 &= \{\underline{y^2} - x^3 - x, \underline{x^3y} - x^2 + xy, \underline{x^6} + 2x^4 - x^2y + x^2\},\\ G_6 &= \{\underline{y^2} - x^3 - x, \underline{-x^6} - 2x^4 + x^2y - x^2, \underline{x^7} + 2x^5 + x^3 - x^2 + xy\},\\ G_7 &= \{\underline{y^2} - x^3 - x, \underline{xy} - x^7 + 2x^5 + x^3 - x^2, \underline{x^8} + 3x^6 + 3x^4 - x^3 + x^2\}.\end{aligned}$$

The Gröbner fan of I is pictured in Figure 3.1, where C_{G_1} is the cone bordering with axis a. The remaining cones are labeled so that C_{G_i} borders with $C_{G_{i-1}}$ and C_{G_7} is the cone bordering with axis b.

In the following part we will present the *Gröbner Walk* algorithm which is an interesting application of the Gröbner fan. The FGLM algorithm from previous section works only for zero-dimensional ideals. However, the Gröbner Walk is a general method for basis conversion.

In order to convert a given basis, we will repeat two simple steps:

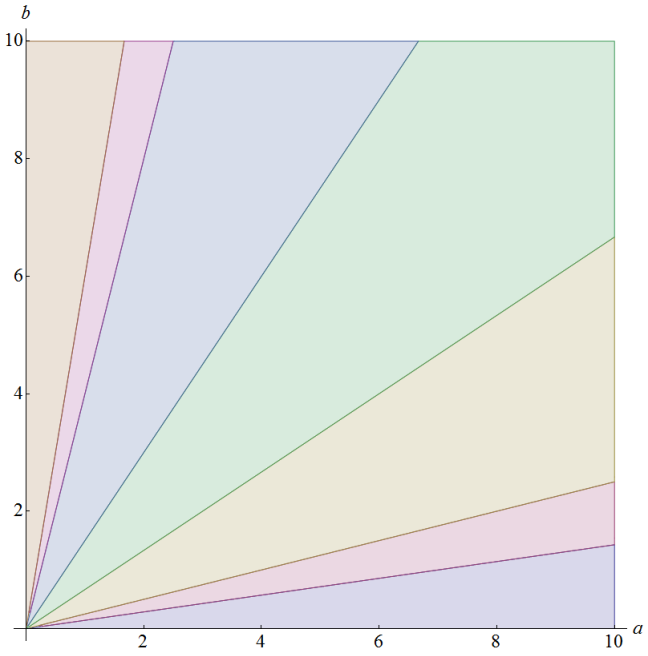


Figure 3.1: Gröbner Fan of $I = \langle x^2 - y^3, x^3 - y^2 + x \rangle$ in $\mathbb{R}^2$.

- Crossing from one cone to the next
- Computing the Gröbner basis of I corresponding to the new cone.

Suppose we have a marked Gröbner basis G of I w.r.t. some admissible ordering $>_s$. We call $>_s$ the *start ordering* for the walk. The goal is to compute a Gröbner basis of I w.r.t. some other given *target ordering* $>_t$. We will represent the orderings $>_s$ and $>_t$ by matrices M_s and M_t respectively with first rows $\mathbf{w}_s$ and $\mathbf{w}_t$.

We will follow a piecewise linear path from $\mathbf{w}_s$ to $\mathbf{w}_t$ in $(\mathbb{R}^n)^+$. The two steps are repeated until the end of the path is reached and at that point we have the desired Gröbner basis of I w.r.t. the target ordering.

Lets discuss the two steps separately.

3.3.1 Crossing Cones

Lets assume that we have the marked Gröbner basis G_{old} corresponding to the cone C_{old}, and a matrix M_{old} with first row $\mathbf{w}_{\text{old}}$ representing $>_{\text{old}}$. As we continue along the path from $\mathbf{w}_{\text{old}}$, let $\mathbf{w}_{\text{new}}$ be the last point on the path that lies in cone C_{old}.

The new weight vector $\mathbf{w}_{\text{new}}$ may be computed as follows. Let $G_{\text{old}} = \{x^{\alpha(i)} + \sum_{i,\beta} c_{i,\beta} x^{\beta} : 1 \leq i \leq t\}$ where $x^{\alpha(i)}$ is the leading term w.r.t. $>_{M_{\text{old}}}$. To simplify notation, let $v_1, \ldots, v_m$ denote the vectors $\alpha(i) - \beta$ where $1 \leq i \leq t$ and $c_{i,\beta} \neq 0$. By the definition of cone 3.2, C_{old} consists of those points $\mathbf{w}$ in $(\mathbb{R}^n)^+$ for which

$$\mathbf{w} \cdot v_j \geq 0, \quad 1 \leq j \leq m$$

For simplicity say that the remaining portion of the path to be traversed consists of the straight line segment from $\mathbf{w}_{\text{old}}$ to $\mathbf{w}_t$. This segment can be parametrized as $(1-u)\mathbf{w}_{\text{old}} + u\mathbf{w}_t$, for $u \in [0,1]$, we see that the point for the parameter value u lies in C_{old} if and only if

$$(1-u)(\mathbf{w}_{\text{old}} \cdot v_j) + u(\mathbf{w}_t \cdot v_j) \geq 0, \quad 1 \leq j \leq m.$$

Then $\mathbf{w}_{\text{new}} = (1-u_{\text{last}})\mathbf{w}_{\text{old}} + u_{\text{last}}\mathbf{w}_t$, where u_{last} is computed by the procedure given in Algorithm 3.6.

Algorithm 3.6 Procedure NextCone

Input: $\mathbf{w}_{\text{old}}, \mathbf{w}_t, v_1, \ldots, v_m$
Output: u_{last}

```
u_last = 1
for j = 1 to m do
  if w_t · v_j < 0 then
    u_j = (w_old · v_j) / (w_old · v_j − w_t · v_j)
  if u_j < u_last then
    u_last = u_j
return u_last
```

The idea of the algorithm is that if $\mathbf{w}_t \cdot v_j \geq 0$, then the condition holds for all $u \in [0,1]$ since $\mathbf{w}_{\text{old}} \cdot v_j \geq 0$. On the other hand, if $\mathbf{w}_t \cdot v_j < 0$, then the formula for u_j gives the largest value of u such that the condition holds for this particular j. Note that $0 \leq u_j \leq 1$ in this case.

Once we have $\mathbf{w}_{\text{new}}$, we need to choose the next cone in the Gröbner fan. Let $>_{\text{new}}$ be the ordering where we first compare $\mathbf{w}_{\text{new}}$-weights and break ties using the target ordering. Since $>_t$ is represented by M_t, it follows that $>$ new is represented by $\binom{\mathbf{w}_{\text{new}}}{M_t}$. This gives the cone C_{new} .

Furthermore, if we are in the situation where M_t is the bottom of the matrix representing $>_{\text{old}}$, the following lemma shows that whenever $\mathbf{w}_{\text{old}} \neq \mathbf{w}_t$, the above process is guaranteed to move us closer to $\mathbf{w}_t$.

Lemma 3.19. *Let u_{last} be as in Algorithm 3.6 and assume that $>_{\text{old}}$ is represented by $\binom{\mathbf{w}_{\text{new}}}{M_t}$. Then $u_{\text{last}} > 0$.*

Proof. From Algorithm 3.6, $u_{\text{last}} = 0$ implies that $\mathbf{w}_{\text{old}} \cdot v_j = 0$ and $\mathbf{w}_t \cdot v_j < 0$ for some j. But recall that $v_j = \alpha(i) - \beta$ for some $g = x^{\alpha(i)} + \sum_{i,\beta} c_{i,\beta} x^{\beta} \in G$, where $x^{\alpha(i)}$ is the leading term for $>_{\text{old}}$ and $c_{i,\beta} \neq 0$. It follows that

$$\mathbf{w}_{\text{old}} \cdot \alpha(i) = \mathbf{w}_{\text{old}} \cdot \beta \ \text{ and } \ \mathbf{w}_t \cdot \alpha(i) < \mathbf{w}_t \cdot \beta.$$

Since $>_{\text{old}}$ is represented by $\binom{\mathbf{w}_{\text{old}}}{M_t}$, the above equality tells us that $x^{\alpha(i)}$ and x^{β} have the same $\mathbf{w}_{\text{old}}$-weight, so that we break the tie using M_t. But $\mathbf{w}_t$ is the first row of M_t, so that the above inequality implies that $x^{\alpha(i)}$ is not the leading term for $>_{\text{old}}$. This contradiction proves the lemma. □

3.3.2 Converting Gröbner Bases

Once we have crossed from C_{old} into C_{new}, we need to convert the marked Gröbner basis G_{old} into a Gröbner basis for I w.r.t. the admissible ordering $>_{\text{new}}$ represented by $\begin{pmatrix} \mathbf{w}_{\text{new}} \\ M_t \end{pmatrix}$. This is done as follows.

The key feature of $\mathbf{w}_{\text{new}}$ is that it lies on the boundary of C_{old}, so that some of the inequalities defining C_{old} become equalities. This means that the leading term of some $g \in G_{\text{old}}$ has the same $\mathbf{w}_{\text{new}}$-weight as some other term in g. In general, given a weight vector $\mathbf{w}$ in the positive orthant $(\mathbb{R}^n)^+$ and a polynomial $f \in k[x_1, \ldots, x_n]$, the *initial form* of f for $\mathbf{w}$, denoted $\text{in}_{\mathbf{w}}(f)$, is the sum of all terms in f of maximum $\mathbf{w}$-weight. Also given a set S of polynomials, we let $\text{in}_{\mathbf{w}}(S) = \{\text{in}_{\mathbf{w}}(f) : f \in S\}$.

Using this notation, we can form the ideal $\langle \text{in}_{\mathbf{w}_{\text{new}}}(G_{\text{old}}) \rangle$ of $\mathbf{w}_{\text{new}}$-initial forms of elements of G_{old}. Note that $\mathbf{w}_{\text{new}} \in C_{\text{old}}$ guarantees that the marked term of $g \in G_{\text{old}}$ appears in $\text{in}_{\mathbf{w}_{\text{new}}}(g)$. The important thing to realize here is that in nice cases, $\text{in}_{\mathbf{w}_{\text{new}}}(G_{\text{old}})$ consists mostly of monomials, together with a small number of polynomials.

It follows that finding a monic Gröbner basis $H = \{h_1, \ldots, h_s\}$ of $\langle \text{in}_{\mathbf{w}_{\text{new}}}(G_{\text{old}}) \rangle$ w.r.t. $>_{\text{new}}$ may usually be done very quickly. The surprise is that once we have H, it is relatively easy to convert G_{old} into the desired Gröbner basis.

Theorem 3.20. *Let G_{old} be the marked Gröbner basis for an ideal I w.r.t. $>_{\text{old}}$. Also let $>_{\text{new}}$ be represented by $\begin{pmatrix} \mathbf{w}_{\text{new}} \\ M_t \end{pmatrix}$, where $\mathbf{w}_{\text{new}}$ is any weight vector in C_{old}, and let H be the monic Gröbner basis of $\langle \text{in}_{\mathbf{w}_{\text{new}}}(G_{\text{old}}) \rangle$ w.r.t. $>_{\text{new}}$ as above. Express each $h_j \in H$ as*

$$h_j = \sum_{g \in G_{\text{old}}} p_{j,g} \, \text{in}_{\mathbf{w}_{\text{new}}}(g).$$

Then replacing the initial forms by the g themselves, the polynomials

$$\bar{h}_j = \sum_{g \in G_{\text{old}}} p_{j,g} \, g, \quad 1 \leq j \leq s,$$

form a Gröbner basis $\bar{H}$ of I w.r.t. $>_{\text{new}}$.

Proof. Proof is rather technical and can be found in Cox, Little, O'Shea [21] in chapter 8, §5. □

The Gröbner basis $\bar{H}$ from the Theorem 3.20 is minimal, but not necessarily reduced. Hence a complete interreduction is usually necessary to obtain the marked Gröbner basis G_{new} corresponding to the next cone. However, this is a relatively quick process in practice. The process of replacing initial forms of $g \in G$ by g themselves is called *lifting* the initial forms to the new Gröbner basis.

Combining all of the previous thoughts, we can give a description of the Gröbner Walk in Algorithm 3.7. NextCone is defined in Algorithm 3.6. Lift is a procedure that lifts the initial forms to the Gröbner basis G_{new}, following Theorem 3.20. Gbasis$(I, >)$ computes a Gröbner basis of I w.r.t. $>$ and Interreduce$(G, >)$ takes a given set of polynomials G and interreduces them w.r.t. $>$.

Algorithm 3.7 Gröbner Walk

Input: M_s and M_t representing start and target orders with first rows $\mathbf{w}_s$ and $\mathbf{w}_t$, G_s a Gröbner basis w.r.t. $>_{M_s}$

Output: G_{new} a Gröbner basis of $\langle G_s \rangle$ w.r.t. $>_{M_t}$

$M_{\text{old}} = M_s$
$G_{\text{old}} = G_s$
$\mathbf{w}_{\text{new}} = \mathbf{w}_s$
$M_{\text{new}} = \begin{pmatrix} \mathbf{w}_{\text{new}} \\ M_t \end{pmatrix}$
done = false
while done = false **do**
  $In = \text{in}_{\mathbf{w}_{\text{new}}}(G_{\text{old}})$
  $InG = \text{Gbasis}(In, >_{M_{\text{new}}})$
  $G_{\text{new}} = \text{Lift}(InG, G_{\text{old}}, In, M_{\text{new}}, M_{\text{old}})$
  $G_{\text{new}} = \text{Interreduce}(G_{\text{new}}, M_{\text{new}})$
  $u = \text{NextCone}(G_{\text{new}}, \mathbf{w}_{\text{new}}, \mathbf{w}_t)$
  if $\mathbf{w}_{\text{new}} = \mathbf{w}_t$ **then**
    done = true
  else
    $M_{\text{old}} = M_{\text{new}}$
    $G_{\text{old}} = G_{\text{new}}$
    $\mathbf{w}_{\text{new}} = (1-u)\mathbf{w}_{\text{new}} + u\mathbf{w}_t$
    $M_{\text{new}} = \begin{pmatrix} \mathbf{w}_{\text{new}} \\ M_t \end{pmatrix}$
return G_{new}

Theorem 3.21. *The Gröbner Walk described in Algorithm 3.7 is correct and terminates.*

Proof. We traverse the line segment from $\mathbf{w}_s$ to $\mathbf{w}_t$. To prove termination, observe that by Theorem 3.15, we will pass only finitely many cones. The procedure NextCone always moves us farther along the line segment. Hence we must eventually reach $\mathbf{w}_t$, at which point the algorithm terminates.

To prove correctness, observe that in each pass through the main loop, the assumptions of Theorem 3.20 are satisfied. Furthermore, once the value of $\mathbf{w}_{\text{new}}$ reaches $\mathbf{w}_t$, the next pass through the loop computes a Gröbner basis of I for the admissible ordering represented by $\begin{pmatrix} \mathbf{w}_t \\ M_t \end{pmatrix}$. Now, one can easily see that the returned G_{new} is the marked Gröbner basis for $>_t$. □

Since Gröbner Walk is a very general method, there are no good results on complexity of the algorithm. Still, some observations can be made. The Complexity of the Gröbner Walk depends mostly on number of cones that are passed along the path through the Gröbner fan, and also on the number of cones that contain the point $\mathbf{w}_{\text{new}}$ at each step.

Example 3.3.3. We will use the Example 3.3.2 from this section and try to convert the Gröbner basis $G_s = G_4 = \{\underline{y^3} - x^2, \underline{x^3} - y^2 + x\}$ w.r.t. $>_{\text{grevlex}}$ to a

Gröbner basis w.r.t. $>_{\text{lex}}$. We will proceed as follows. Let

$$M_s = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix},$$

so $\mathbf{w}_s = (1,1)$. The target ordering can be represented by

$$M_t = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and $\mathbf{w}_t = (1,0)$. From now on, we will choose square matrices defining the appropriate admissible orderings by deleting linearly dependent rows. Thus, we begin by considering the ordering defined by

$$M_{\text{new}} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

The $\mathbf{w}_{\text{new}}$-initial forms of the Gröbner basis polynomials w.r.t. this ordering are the same as those for G_s, so the basis does not change in the first pass through the main loop.

We then enter the NextCone procedure with $\mathbf{w}_{\text{new}}$ in place of $\mathbf{w}_{\text{old}}$. The cone of $>_{M_{\text{new}}}$ is defined by the three inequalities obtained by comparing y^3 vs. x^2 and x^3 vs. y^2 and x^3 vs. x. The u_{last} is computed as follows:

$$\begin{aligned}
v_1 &= (-2,3), \mathbf{w}_t \cdot v_1 = -2 < 0 \Rightarrow u_1 = \frac{\mathbf{w}_{\text{new}} \cdot v_1}{\mathbf{w}_{\text{new}} \cdot v_2 - (-2)} = \frac{1}{3} \\
v_2 &= (3,-2), \mathbf{w}_t \cdot v_2 = 3 \geq 0 \Rightarrow u_2 = 1 \\
v_3 &= (2,0), \mathbf{w}_t \cdot v_1 = 2 \geq 0 \Rightarrow u_3 = 1
\end{aligned}$$

The smallest value is $u_1 = \frac{1}{3}$. Thus, the new weight vector is $\mathbf{w}_{\text{new}} = (1-\frac{1}{3})(1,1) + \frac{1}{3}(1,0) = (1, \frac{1}{3})$, and M_{old} and

$$M_{\text{new}} = \begin{pmatrix} 1 & \frac{1}{3} \\ 1 & 0 \end{pmatrix}$$

are updated for the next pass through the main loop.

In the second pass, $In = \{y^3 - x^2, x^3\}$. We compute the Gröbner basis for $\langle In \rangle$ w.r.t. $>_{\text{new}}$, and find

$$H = \{x^2 - y^3, xy^3, y^6\}.$$

In the terms of the generators for $\langle In \rangle$, we have

$$\begin{aligned}
-1 \cdot (y^3 - x^2) + 0 \cdot x^3 &= x^2 - y^3, \\
x \cdot (y^3 - x^2) + 1 \cdot x^3 &= xy^3, \\
(y^3 + x^2) \cdot (y^3 - x^2) + x \cdot x^3 &= y^6.
\end{aligned}$$

So by the Theorem 3.20, to get the next Gröbner basis, we lift to

$$\begin{aligned} -1 \cdot (y^3 - x^2) + 0 \cdot (x^3 - y^2 + x) &= x^2 - y^3, \\ x \cdot (y^3 - x^2) + 1 \cdot (x^3 - y^2 + x) &= xy^3 - y^2 + x, \\ (y^3 + x^2) \cdot (y^3 - x^2) + x \cdot (x^3 - y^2 + x) &= y^6 - xy^2 + x^2. \end{aligned}$$

Interreducing w.r.t. $>_{\text{new}}$, we obtain the marked Gröbner basis G_{new} given by

$$\{\underline{x^2} - y^3, \underline{xy^3} - y^2 + x, \underline{y^6} - xy^2 + y^3\}.$$

We continue along the path from $\mathbf{w}_{\text{new}}$ to $\mathbf{w}_t$. When the Gröbner Walk terminates, it returns

$$G_t = G_1 = \{\underline{y^9} + 2y^6 - y^4 + y^3, \underline{x} - y^7 + y^4 - y^2\},$$

the marked Gröbner basis w.r.t. $>_{\text{lex}}$.

Chapter 4

Applications in Cryptography

In this chapter we will sum up-to-date applications of Gröbner basis techniques used in cryptography. Unfortunately, the research in this area was for a while influenced by famous paper *"Why you cannot even hope to use Gröbner Bases in Public Key Cryptography: An open letter to a scientist who failed and a challenge to those who have not yet failed"*, published under few pseudonyms [7]. This paper caused many cryptographers to abandon Gröbner bases approaches. However, the authors focused on constructing a secure public key cryptosystem based on hardness of Gröbner bases problems. On the other hand, the use of Gröbner bases approaches for different applications in cryptography can not be dismissed because of the arguments of that article.

4.1 Algebraic Cryptanalysis

We begin with the definition of *cryptosystem*, a basic structure used in cryptography.

Definition 4.1. *A* cryptosystem *consists of the following parts:*

1. *A set $\mathcal{P}$, called the set of* plaintext units.
2. *A set $\mathcal{C}$, called the set of* ciphertext units.
3. *A set $\mathcal{K}$, called the* key space.
4. *For every key $k \in \mathcal{K}$, an* encryption map $\epsilon_k : \mathcal{P} \to \mathcal{C}$.
5. *For every key $k \in \mathcal{K}$, a* decryption map $\delta_k : \mathcal{C} \to \mathcal{P}$.
6. *A map $\eta : \mathcal{K} \to \mathcal{K}$ such that $\delta_{\eta(k)} \circ \epsilon_k = id_{\mathcal{P}}$ for all $k \in \mathcal{K}$. A pair $(k, \eta(k))$ is called a* key pair.

Cryptosystems are building blocks of modern cryptographic protocols that are being used every day all over the world. The definition of a cryptosystems naturally extends to so-called symmetric and public-key cryptosystems, depending

on the properties of the key pair. We say that cryptosystem is *symmetric* iff the η mapping is identity, i.e. the encryption key and decryption key are equal. Symmetric cryptosystems are block ciphers and stream ciphers. From now, we will focus on block ciphers.

Modern block ciphers repeatedly operate on blocks of plaintext. Each iteration is called a *round*. The round operations are chosen to provide diffusion(i.e. spread the influence of the key and the plaintext to all parts of the the ciphertext) and confusion(i.e. make the relationship between plaintext, ciphertext and key complicated). A round key is usually derived from the user key during the *key schedule*, to be used in the round operation.

In most practical cryptosystems, the plaintext and ciphertext spaces are some vector fields over a finite field. Such cryptosystems can be viewed as polynomial maps because of the next theorem .

Theorem 4.2. *Over a finite field k, every map $\phi : k^n \to k^m$ is polynomial, i.e. there exist polynomials $f_1, \ldots, f_m \in k[x_1, \ldots, x_n]$ such that:*

$$\phi(a_1, \ldots, a_n) = (f_1(a_1, \ldots, a_n), \ldots, f_m(a_1, \ldots, a_n))$$

for all $a_1, \ldots, a_n \in K$. The polynomials are however not uniquely determined.

Proof. Can be found in Bard [6]. □

This allows us to transform the problem of breaking given cryptosystem into the settings of polynomial ideals. It is the motivation for *algebraic cryptanalysis*. The idea of algebraic cryptanalysis is very simple. Lets write down given cryptosystem as a set of equations on input, key and the output, then try to solve arising system of equations. Some general methods of algebraic cryptanalysis are described in Kreuzer [28] and a nice overview of algebraic cryptanalysis of block ciphers can be found in Cid and Weinmann [17].

We should realize that there is the following the problem. We are interested only in solutions in the base field and not over the algebraic closure. Therefore, we also add so called *field polynomials* into our system.

Definition 4.3. *Let $k[x_1, \ldots, x_n]$ be a polynomial ring over finite field k of characteristics $q = p^n$, where p is prime and $n \in \mathbb{N}$. The* field polynomials *for $k[x_1, \ldots, x_n]$ are of form $x_i^q - x$ for every variable x_i in $k[x_1, \ldots, x_n]$.*

It is easy to see that if ideal $I \subset k[x_1, \ldots, x_n]$ has field polynomials in its generating set, then the variety $\mathrm{V}(I)$ lies in k^n, and this is exactly what we needed.

General Gröbner basis attack against a block cipher then consists of the following steps:

1. Set up a polynomial system $\mathcal{P} = \{p_i = 0\}$ for the cipher in question. The system $\mathcal{P}$ consists of both cipher and key schedule equations.

2. Get a plaintext-ciphertext pair $((P_0, \ldots, P_{t-1}), (C_0, \ldots, C_{t-1}))$ and add the corresponding linear equations

$$x_{0,0} + P_0 = 0, \ldots, x_{t-1,0} + P_{t-1} = 0$$
$$x_{0,r-1} + C_0 = 0, \ldots, x_{t-1,r-1} + C_{t-1} = 0$$

   to the system $\mathcal{P}$. Let I be the the ideal generated by the system $\mathcal{P}$ and the field polynomials for every variable occurring in $\mathcal{P}$. We call it the *key recovery ideal*.

3. Try to compute a Gröbner basis $G_{>_{\text{grevlex}}}$ of I w.r.t. $>_{\text{grevlex}}$.

4. If $G_{>_{\text{grevlex}}} = \{1\}$, i.e. the system is inconsistent, go to step 2, otherwise proceed.

5. Use some Gröbner basis conversion algorithm to obtain $G_{>_{\text{lex}}}$ a Gröbner basis w.r.t. $>_{\text{lex}}$. The variable ordering should be such that the key variables of the first round are the last elements.

6. Compute the variety $V = \mathrm{V}(I)$ using $G_{>_{\text{lex}}}$.

7. Request another plaintext-ciphertext pair (P', C').

8. Try all elements $v \in V$ as key candidates to encrypt P'. If k does not encrypt P' to C', remove v from V, otherwise retain.

9. If V still contains more than one element, go to step 7.

10. Terminate.

Some improvements to this attack based on the structure of the block cipher were proposed in Cid et. al [15]. They explained a conquer approach called *meet-in-the-middle*. The system consisting of r rounds is divided into two subsystems for $\frac{r}{2}$ rounds. We regard the input variables of the first equation subsystem as the input variables of the second equation subsystem. We can compute the Gröbner bases of the two corresponding subsystems, using appropriate ordering suitable for elimination, then eliminate variables that do not appear in rounds $\frac{r}{2}$ and $\frac{r}{2}+1$. This gives two smaller systems of equations and if we combine them with some additional equations from the key schedule, this system can be solved to find the key. It might sometimes be faster than directly attacking the cipher.

If we extend this method further, we get the *Gröbner surfing* proposed in Albrecht [3]. Here the Gröbner basis of the key recovery ideal is computed round by round adding at each step the equations for the new round to the previously computed Gröbner basis.

It might be interesting that the first use of Gröbner bases in symmetric-key cryptography was not a direct attack on a block cipher, but it was rather used for an improvement of the linear cryptanalysis of the Data Encryption Standard by Shimoyama and Kaneko [31]. Also different methods were proposed for solving the

polynomial systems arising from block ciphers. N. Courtois et. al. [20] formulated the XL algorithm, that uses linearization and Bard considers using SAT-solvers in [6].

4.2 Algebraic Properties of AES

The *Advanced Encryption Standard* (AES), also known as Rijndael, is the present U.S. standard block cipher. It can be elegantly described using only operations over the finite field $\mathbb{F}_{2^8}$ because of its relatively simple algebraic structure. It is an iterated block cipher with a block size of 128 bits and a key size $128, 192$ and 256 bits. The three versions are denoted AES-128, AES-192 and AES-256 respectively. The key size determines number of rounds; AES-128, AES-192 and AES-256 have 10,12 and 14 rounds respectively.

The standard view of the AES is as a series of operations on a square array of 16 bytes called *state*. The 128 bits of input are represented as a string of 16 bytes $\mathcal{B}_0, \ldots, \mathcal{B}_{15}$ and then put into the state array in the following order:

$\mathcal{B}_0$	$\mathcal{B}_4$	$\mathcal{B}_8$	$\mathcal{B}_{12}$
$\mathcal{B}_1$	$\mathcal{B}_5$	$\mathcal{B}_9$	$\mathcal{B}_{13}$
$\mathcal{B}_2$	$\mathcal{B}_6$	$\mathcal{B}_{10}$	$\mathcal{B}_{14}$
$\mathcal{B}_3$	$\mathcal{B}_7$	$\mathcal{B}_{11}$	$\mathcal{B}_{15}$

Every AES round consists of four state transformations: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. The last round is an exception because there is no `MixColumns`. And one additional operation `AddRoundKey` is done before the first round.

We will give a detailed description of the AES operations, but first we have to explain that the AES standard uses a representation of a byte in $\mathbb{F}_{2^8}$. We take an irreducible polynomial $m = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$. The field is then defined either as the the quotient field $\mathbb{F}_2/\langle m \rangle$, or as the extension field $\mathbb{F}_2(\theta)$, where θ is a root of m. We refer to the field $\mathbb{F}_{2^8}$ defined by m as the *Rijndael field* and denote it $\mathbf{F}$. Possible representation of a byte $b_7b_6b_5b_4b_3b_2b_1b_0$ in $\mathbf{F}$ is then

$$b_7\theta^7 + b_6\theta^6 + b_5\theta^5 + b_4\theta^4 + b_3\theta^3 + b_2\theta^2 + b_1\theta^1 + b_0\theta^0.$$

We will sometimes represent the byte as a hexadecimal number, e.g. $\theta^6 + \theta^5 + \theta + 1 = \texttt{63}$.

Thus any input block of 16 bytes $\sigma_0, \ldots, \sigma_{15}$ can be interpreted as the square state matrix over $\mathbf{F}$:

$$\begin{pmatrix} \sigma_0 & \sigma_4 & \sigma_8 & \sigma_{12} \\ \sigma_1 & \sigma_5 & \sigma_9 & \sigma_{13} \\ \sigma_2 & \sigma_6 & \sigma_{10} & \sigma_{14} \\ \sigma_3 & \sigma_7 & \sigma_{11} & \sigma_{15} \end{pmatrix}.$$

And we can now continue with the description of the AES operations.

`SubBytes`

This operation is a simple permutation of byte and is given by an S-box S in a form of a look-up table. Every state byte is permuted using the S-box, giving the output of `SubBytes`. The S-box is in fact a composition of the following three operations:

- *Iversion.* It is extended inversion on the Rijndael field $\mathbf{F}$, so that $0 \mapsto 0$. Input to the S-box is regarded as an element $w \in \mathbf{F}$ and for $w \neq 0$ the output x satisfies $x = w^{-1}$ and $wx = 0$.

- $\mathbb{F}_2$*-linear mapping.* The mapping ζ is given by a circulant matrix, and the output is $y = \zeta(x)$, where

$$\begin{pmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix}$$

- *S-box constant.* The output byte of ζ is added to the element `63` as an element of $\mathbf{F}$.

`ShiftRows`

Each row i $(0 \leq i \leq 3)$ of the 4×4 state matrix is rotated to the left by i positions

$$\begin{pmatrix} \mathcal{S}_{0,0} & \mathcal{S}_{0,1} & \mathcal{S}_{0,2} & \mathcal{S}_{0,3} \\ \mathcal{S}_{1,0} & \mathcal{S}_{1,1} & \mathcal{S}_{1,2} & \mathcal{S}_{1,3} \\ \mathcal{S}_{2,0} & \mathcal{S}_{2,1} & \mathcal{S}_{2,2} & \mathcal{S}_{2,3} \\ \mathcal{S}_{3,0} & \mathcal{S}_{3,1} & \mathcal{S}_{3,2} & \mathcal{S}_{3,3} \end{pmatrix} \mapsto \begin{pmatrix} \mathcal{S}_{0,0} & \mathcal{S}_{0,1} & \mathcal{S}_{0,2} & \mathcal{S}_{0,3} \\ \mathcal{S}_{1,1} & \mathcal{S}_{1,2} & \mathcal{S}_{1,3} & \mathcal{S}_{1,0} \\ \mathcal{S}_{2,2} & \mathcal{S}_{2,3} & \mathcal{S}_{2,0} & \mathcal{S}_{2,1} \\ \mathcal{S}_{3,3} & \mathcal{S}_{3,0} & \mathcal{S}_{3,1} & \mathcal{S}_{3,2} \end{pmatrix}.$$

`MixColumns`

Each column of the state matrix is regarded as a vector and it is multiplied by an MDS matrix

$$\begin{pmatrix} \mathcal{S}_{0,j} \\ \mathcal{S}_{1,j} \\ \mathcal{S}_{2,j} \\ \mathcal{S}_{3,j} \end{pmatrix} \mapsto \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} \mathcal{S}_{0,j} \\ \mathcal{S}_{1,j} \\ \mathcal{S}_{2,j} \\ \mathcal{S}_{3,j} \end{pmatrix}.$$

`AddRoundKey`

The AES key schedule processes the user-supplied key to give the 16-byte round keys $\mathcal{K}_{r,0}, \ldots, \mathcal{K}_{r,15}$ $(0 \leq r \leq N_r)$. In round r, `AddRoundKey` updates the state array by $\sigma_i \mapsto \sigma_i + \mathcal{K}_{r,i}$ $(0 \leq i \leq 15)$ or equivalently by $\mathcal{S}_{i,j} \mapsto \mathcal{S}_{i,j} + \mathcal{K}_{r,4j+i}$ $(0 \leq i, j \leq 3)$.

The generation of the AES round keys is straightforward even though three key sizes are supported. We only give a description of the key schedule for AES-128. We assume that the round key at round r ($0 \leq r \leq 10$) is given by $\mathcal{K}_{r,0}, \ldots, \mathcal{K}_{r,15}$ where the user-supplied key forms the round key at round 0. In order to form the round key for round $s = r + 1$, we first define a temporary word $\mathcal{T}_0\mathcal{T}_1\mathcal{T}_2\mathcal{T}_3$ of four bytes by

$$\mathcal{T}_0 = S[\mathcal{K}_{r,13}] + \theta^r,\ \mathcal{T}_1 = S[\mathcal{K}_{r,14}],\ \mathcal{T}_2 = S[\mathcal{K}_{r,15}],\ \text{and}\ \mathcal{T}_3 = S[\mathcal{K}_{r,12}],$$

where θ is the Rijndael root. The key for round s is then given by

$$\mathcal{K}_{s,i} = \begin{cases} \mathcal{K}_{r,i} + \mathcal{T}_i & 0 \leq i \leq 3 \\ \mathcal{K}_{r,i} + \mathcal{K}_{s,i-4} & 4 \leq i \leq 15. \end{cases}$$

All of the operations used during encryption can be inverted, thus the decryption is also very simple. For a thorough description of algebraic properties of AES see Cid, Murphy and Robshaw [15].

4.2.1 System of equations over $\mathbb{F}_2$

We now derive a system of multivariate equations for AES-128 over $\mathbb{F}_2$. Some problems, that might occur, are discussed in Biryukov and De Canière [9]. The process of obtaining this system is described in Cid, Murphy and Robshaw [15].

We start considering the linear equations. They can be obtained from the mapping from $\mathbb{F}_2^{128}$ to $\mathbb{F}_2^{128}$ that transforms the output of S-box inversion into the AES round output:

$$\mathbf{x} \mapsto CR(L\mathbf{x} + \mathbf{63}) + \mathbf{k}_i,$$

where C, R and L are matrices over $\mathbb{F}_2$ corresponding to `MixColumns`, `ShiftRows` and the $\mathbb{F}_2$-linear mapping in S-box respectively. There are similar affine mappings which relate the plaintext to the input of the S-box inversion in the first round and the output of S-box inversion in the last round to the ciphertext. Thus there are $11 \cdot 128 = 1408$ linear equations in this system.

The non-linear equations are given by the S-box inversion. We assume that there is no inversion of zero. The defining relation over $\mathbf{F}$ between input w and output x is $wx = 1$. This is then translated to the components of w and x, giving rise to eight quadratic equations in bits of S-box input and out. Moreover, there are other relations that we have to consider, i.e. $wx^2 = x, w^2x = w, wx^4 = xx^2$ and $w^4x = ww^2$. Altogether, we end up with 40 quadratic multivariate equation over $\mathbb{F}_2$ for the S-box inversion.

We can now start counting the variables and equations. The equations are very sparse, in which the variables represent the input and output of the inversion operations and the round keys. We also require some auxiliary key variables to describe the key schedule.

We first note that since the equation system is over $\mathbb{F}_2$, any variable z satisfies the field equation $z^2 + z = 0$. In the equation system for an AES encryption, there are 1280 inversion input and 1280 inversion output variables. These are

	Source		
	State	Key schedule	Total
Variables	2560	1728	4288
Field equations	2560	1728	4288
Linear equations	1408	1280	2688
Inversion equations	6400	1600	8000
Overall equations	10368	4608	14976

Figure 4.1: The number of variables and equations in a sparse polynomial system over $\mathbb{F}_2$ for AES.

used in 1408 linear equations and together give 2560 field equations. There are $10 \cdot 16$ S-box inversions with 40 quadratic equations each, giving 6400 quadratic equations in total.

In the AES key schedule, there are $11 \cdot 128$ round key variables. The S-box inversion is only applied to four of sixteen round key bytes, so the key uses $4 \cdot 10$ inversions. We also add the components of key schedule inversion output as variables; there are $8 \cdot 40 = 320$ of those, thus giving 1728 key schedule variables in total. We have $40 \cdot 40 = 1600$ quadratic equations, 1728 field equations, and $10 \cdot 128$ linear equations in the key schedule.

This is summarised in the Figure 4.1. We see that overall we obtain a system with 14976 equations in 4288 variables. We can, however, try to produce a more compact system of equations using the linear equations in AES to substitute and eliminate variables.

We first consider the linear relations in the encryption process. Let $\mathbf{w}_i$ and $\mathbf{x}_i$ denote the input to and output from an AES inversion respectively, and let $\mathbf{k}_i$ denote the round key and 63 the vector of repeated S-box constants. These are considered as vectors in $\mathbb{F}_2^{128}$. For plaintext $\mathbf{p}$ and ciphertext $\mathbf{c}$, AES encryption is described by

$$\begin{aligned} \mathbf{w}_0 &= \mathbf{p} + \mathbf{k}_0 + \mathbf{63} \\ \mathbf{w}_i &= M\mathbf{x}_{i-1} + \mathbf{k}_i + \mathbf{63} \quad i \in \{1, \ldots, 9\} \\ \mathbf{c} &= M^*\mathbf{x}_9 + \mathbf{k}_{10} + \mathbf{63}, \end{aligned}$$

where M^* is the modified matrix for the final round. We therefore write

$$\begin{aligned} \mathbf{x}_i &= M^{-1}(\mathbf{w}_{i+1} + \mathbf{k}_{i+1} + \mathbf{63}) \quad i \in \{0, \ldots, 8\} \\ \mathbf{x}_9 &= (M^*)^{-1}(\mathbf{k}_{10} + \mathbf{c} + \mathbf{63}), \end{aligned}$$

and eliminate the 1280 variables arising from the vectors $\mathbf{x}_i$ $(0 \leq i \leq 9)$.

We now consider linear relations in the key schedule. Let $\mathbf{s}_i$ denote the 32-bit output vector of the inversion operation in the AES key schedule. The output The relationship between successive round keys is given by

$$\begin{pmatrix} \mathbf{k}_i^{(0)} \\ \mathbf{k}_i^{(1)} \\ \mathbf{k}_i^{(2)} \\ \mathbf{k}_i^{(3)} \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & 0 \\ I & I & 0 & 0 \\ I & I & I & 0 \\ I & I & I & I \end{pmatrix} \cdot \begin{pmatrix} \mathbf{k}_{i-1}^{(0)} \\ \mathbf{k}_{i-1}^{(1)} \\ \mathbf{k}_{i-1}^{(2)} \\ \mathbf{k}_{i-1}^{(3)} \end{pmatrix} + \begin{pmatrix} Q(\mathbf{s}_{i-1} + \mathbf{r}_{i-1}) \\ Q(\mathbf{s}_{i-1} + \mathbf{r}_{i-1}) \\ Q(\mathbf{s}_{i-1} + \mathbf{r}_{i-1}) \\ Q(\mathbf{s}_{i-1} + \mathbf{r}_{i-1}) \end{pmatrix},$$

where $\mathbf{k}_i^{(j)}$ denotes a vector of length 32 corresponding to a column of the round key array. Thus there exists a 128×128 matrix A and a 128×32 matrix B over $\mathbb{F}_2$ such that

$$\mathbf{k}_i = A\mathbf{k}_{i-1} + B(\mathbf{s}_{i-1} + \mathbf{r}_{i-1}) = A^i\mathbf{k}_0 + \sum_{j=1}^{i} A^{j-1}B(\mathbf{s}_{i-j} + \mathbf{r}_{i-j}).$$

If we use the relation $\mathbf{k}_0 = \mathbf{w}_0 + \mathbf{p} + \mathbf{63}$ from encryption, we have

$$\mathbf{k}_i = A^i\mathbf{w}_0 + \sum_{j=1}^{i} A^{j-1}B\mathbf{s}_{i-j} + \sum_{j=1}^{i} A^{j-1}B\mathbf{r}_{i-j} + A^i(\mathbf{p} + \mathbf{63}),$$

and we can eliminate the $128 \cdot 11 = 1408$ variables that refer to the vectors $\mathbf{k}_i$ $(0 \leq i \leq 10)$.

We used the linear equations in the AES system to eliminate $1280 + 1408 = 2688$ variables, thus there remain $4288 - 2688 = 1600$ variables. This variables give 1600 field equations and together with 8000 quadratic equations for the inversions create a system of 9600 equations over 1600 variables. It is better, but still not good enough to allow successful algebraic attack.

4.2.2 Small Scale Variants of AES

The system for full AES-128 is too complicated to get any insight. Therefore it might be helpful to somehow shrink the problem. This motivated the definition of small scale variants of AES presented in Cid, Murphy, and Robshaw [16]. The idea of creating small educational versions of ciphers is quite common. Courtois described such cipher CTC (for its algebraic properties see Albrecht [3]) and two families of block ciphers FLURRY and CURRY were introduced in Buchmann, Pychkine and Weinmann [14].

The small scale variants of AES are $\mathrm{SR}(n, r, c, e)$ and $\mathrm{SR}^*(n, r, c, e)$, with $\mathrm{SR}(n, r, c, e)$ including a `MixColumns` operation in the last round. Both are parametrised in the following way.

- n is the number of rounds,
- r is the number of rows in the rectangular grid of the state,
- c is the number of columns in the rectangular grid of the state,
- e is the word size (in bits).

Both $\mathrm{SR}(n, r, c, e)$ and $\mathrm{SR}^*(n, r, c, e)$ have a block size of $r{\cdot}c{\cdot}e$ bits and the full AES is modeled by $\mathrm{SR}^*(10, 4, 4, 8)$. The data block is viewed as an $r \times c$ array of e-bit words. Useful small scale variants exist when both r and c are restricted to $1, 2$, or 4.

The word sizes $e = 4$ and $e = 8$ are the most relevant and are defined with respect to $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^8}$. The field $\mathbb{F}_{2^4}$ is defined by the primitive polynomial

Inversion in $\mathbb{F}_{2^4}$

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	0	1	9	E	D	B	7	6	F	2	C	5	A	4	3	8

$\mathbb{F}_2$-linear mapping in $\mathbb{F}_{2^4}$

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	0	D	B	6	7	A	C	1	E	3	5	8	9	4	2	F

Full S-box over $\mathbb{F}_{2^4}$ with S-box constant 6.

Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output	6	B	5	4	2	E	7	A	9	D	F	C	3	1	0	8

Figure 4.2: An equivalent S-box over $\mathbb{F}_{2^4}$ for small scale variants of AES.

$x^4 + x + 1$ over $\mathbb{Z}_2$ with root ρ. Thus $\mathrm{SR}(n, r, c, 4)$ uses the field $\mathbb{Z}_2[x]/\langle x^4 + x + 1\rangle$ or equivalently $\mathbb{Z}_2(\rho)$. Small scale variants over $\mathbb{F}_{2^8}$ use the Rijndael field $\mathbf{F}$.

We define a round of the small scale variants over the field $\mathbb{F}_{2^4}$ by describing variants of the AES operations. An S-box over $\mathbb{F}_{2^4}$ consists of the following three operations, which are summarized in Figure 4.2.

- *Inversion.* The first operation is an extended inversion in the field $\mathbb{F}_{2^4}$ with $0 \mapsto 0$.
- $\mathbb{F}_2$*-linear mapping in* $\mathbb{F}_{2^4}$*.* The 4×4 matrix over $\mathbb{F}_2$ used to define the $\mathbb{F}_2$-linear mapping is the circulant matrix

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

- *S-box constant.* The output from the S-box is produced by adding the S-box constant 6 to the output of the $\mathbb{F}_2$-linear mapping.

The remaining operations, i.e. `ShiftRows`, `MixColumns` and `AddRoundKey`, are equivalent to the original AES.

- `ShiftRows` is the simultaneous left rotation of row i in the state array by i positions for $0 \leq i \leq r - 1$.
- `MixColumns` multiplies each column of the state array by an invertible circulant MDS matrix over $\mathbb{F}_{2^e}$.
- `AddRoundKey`: The key schedule for an n-round small scale variant of the AES produces $n + 1$ subkey blocks. `AddRoundKey` simultaneously adds (as elements of $\mathbb{F}_{2^e}$) each element of the subkey block to some intermediate data block. Since the AES begins with an initial `AddRoundKey`, the small scale variants $\mathrm{SR}(n, r, c, e)$ also begin with this operation.

	Number of variables	Number of equations	Time in seconds
$\mathrm{SR}(2,1,1,4)$	36	68	0.12
$\mathrm{SR}(3,1,1,4)$	52	100	0.25
$\mathrm{SR}(4,1,1,4)$	68	132	0.37
$\mathrm{SR}(5,1,1,4)$	84	164	0.55
$\mathrm{SR}(6,1,1,4)$	100	196	0.69
$\mathrm{SR}(7,1,1,4)$	116	228	1.01
$\mathrm{SR}(8,1,1,4)$	132	260	1.34
$\mathrm{SR}(9,1,1,4)$	148	292	1.65
$\mathrm{SR}(10,1,1,4)$	164	324	1.88
$\mathrm{SR}(2,1,1,8)$	72	136	3.41
$\mathrm{SR}(3,1,1,8)$	104	200	54.88
$\mathrm{SR}(4,1,1,8)$	136	264	461.71
$\mathrm{SR}(5,1,1,8)$	168	328	N/A

Figure 4.3: The computation time of Gröbner basis finding for the system of equations generated by $\mathrm{SR}(r,1,1,e)$ over $\mathbb{F}_2$.

The small scale variants retain the algebraic properties of the AES as much as possible and the equation system for small scale variants of AES over $\mathbb{F}_2$ can be obtained in the same way as for the full AES-128 as proposed earlier. We performed some experimentations with the polynomial system. The experimentations were performed in Sage [33], an open-source mathematics software. Sage is actually equipped with a generator of polynomial equations for small scale variants of AES. We used this one to compare with our results and they was basically the same. To compute a Gröbner basis, we used the algorithm provided in the Gröbner basis computation framework PolyBoRi [11] designed for working with boolean polynomials. The results are shown in Figure 4.3. For $\mathrm{SR}(5,1,1,8)$ the computation did not even terminate and it is obvious that we can not hope to break full AES-128 using this techniques.

4.2.3 Zero-Dimensional Gröbner Basis for AES-128

We will show how to obtain a polynomial system describing AES-128 that is in fact already a Gröbner basis. This ideas were presented by Buchmann, Pyshkin and Weinmann [13].

Lets consider the round input as $\mathbf{w}_i = (w_{i,0}, \ldots, w_{i,15}) \in \mathbf{F}^{16}$ for $0 \leq i \leq 9$ and $\mathbf{k}_i = (k_{i,0}, \ldots, k_{i,15})$ the round key for $0 \leq i \leq 10$. Furthermore, we let $S(\mathbf{w}_i) = (g(w_{i,0}), \ldots, g(w_{i,15}))$ denote the output of the `SubBytes` operation, where the polynomial $g(z)$ is the interpolation polynomial for S-box and is given by:

$$\mathtt{05}z^{254} + \mathtt{09}z^{253} + \mathtt{F9}z^{251} + \mathtt{25}z^{247} + \mathtt{F4}z^{239} + \mathtt{01}z^{223} + \mathtt{B5}z^{191} + \mathtt{8F}z^{127} + \mathtt{63}.$$

If p and c denote the plaintext and ciphertext respectively, then AES encryp-

tion is given by

$$\begin{aligned} \mathbf{w}_0 &= \mathbf{p} + \mathbf{k}_0 \\ \mathbf{w}_i &= \overline{C}\,\overline{R}(S(\mathbf{w}_{i-1})) + \mathbf{k}_i \quad i \in \{1, \ldots, 9\} \\ \mathbf{c} &= \overline{R}(S(\mathbf{w}_9)) + \mathbf{k}_{10}, \end{aligned}$$

where $\overline{R}$ and $\overline{C}$ are the matrices corresponding to operations `ShiftRows` and `MixColumns`.

After rearranging the system, we obtain:

$$\begin{aligned} 0 &= \mathbf{w}_0 + \mathbf{k}_0 + \mathbf{p} \\ 0 &= S(\mathbf{w}_{i-1}) + (\overline{C}\,\overline{R})^{-1}(\mathbf{w}_i + \mathbf{k}_i) \quad i \in \{1, \ldots, 9\} \\ 0 &= S(\mathbf{w}_9) + \overline{R}^{-1}(\mathbf{k}_{10} + \mathbf{c}), \end{aligned}$$

This yields an equation system with 176 equations, of which 16 equations are linear and the other 160 equations each have total degree 254.

We can perform a similar rearrangement with the key schedule equations using inverse S-box, though its interpolation polynomial $h(z)$ is dense. Thus we obtain for $1 \leq i \leq 10$:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} h(k_{i,0} + k_{i-1,0} + \theta^{i-1}) \\ h(k_{i,1} + k_{i-1,1}) \\ h(k_{i,2} + k_{i-1,2}) \\ h(k_{i,3} + k_{i-1,3}) \\ h(k_{i,4} + k_{i-1,4}) \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} k_{i-1,15} \\ k_{i-1,12} \\ k_{i-1,13} \\ k_{i-1,14} \\ k_{i,0} \\ \vdots \\ k_{i,11} \end{pmatrix}.$$

Lets consider the system as polynomials in polynomial ring

$$R_{\text{AES}} = \mathbf{F}[w_{0,0}, \ldots, w_{0,15}, k_{0,0}, \ldots, k_{10,15}, w_{1,0}, \ldots, w_{9,15}],$$

and as an ordering choose the $>_{\text{grlex}}$ with

$$w_{0,0} < \ldots < w_{0,15} < k_{0,0} < \ldots < k_{10,15} < w_{1,0} < \ldots < w_{9,15}.$$

Under this ordering, the 160 polynomials of degree 254 modeling the encryption operation have $w_{i,j}^{254}$ as their leading monomial. The remaining 16 linear equations are those containing the plaintext and have $k_{0,j}$ as their leading monomial. In the key schedule, the linear equations have $k_{i,j}$ as their leading monomial for $1 \leq i \leq 10$ and $4 \leq j \leq 15$, whilst the nonlinear equations have $k_{i,j}^{254}$ as their leading monomial for $1 \leq i \leq 10$ and $0 \leq j \leq 3$. Thus the leading terms of all polynomials are pairwise coprime, and therefore the polynomial system modeling AES defined in such way is a Gröbner basis. We used a consequence of the definition of Gröbner basis, i.e. G is a Gröbner basis of $\langle G \rangle$ iff $\gcd(\mathrm{LT}(f), \mathrm{LT}(g)) = 1$ for all distinct $f, g \in G$.

It is quite surprising result, that we can obtain Gröbner basis directly as a description of the cipher. However, this fact seems to have no practical implications for security of AES. For ideal $I_{\mathrm{AES}} \subset R_{\mathrm{AES}}$ generated by the derived polynomials, the vector space dimension of $\dim(R_{\mathrm{AES}}/I_{\mathrm{AES}})$ can be computed using results from previous chapter and it is $254^{200} \approx 2^{1598}$. It makes it impossible to use FGLM algorithm to transform this basis into a lexicographic basis. Also the Gröbner Walk fails for this task.

Whether this can be transformed into a successful attack on AES is still an open question. Nevertheless, this results support the beliefs that the algebraic structure within AES might be eventually exploited.

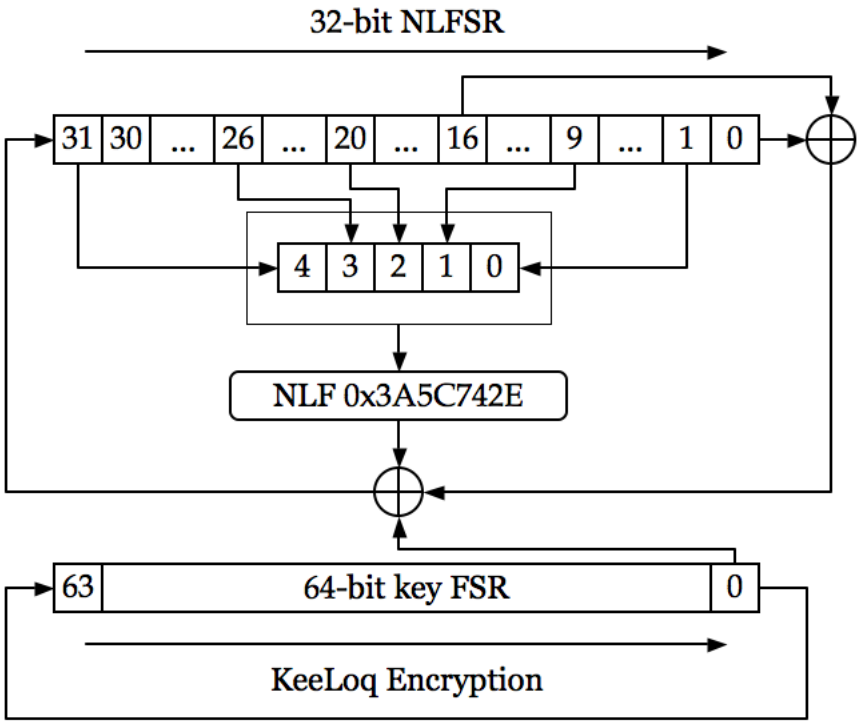


Figure 4.4: Scheme of KeeLoq encryption.

4.3 KeeLoq

KeeLoq is a block cipher used worldwide in remote entry systems by the automotive industry companies. It is an example of a real-life cipher broken by an algebraic attack. One of the first successful attacks was due to Gregory V. Bard and is described in his monograph on algebraic cryptanalysis [6]. We will use his description of KeeLoq and explain his attack.

The encryption process is shown in the Figure 4.4. The top rectangle is a 32-bit shift-register. It initially is filled with the plaintext. At each round, it is shifted one bit to the right, and a new bit is introduced. The computation of this new bit is the core of the cipher.

Five particular bits of the top shift-register are given as an input to non-linear function NLF. Meanwhile the key is placed initially in a 64-bit shift-register, which rotates cyclically to the right each iteration. The least significant bit of the key, the output of the non-linear function, and two particular bits of the 32-bit shift-register are XOR-ed together. This value becomes the new bit in the 32-bit shift-register. After 528 rounds, the contents of the 32 bit shift-register form the ciphertext.

The description of the cipher is quite simple. Lets denote by L_i the i-th bit of the top shift-register, by k_i the i-th bit of the key shift-register, by P_i the i-th bit of the plaintext and by C_i the i-th bit of the ciphertext. Using this variables, the description gives rise to the following system of equations:

$$
\begin{aligned}
i \in [0,31]: \quad & L_i &=& P_i, \\
i \in [32,559]: \quad & L_i &=& k_{i-32 \bmod 64} + L_{i-32} + L_{i-16} + \\
& & & + NLF(L_{i-1}, L_{i-6}, L_{i-12}, L_{i-23}, L_{i-31}), \\
i \in [528,559]: \quad & L_i &=& C_{i-528}.
\end{aligned}
$$

The input of NLF function is a 5-bit number $i \in [0,31]$ and the value of NLF on input i is defined as the i-th bit of hexadecimal number `3A5C742E`. It can be polynomially represented in the following way:

$$
\begin{aligned}
NLF(a,b,c,d,e) &= d + e + ac + ae + bc + be + cd + de + \\
&\quad + ade + ace + abd + abc
\end{aligned}
$$

The representation was obtained by the Karnaugh map. For more detail see [6].

Consider a plaintext-ciphertext pair $\mathcal{P}, \mathcal{C}$. There are 560 equations, one for each L_i, with $i \in [0, 559]$, plus another 32 for the C_i, with $i \in [0, 32]$. However, the first 32 of these are of the form $L_i = P_i$ for $i \in [0, 32]$, and the last 32 of these are of the form $Li = C_{i-528}$ for $i \in [528, 559]$. Thus we can use string substitution and drop down to 528 equations. This is precisely one equation for each round, which defines the new bit introduced into the shift register.

The 64 bits of the key are unknown. Also, of the 560 L_i, the first 32 and the last 32 are known, but the inner 496 are not. This yields 560 unknowns. If there are m plaintext-ciphertext message pairs, then there are $528m$ equations. However, there are only $496m + 64$ variables, because the key does not change from pair to pair.

To reduce the degree of the polynomials in our description, we introduce two new variables $\alpha = ab, \beta = ae$ and substitute them into our equations:

$$
\begin{aligned}
i \in [0,31]: \quad & L_i &=& P_i \\
i \in [528,559]: \quad & L_i &=& C_{i-528}
\end{aligned}
$$

$i \in [32, 559]$:

$$
\begin{aligned}
L_i &= k_{i-32 \bmod 64} + L_{i-32} + L_{i-16} + L_{i-23} + L_{i-31} + \\
&\quad + L_{i-1}L_{i-12} + \beta_i + L_{i-6}L_{i-12} + L_{i-6}L_{i-31} + L_{i-12}L_{i-23} + \\
&\quad + L_{i-23}L_{i-31} + \beta_i L_{i-23} + \beta_i L_{i-12} + \alpha_i L_{i-23} + \alpha_i L_{i-12} \\
\alpha_i &= L_{i-1}L_{i-6} \\
\beta_i &= L_{i-1}L_{i-31}
\end{aligned}
$$

As we can see, this introduced two new variables per original equation and two new equations as well. Thus m equations and n variables becomes $3m$ equations and $n + 2m$ variables. Thus with μ plaintext-ciphertext pair, we have 1548μ equations and $1552\mu + 64$ variables. It is obvious, that it must be the case when $\mu > 1$ for the system to be expected to have at most one solution. We know that

the system has at least one solution, because the message was sent. However for $\mu = 2$, this gives rise to 3168 equations over 3168 unknowns.

We implemented the system of equations using SAGE and solving it seems to be well beyond the capabilities of available Gröbner basis finding algorithms.

4.3.1 The Fixed-Point Attack

We have seen that the frontal attack fails. In his dissertation, Bard presented an attack based on decomposing the encryption into two functions and finding fixed points of the decomposition, allowing recovery of the secret key by solving a polynomial system of equations.

Recall that each 64th round uses the same key bit. This means that the same bit is used in rounds $t, t+64, t+128, \ldots$. There are 528 rounds, thus the key bits $k_0, \ldots k_{15}$ are used nine times, and the key bits $k_{16}, \ldots k_{63}$ eight times. We can use this observation to represent the encryption operation as:

$$E_k(\mathbf{P}) = g_k \circ \underbrace{f_k \circ f_k \circ \cdots \circ f_k}_{8 \text{ times}}(\mathbf{P}) = g_k(f_k^{(8)}(\mathbf{P})) = \mathbf{C},$$

where the f_k represents 64 rounds, the g_k represents the final 16 rounds and eight subsequent evaluations of the function f_k is denoted by $f_k^{(8)}$.

Now suppose that we gain access to the 16 key bits $k_0, \ldots, k_{15}$. We can actually guess those bits with probability 2^{-16}. Knowing the bits, we can evaluate g_k or its inverse g_k^{-1} and use it to transform the oracle for encryption function E_k to an oracle for $f_k^{(8)}$:

$$g_k^{-1}(E_k(\mathbf{P})) = g_k^{-1}(g_k(f_k^{(8)}(\mathbf{P}))) = f_k^{(8)}(\mathbf{P}).$$

We will employ the idea of fixed points now. Assume that we find x and y such that $f_k(x) = x$ and $f_k(y) = y$. How these are obtained will be explained later, but for now assume such points are known. At first, this seems strange to discuss at all. Because $f_k(x) = x$ and therefore $f_k^{(8)}(x) = x$, we know $E_k(x) = g_k(f_k^{(8)}(x)) = g_k(x)$. But, $g_k(x)$ is part of the cipher that we can remove by guessing a quarter of the key. Thus, if we "know something" about x we know something about multiple internal points, the input, and output of $E_k(x)$. Now we will make this idea more precise.

Intuitively, we now know 64 bits of input and 64 bits of output (32 bits from each message) of the functions f_k and $f_k^{(8)}$ as well. This forms a very rigid constraint, and it is highly likely that only one key could produce these outputs. This means that if we solve the system of equations for that key, we will get exactly one answer, which is the secret key. The only question is if the system of equations is rapidly solvable or not.

The resulting system of equations must have equations for the 64 rounds of f. For both x and y, there are equations for $L_0, \ldots, L_{95}$ and additional 32 output equations, but the first 32 of these and last 32 of these are of form $L_i = x_i$ and $L_{i+64} = x_i$, and can be eliminated by substituting as shown in the previous section. Therefore, there are actually $96 + 32 - 32 - 32 = 64$ equations for both x and y,

that gives 128 total equations. This is actually the same system of equations as in the previous section with the only difference that $i \in [0, 95]$ not $i \in [0, 559]$, but otherwise the equations are unchanged.

The bits x_i and y_i are known. The only unknowns are the 64 bits of key and the 32 intermediate variables L_i for both x and y. This is in total 128 unknowns.

Lets count the number of equations. After transformation from cubic into quadratic, it becomes 384 equations and 384 unknowns. This is obviously much smaller than the 3168 equations and 3168 unknowns that we had before. In fact, when we tried to solve the arising equations with SAGE, we found a solution on average in 55 seconds. It should also be noted that we needed at least two fixed points for a successful attack. One fixed point does not narrow the key space sufficiently down. The attack is outlined in the Algorithm 4.1.

Algorithm 4.1 The Fixed Point Attack on KeeLoq

Input: Oracular access to $E_k(x)$ for some unknown randomly generated key k
Output: The secret key k with probability $2^{-17.92}$ or Fail

```
Guess the 16 bits k_0, ..., k_15 of the key.
f_k^(8) = g_k^-1(E_k(x))
P = ∅
for x = 0 to 2^32 − 1 do
  if f_k^(8)(x) = x then
    for all y ∈ P do
      Write equations assuming f_k(x) = x and f_k(y) = y.
      Try to find k', a solution to the equations.
      if E_k(x) = g_k'(x) and E_k(y) = g_k'(y) then
        return k'
    P = P ∪ {x}
return Fail
```

We usually search only some part η of the codebook. We will compare the attack to the brute force attack, i.e. trying all possible keys. Lets assume that we have to search the whole codebook for a pair of fixed points. It can be shown that f has two or more fixed points with probability $1-2/e$. We also have to successfully guess the 16 key bits. Thus the probability of success is $2^{-16}(1-2/e) \approx 2^{-17.92}$. A brute force attack which would itself have probability $2^{-17.92}$ of success would consist of guessing $2^{46.08}$ possible keys and then aborting, because $46.08+17.92 = 64$, the length of key. Therefore, the fixed point attack must be faster than $2^{46.08}$ encryptions of guesses, or $528 \times 2^{46.08} \approx 2^{55.124}$ rounds.

The fixed point attack requires $g^{-1}(E_k(\mathbf{P}))$, which will need additional 16 rounds. Even if we use the whole codebook of 2^{32} plaintexts, this comes to $(528+16)2^{32} \approx 2^{41.087}$ rounds, that is about $2^{14.04}$ times faster than brute force.

Since the first attack by Bard, KeeLoq was analysed by many researches and another successful attacks were published. Interesting might be for example the slide attack proposed by Courtois, Bard, and Wagner [19].

4.4 Conclusions

Algebraic cryptanalysis attracted a great attention soon after publication of its first attacks. It might seem as an ultimate new method for cryptanalysis of modern block ciphers with high algebraic structure. General algebraic attack needs only a few plaintext-ciphertext pairs and if we succeed in solving the polynomial system, we achieve full key recovery. It is a great improvement because the traditional method of cryptanalysis of block ciphers, the differential cryptanalysis, requires thousands of plaintexts for successful attack. Also, modern block ciphers were designed with the traditional methods in mind and for example AES is proven to be secure against differential cryptanalysis. However, we have seen that we can not just write down the system of equations for a real-life cipher and expect it to be solvable using the available Gröbner basis implementations. Even for such a simple cipher as KeeLoq, we had to use some nontrivial observations about its structure to perform a successful attack.

Algebraic cryptanalysis can be viewed not only as an alternative to traditional methods in cryptanalysis. Those two concepts can also be combined. Albrecht and Cid [4] proposed algebraic techniques to speed up key-recovery differential attacks against block ciphers and applied this against block cipher PRESENT-128. In differential cryptanalysis, given a differential characteristics covering r out of N rounds of a block cipher, the attacker usually guesses subkey bits to overcome last $l = N - r$ rounds. This tends to be impractical for growing number l of last rounds. In the new attack, Gröbner bases are employed to check consistency of the equation system for the last l rounds based on the guessed key bits and this actually allows to increase l from 2 to 4.

So far, many different methods have been considered for algebraic cryptanalysis, with however limited success in targeting modern block ciphers. Most of the recent algebraic attacks are at most comparable to the traditional attacks. Nonetheless, algebraic cryptanalysis might get cryptographer an insight into the structure of modern block ciphers and becomes established as an important tool in the cryptanalysis.

Bibliography

[1] Ackermann, P., and Kreuzer, M. Gröbner Basis Cryptosystems. *Appl. Algebra Engrg. Comm. Comput. 17* (2006), 173–194.

[2] Adams, W. W., and Loustaunau, P. *An Introduction to Gröbner Bases*, vol. 3 of *Graduate Studies in Mathematics*. Amer Mathematical Society, Providence, 1994.

[3] Albrecht, M. Algebraic attacks on the Courtois toy cipher. *Cryptologia 32* (2008), 220–276.

[4] Albrecht, M., and Cid, C. Algebraic Techniques in Differential Cryptanalysis. In *FSE* (2009), O. Dunkelman, Ed., vol. 5665 of *Lecture Notes in Computer Science*, Springer, pp. 193–208.

[5] Albrecht, M., and Perry, J. *F*4/5. *arXiv* (2010), arXiv:1006.4933v1.

[6] Bard, G. V. *Algebraic Cryptanalysis*. Springer-Verlag, New York, 2009.

[7] Barkee, B., Can, D. C., Ecks, J., Moriarty, T., and Ree, R. F. Why you cannot even hope to use Gröbner bases in public key cryptography: an open letter to a scientist who failed and a challenge to those who have not yet failed. *J. Symbolic Comput. 18* (1994), 497–501.

[8] Becker, T., and Weispfenning, V. *Gröbner Bases: A Computational Approach to Commutative Algebra*, vol. 141 of *Graduate Texts in Mathematics*. Springer-Verlag, London, UK, 1993.

[9] Biryukov, A., and Cannière, C. D. Block Ciphers and Systems of Quadratic Equations. In *FSE* (2003), T. Johansson, Ed., vol. 2887 of *Lecture Notes in Computer Science*, Springer, pp. 274–289.

[10] Björck, G., and Fröberg, R. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symbolic Comput. 12* (1991), 329–336.

[11] Brickenstein, M., and Dreyer, A. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *J. Symbolic Comput. 44* (2009), 1326–1345.

[12] Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

[13] Buchmann, J., Pyshkin, A., and Weinmann, R.-P. A Zero-Dimensional Gröbner Basis for AES-128. In *FSE* (2006), M. J. B. Robshaw, Ed., vol. 4047 of *Lecture Notes in Computer Science*, Springer, pp. 78–88.

[14] Buchmann, J., Pyshkin, A., and Weinmann, R.-P. Block Ciphers Sensitive to Gröbner Basis Attacks. In *CT-RSA* (2006), D. Pointcheval, Ed., vol. 3860 of *Lecture Notes in Computer Science*, Springer, pp. 313–331.

[15] Cid, C., Murphy, S., and Robshaw, M. *Algebraic Aspects of the Advanced Encryption Standard*. Springer-Verlag, New York, 2006.

[16] Cid, C., Murphy, S., and Robshaw, M. J. B. Small Scale Variants of the AES. In *FSE* (2005), H. Gilbert and H. Handschuh, Eds., vol. 3557 of *Lecture Notes in Computer Science*, Springer, pp. 145–162.

[17] Cid, C., and Weinmann, R.-P. Block Ciphers: Algebraic Cryptanalysis and Groebner Bases. In *Groebner Bases: Coding and Cryptography* (2009), M. Sala, T. Mora, L. Perret, S. Sakata, and C. Traverso, Eds., Springer, pp. 307–328.

[18] Collart, S., Kalkbrener, M., and Mall, D. Converting bases with the Gröbner Walk. *J. Symbolic Comput. 24* (1997), 465–469.

[19] Courtois, N., Bard, G. V., and Wagner, D. Algebraic and Slide Attacks on KeeLoq. In *FSE* (2008), K. Nyberg, Ed., vol. 5086 of *Lecture Notes in Computer Science*, Springer, pp. 97–115.

[20] Courtois, N., Klimov, A., Patarin, J., and Shamir, A. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *EUROCRYPT* (2000), B. Preneel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer, pp. 392–407.

[21] Cox, D. A., Little, J., and O'Shea, D. *Using Algebraic Geometry*, vol. 185 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2005.

[22] Cox, D. A., Little, J., and O'Shea, D. *Ideals, Varieties, and Algorithms*. Springer-Verlag, New York, 2007.

[23] Eisenbud, D. *Commutative Algebra with a View Toward Algebraic Geometry*, vol. 150 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.

[24] Faugère, J.-C. A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra 139* (1999), 61–88.

[25] Faugère, J.-C. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *ISSAC '02: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation* (New York, NY, USA, 2002), ACM, pp. 75–83.

[26] Faugère, J.-C., Gianni, P., Lazard, D., and Mora, T. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Comput. 16* (1993), 329–344.

[27] Faugère, J.-C., and Joux, A. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In *CRYPTO* (2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 44–60.

[28] Kreuzer, M. Algebraic Attacks Galore! *Groups - Complexity - Cryptology 1* (2009), 231–259.

[29] Mayr, E. W. Some complexity results for polynomial ideals. *Journal of Complexity 13* (1997), 303–325.

[30] Shannon, C. E. Communication theory of secrecy systems. *Bell Systems Technical Journal 28* (1949), 656–715.

[31] Shimoyama, T., and Kaneko, T. Quadratic Relation of S-box and its Application to the Linear Attack of Full Round DES. In *CRYPTO* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 200–211.

[32] Stegers, T. Faugère's $F5$ algorithm revisited. Master's thesis, Technische Universität Darmstadt, 2005.

[33] Stein, W. A., et al. *Sage Mathematics Software (Version 4.4.3)*. The Sage Group, 2010. `http://www.sagemath.org`.

[34] Winkler, F. *Polynomial Algorithms in Computer Algebra*. Springer-Verlag, New York, 1989.