Algorithmic techniques in algebraic cryptanalysis and their applications

Makarim, R.H.

4 Gröbner Bases Algorithms

Contents

The previous chapter described Gröbner basis, its properties and some of its applications. We will now explain how to compute a Gröbner basis of an ideal $I = \langle f_1, \ldots, f_m \rangle \subseteq \mathcal{R}$ from its initial basis $\{f_1, \ldots, f_m\}$.

This chapter serves as a survey on existing Gröbner bases algorithms. We will explain three different algorithms. The first Gröbner basis algorithm is the Buchberger algorithm. It was introduced by Bruno Buchberger in his PhD thesis [Buc65, Buc06]. We give its description in Section 4.1. Since then, various attempts to improve its performance have been proposed. One of the main directions in improving the Buchberger algorithm is by detecting useless computations in advance. Some of the strategies to avoid useless computations in Gröbner bases algorithm will be mentioned in the same section.

The description of the Buchberger algorithm will be followed by the F_4 algorithm [Fau99], which is considered as a state-of-the-art Gröbner basis algorithm. One of its novel features is its ability to efficiently perform many polynomial reductions together. This is accomplished by computing the (reduced-)row echelon form of the coefficient matrix corresponding to a set of polynomial equations. We will describe the algorithm in Section 4.2.

The last section shall describe the XL Algorithm [CKPS00]. Although it was originally proposed as a way to solve overdefined system of equations, it turns out that the XL Algorithm can be seen as a redundant variant of the F_4 algorithm [AFI+04]. In that respect, the XL algorithm falls into the category of Gröbner basis algorithms. The cryptographic relevance of XL encourages us to mention it in this thesis.

This chapter, however, is not intended to be a complete survey that covers all existing Gröbner bases algorithms in the literature. One particular family of algorithms that we do not discuss is *signature-based Gröbner bases algorithms*. The concept of *signature* for a Gröbner basis algorithm was formally introduced by Faugère in the F_5 algorithm [Fau02]. Since then, there are numerous proposals of signature-based algorithms [AP10, GGV10, GIW16, EP10, EP11, RS12]. We do not cover such family of algorithms because the subject of signatures in the Gröbner bases algorithms deserves dedicated attention. Interested readers are referred to [EF17] for a recent survey on signature-based Gröbner bases algorithms.

4.1 Buchberger Algorithm

Let $I = \langle f_1, \ldots, f_m \rangle \subseteq \mathcal{R}$ be an ideal generated by $F = \{f_1, \ldots, f_m\}$. Recall that the set $G = \{g_1, \ldots, g_t\} \subseteq I$ is a Gröbner basis of I if for all $f \in I$, there exists a polynomial $g_i \in G$ such that $\mathsf{LT}(g_i) \mid \mathsf{LT}(f)$. As an element of I, each g_i can be written as a polynomial combination of $f_1, \ldots, f_m$. Thus, Gröbner bases algorithms can be viewed as a systematic approach of finding G using appropriate polynomial combinations of elements in the basis of an ideal.

The main condition of F not being a Gröbner basis of I is the existence of a polynomial $h \in I$ such that $\mathsf{LT}(f_i) \nmid \mathsf{LT}(h)$ for any $i \in \{1, \ldots, m\}$. The simplest intuition to obtain candidates for such h is by taking two distinct polynomials in the basis, say f_i, f_j with $i \neq j$, and forming a suitable combination

$$ax^\alpha f_i - bx^\beta f_j \in I, \quad a, b \in \mathbb{F},\ \alpha, \beta \in \mathbb{Z}^n_{\geq 0}$$

such that a cancellation of the leading terms occurs, leaving only smaller terms. Following this approach, Buchberger introduced the notion of S-polynomial.

Definition 4.1. *Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$ where $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$. Let $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{Z}^n_{\geq 0}$ such that $\gamma_i = \max(\alpha_i, \beta_i)$. We say that the monomial x^γ is the **least common multiple** of x^α and x^β, denoted as $x^\gamma = \mathsf{LCM}(x^\alpha, x^\beta)$.*

Definition 4.2 (S-polynomials)**.** *Let $f, g \in \mathcal{R}$ be nonzero polynomials and let $u = \mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g))$. The **S-polynomial** of f and g is defined as*

$$\mathsf{Spoly}(f, g) = \frac{u}{\mathsf{LT}(f)} \cdot f - \frac{u}{\mathsf{LT}(g)} \cdot g.$$

Definition 4.3 (Critical Pair)**.** *For an S-polynomial $\mathsf{Spoly}(f, g)$, we call the pair $p = (f, g)$ a **critical pair**. The degree of a critical pair p is defined as $\deg(p) = \deg(\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g)))$.*

Remark 4.4. *Note that the S-polynomial $\mathsf{Spoly}(f, g)$ is defined in order to yield the desired cancellation of leading terms, i.e.*

$$\begin{aligned}
\mathsf{Spoly}(f, g) &= \frac{x^\gamma}{\mathsf{LT}(f)} \cdot f - \frac{x^\gamma}{\mathsf{LT}(g)} \cdot g \\
&= \frac{x^\gamma}{\mathsf{LT}(f)}(\mathsf{LT}(f) + \mathsf{Tail}(f)) - \frac{x^\gamma}{\mathsf{LT}(g)}(\mathsf{LT}(g) + \mathsf{Tail}(g)) \\
&= \frac{x^\gamma}{\mathsf{LT}(f)}\mathsf{Tail}(f) - \frac{x^\gamma}{\mathsf{LT}(g)}\mathsf{Tail}(g).
\end{aligned}$$

The following proposition shows that every cancellation of leading terms among polynomials with identical leading monomial is attributed to the cancellation that occur for S-polynomials.

Proposition 4.5. *Let $f_1, \ldots, f_m \in \mathcal{R}$ and $\mathsf{LM}(f_i) = x^\alpha$ for all $i = 1, \ldots, m$. Suppose we have a polynomial $h = \sum_{i=1}^m f_i$. If $\mathsf{LM}(h) < x^\alpha$, then h is a linear combination, with coefficients in $\mathbb{F}$, of the S-polynomials $\mathsf{Spoly}(f_j, f_k)$ for $1 \leq j, k \leq m$. Furthermore, we have $\mathsf{LM}(\mathsf{Spoly}(f_j, f_k)) < x^\alpha$ for each j, k.*

Proof. See Lemma 5 in Section §2.6 of [CLO15, pg. 85] ■

The notion of S-polynomials together with multivariate division and the proposition above lead to an important and easily verifiable characterization of Gröbner bases. This characterization is also a foundation for the formulation of the Buchberger algorithm.

Theorem 4.6 (Buchberger's Criterion)**.** *Let I be an ideal of the polynomial ring $\mathcal{R}$. A basis $G = \{g_1, \ldots, g_t\}$ of I is a Gröbner basis of I if and only if for all pairs $g_i \neq g_j$, the remainder after division of $\mathsf{Spoly}(g_i, g_j)$ by G (listed in some order) is zero.*

Proof. See Theorem 6 in Section §2.6 of [CLO15, pg. 86]. ■

Following Buchberger's Criterion, the question now is what can be done with the nonzero remainders resulting from S-polynomials divided by a (non-Gröbner) basis of an ideal. Recall that a remainder from multivariate division algorithm consists of terms, including the leading term, that are not divisible by any leading term of polynomials in the divisor. If I is a polynomial ideal and $g_i, g_j \in G \subseteq I$, the remainder after division of $\mathsf{Spoly}(g_i, g_j)$ by G is also an element of I. These remainders are then considered as polynomials that provide more informations about the ideal I. Thus, expanding the existing basis of an ideal by adding these remainders is the most obvious method to reach a new basis that eventually satisfies Buchberger's criterion.

Theorem 4.7 (Buchberger's Algorithm)**.** *Let $I = \langle f_1, \ldots, f_m \rangle \neq \{0\}$ be an ideal of $\mathcal{R}$. Then a Gröbner basis for I can be constructed in a finite number of steps by Algorithm 4.1.*

Proof. We first show that $G \subseteq I$ holds in every iteration. This is true in the initial step. The cardinality of G is increased by adding the remainder after division of $\mathsf{Spoly}(h_1, h_2)$ by G' with $h_1, h_2 \in G' \subseteq G$. We have $G \subseteq I$, so $h_1, h_2 \in I$. This implies that $\mathsf{Spoly}(h_1, h_2) \in I$, and since we are dividing by G', the remainder r is also in I. Hence $G \cup \{r\} \subseteq I$.

The algorithm stops when $G = G'$, that is for all $h_1, h_2 \in G$, the remainder after division of $\mathsf{Spoly}(h_1, h_2)$ by G' is equal to 0. By Theorem 4.6, the set G is a Gröbner basis of I.

For a proof of its termination, we refer to the proof of Theorem 2 in Section §2.7 of [CLO15, pg. 91]. ■

Remark 4.8. *We refer to the set G in Algorithm 4.1 as the intermediate basis of $I = \langle f_1, \ldots, f_m \rangle$.*

4.1.1 Improvements with Gebauer-Möller Installation

The Buchberger's criterion and the Buchberger's algorithm lay the foundation for the development of other Gröbner bases algorithms. We will now discuss some improvements in line with the strategy to avoid unnecessary reductions.

By "unnecessary reductions", we mean reductions that yield the zero polynomial, since these do not provide useful new information about the ideal. The approach to avoid reductions to zero occurring in the main loop of Buchberger's algorithm is by detecting them in advance. Some criteria that identify some zero-reductions beforehand are discussed below.

Input: $F = (f_1, \ldots, f_m) \subseteq \mathcal{R}$
Result: A Gröbner basis G for $I = \langle f_1, \ldots, f_m \rangle$

```
1 G ← F
2 repeat
3 |  G' ← G
4 |  forall (h1, h2) ∈ G' × G' and h1 ≠ h2 do
5 |  |  r ← FullReduce(Spoly(h1, h2), G')
6 |  |  if r ≠ 0 then
7 |  |  |  G ← G ∪ {r}
8 until G = G';
9 return G
```

Algorithm 4.1: Buchberger's Algorithm

Definition 4.9. *Let $G = \{g_1, \ldots, g_t\}$ be a finite subset of $\mathcal{R}$. We say that a polynomial $f \in \mathcal{R}$ reduces to r modulo G, written $f \underset{G}{\longrightarrow} r$, if f can be written in the form*

$$f = q_1 g_1 + \ldots + q_t g_t + r \qquad q_i, r \in R \tag{4.1}$$

such that whenever $q_i g_i \neq 0$, we have

$$\mathsf{LM}(q_i g_i) \leq \mathsf{LM}(f)$$

and r is either 0 or r is not reducible by G.

Remark. *Definition 4.9 is defined independently from the existence of an algorithm, such as Algorithm 3.1, that calculates $q_1, \ldots, q_t, r$ given f and G. That is, if r is the remainder after the division of f by G, then $f \underset{G}{\longrightarrow} r$. However, the converse is not true in general as the remainder after a division does not have to be unique.*

Definition 4.9 allows us to state a more general version of Buchberger's criterion below.

Theorem 4.10. *A basis G for an ideal I is a Gröbner basis of I if and only if $\mathsf{Spoly}(g_i, g_j) \underset{G}{\longrightarrow} 0$ for all $i \neq j$.*

Theorem 4.11 (Buchberger's First Criterion)**.** *Let G be a finite subset of the ring $\mathcal{R}$. Let $f, g \in G$ with $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g)) = \mathsf{LM}(f) \cdot \mathsf{LM}(g)$. Then*

$$\mathsf{Spoly}(f, g) \underset{G}{\longrightarrow} 0.$$

Proof. See Proposition 4 in Section §2.9 of [CLO15, pg. 106]. ■

Definition 4.12 (Standard Representation)**.** *Let $f \in \mathcal{R}$ and $G = \{g_1, \ldots, g_k\}$ be a finite subset of $\mathcal{R}$. A representation*

$$f = \sum_{i=1}^{k} \mathsf{t}_i \cdot g_i$$

with terms $\mathsf{t}_i \in \mathsf{T}(\mathcal{R})$, $g_i \in G$ pairwise different is called a standard representation if $\max\{\mathsf{LM}(\mathsf{t}_i g_i) : 1 \leq i \leq k\} \leq \mathsf{LM}(f)$.

Theorem 4.13 (Buchberger's Second Criterion)**.** *Let* $G = \{g_1, \ldots, g_k\}$ *be a finite subset of* $\mathcal{R}$ *and* $f, h_1, h_2 \in \mathcal{R}$*. If*

1. $\mathsf{LM}(f) \mid \mathsf{LCM}(\mathsf{LM}(h_1), \mathsf{LM}(h_2))$ *and*
2. $\mathsf{Spoly}(f, h_1)$ *and* $\mathsf{Spoly}(f, h_2)$ *have a standard representation w.r.t.* G*.*

then, the S-polynomial $\mathsf{Spoly}(h_1, h_2)$ *has a standard representation w.r.t.* G*.*

Proof. See Proposition 5.70 in Section §5.5 of [BW93, pg. 223]. ■

There are at least two ways to incorporate both the Buchberger's first and the second criterion inside the Buchberger's algorithm. The first one is done by tracking which critical pairs have already been selected from the list during an execution of the main loop. Following the first criterion of Buchberger, whenever two polynomials g_1, g_2 are found in the intermediate basis G such that $\mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2)) = \mathsf{LM}(g_1)\mathsf{LM}(g_2)$ the algorithm marks (g_1, g_2) as having been treated and such pair is not added into the set of critical pairs. Now suppose that a critical pair (g_1, g_2) is selected. The algorithm tries to find a polynomial $g \in G$ such that $\mathsf{LM}(g) \mid \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$ and then it marks two pairs $(g, g_1), (g, g_2)$ as having been treated. If such event occurred, then nothing is done about (g_1, g_2). Otherwise, the pair (g_1, g_2) is treated similarly as in the Buchberger's algorithm (see Algorithm 4.1). The algorithm GRÖBNERNEW1 in [BW93, pg. 226] gives a more detail description of the above variant of Buchberger's algorithm.

```
   Input: Polynomials f_1, ..., f_m ∈ R
   Input: A selection function SELECT
   Result: A Gröbner basis G for the ideal ⟨f_1, ..., f_m⟩
1  for i ← 1 to m do
2      (G, P) ← UPDATE(G, P, f_i)
3  while P ≠ {} do
4      (p_1, p_2) ← SELECT(P)
5      P ← P \ {(p_1, p_2)}
6      f ← FULLREDUCE(Spoly(p_1, p_2), G)
7      if f ≠ 0 then
8          (G, P) ← UPDATE(G, P, f)
9  return G
```

Algorithm 4.2: Improved Buchberger's algorithm with Gebauer-Möller installation.

The second way of integrating Buchberger's first and second criterion is given in Algorithm 4.2. The actual implementation of both criteria is done by the UPDATE routine. Unlike the previous variant that waits until a critical pair is selected before making decision to eliminate it on the basis of the second criterion, Algorithm 4.2 tries to achieve this as early as possible. In addition to that, it also eliminates redundant elements in the intermediate basis G. Removing such redundant elements also reduces the number of critical pairs that needs to be processed.

The UPDATE function takes the intermediate basis G, the set of critical pairs P and a polynomial f as inputs. It incorporates Buchberger's first and second criterion together with elimination of redundant polynomials in G by employing four (4) **while**-loops. The first and second loops deal with the new critical pairs constructed from the input f and other elements of the intermediate basis G. The third and the fourth loop process elements in the old critical pair P and remove redundant elements in G respectively.

Definition 4.14. *Let $g_1, f, g_2 \in \mathcal{R}$ be polynomials that satisfy the equivalent conditions*

1. $\mathsf{LM}(f) \mid \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$,
2. $\mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(f)) \mid \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$ *and* $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_2)) \mid \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$

*then we call (g_1, f, g_2) a **Buchberger triple**.*

The first **while**-loop in the UPDATE function checks for each new critical pair (f, g_1) and tries to find another critical pair (f, g_2) on the existing list such that $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_2)) \mid \mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_1))$. The occurrence of such event causes elimination of the pair (f, g_1) due to (f, g_2, g_1) being a Buchberger triple. Furthermore, the pair (g_1, g_2) does not exist in this list at all. Notice that this loop keeps the pairs (f, g_1) with disjoint leading monomial. The rationale behind keeping these pairs is as follows :

> If two or more pairs in C have the same LCM of the leading monomials, so that there is a choice as to which one(s) should be deleted, then it is advantageous to try and keep one where the leading monomials are disjoint; that way, one eventually gets rid of all of them. [BW93, pg. 231]

The task of eliminating those new pairs with disjoint leading term is accomplished by the second **while**-loop, which is based on the Buchberger first criterion.

A more general version of Buchberger's criterion in Theorem 4.10 together with Buchberger second criterion tell that whenever a Buchberger triple (g_1, f, g_2) occurs in a Buchberger algorithm and both pairs $(g_1, f), (f, g_2)$ are already treated, then the critical pair (g_1, g_2) can be discarded from the computation. This is achieved by the third **while**-loop. Note that if two out of three LCM of leading monomials involved are equal, for instance

$$\mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(f)) = \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2)),$$

then both (g_1, f, g_2) and (f, g_2, g_1) are Buchberger triples. Thus, either (g_1, g_2) or (f, g_1) can be eliminated. However, there is a risk of erroneously eliminating both pairs (the first one on account of f and the latter one on account of g_2). This loop avoids it by removing only those triples (g_1, f, g_2) such that both $\mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(f))$ and $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_2))$ properly divide $\mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$.

After the updated new and old set of critical pairs are combined in P', the UPDATE function finally removes the redundant elements in G. Whenever a new element f in the ideal was found during the execution of the algorithm,

Input: An intermediate basis G
Input: A set of critical pairs P
Input: A nonzero polynomial $f \in \mathcal{R}$
Result: An updated intermediate basis G'
Result: An updated set of critical pairs P'

$C \leftarrow \{(f, g) : g \in G\}$
$D \leftarrow \{\}$
while $C \neq \{\}$ **do**
  Select (f, g_1) from C
  $C \leftarrow C \setminus \{(f, g_1)\}$
  $\mathfrak{m} \leftarrow \mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_1))$
  if $\mathfrak{m} = \mathsf{LM}(f) \cdot \mathsf{LM}(g_1)$ **or**
  (
   $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_2)) \not| \, \mathfrak{m} \quad \forall (f, g_2) \in C$
   and
   $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g_2)) \not| \, \mathfrak{m} \quad \forall (f, g_2) \in D$
)
   then
    $D \leftarrow D \cup \{(f, g_1)\}$
$E \leftarrow \{\}$
while $D \neq \{\}$ **do**
  Select (f, g) from D
  $D \leftarrow D \setminus \{(f, g)\}$
  if $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g)) \neq \mathsf{LM}(f)\mathsf{LM}(g)$ **then**
    $E \leftarrow E \cup \{(f, g)\}$
$P' \leftarrow \{\}$
while $P \neq \{\}$ **do**
  Select (p_1, p_2) from P
  $P \leftarrow P \setminus \{(p_1, p_2)\}$
  $\mathfrak{m} \leftarrow \mathsf{LCM}(\mathsf{LM}(p_1), \mathsf{LM}(p_2))$
  if $\mathsf{LM}(f) \not| \, \mathfrak{m}$ **or** $\mathsf{LCM}(\mathsf{LM}(p_1), \mathsf{LM}(f)) = \mathfrak{m}$ **or**
   $\mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(p_2)) = \mathfrak{m}$ **then**
    $P' \leftarrow P' \cup \{(p_1, p_2)\}$
$P' \leftarrow P' \cup E$
$G' \leftarrow \{\}$
while $G \neq \{\}$ **do**
  Select g from G
  $G \leftarrow G \setminus \{g\}$
  if $\mathsf{LM}(f) \not| \, \mathsf{LM}(g)$ **then**
    $G' \leftarrow G' \cup \{g\}$
$G' \leftarrow G' \cup \{f\}$
return G', P'

Algorithm 4.3: Gebauer-Möller Installation [BW93, GM88].

all the polynomials $g \in G$ such that $\mathsf{LM}(f) \mid \mathsf{LM}(g)$ are eliminated from G. One can justify this approach with the two arguments. First, if $\mathsf{LM}(f) \mid \mathsf{LM}(g)$ this implies that $\mathsf{LM}(f) \mid \mathsf{LCM}(\mathsf{LM}(g), \mathsf{LM}(h))$ for any $h \in \mathbb{F}[x_1, \ldots, x_n]$. Thus (g, f, h) is a Buchberger triple. The second argument is, by removing g, at the end the set G is still a Gröbner basis. In fact after the execution of the algorithm is finished, normalizing all $g \in G$ such that $\mathsf{LC}(g) = 1$ will make G a minimal Gröbner basis. However, later we will describe the drawback of this approach and we will also propose a better strategy to optimally maintain G.

The above mechanism of incorporating both Buchberger first and second criterion is due to Gebauer and Möller [GM88]. It is commonly referred as Gebauer-Möller installation. The pseudo-code is available in Algorithm 4.3.

4.2 Faugère's F_4 Algorithm

In this section another Gröbner basis algorithm called the F_4 algorithm[Fau99] is described. Generally the F_4 algorithm follows the same principles as the Buchberger algorithm, i.e., iteratively extending the existing intermediate basis by nonzero remainders after the division of S-polynomials with the current intermediate basis. Though a notable difference here is the F_4 algorithm performs the reduction of multiple S-polynomials together. This is achieved by mapping a finite set of polynomials to its corresponding coefficient matrix over $\mathbb{F}$, followed by the computation of the (reduced) row echelon form of that matrix.

Before going to the actual description of the F_4 algorithm, we first discuss the relation between polynomial reduction and linear algebra. We will define the notion of the coefficient matrix of a finite set of polynomials. This will be followed by a short theorem that shows the usefulness of row echelon reduced matrices for Gröbner bases computation. After that, we will continue by describing the basic version of the F_4 algorithm. However, this basic version generally has poor performance in practice. Thus, in the same paper Faugère also provided an enhanced version of the F_4 algorithm with better performance in practice compared to its basic version. The enhanced version is discussed in the last part of this section.

4.2.1 Linear Algebra and Gaussian Elimination of Polynomials

Let $\mathcal{M}$ be an $m \times r$ matrix defined over $\mathbb{F}$. We denote by $\mathcal{M}_{i,j} \in \mathbb{F}$ the element in row $i \in \{1, \ldots, m\}$ and column $j \in \{1, \ldots, r\}$ of $\mathcal{M}$. The notation $\mathsf{Row}(\mathcal{M}, i) \in \mathbb{F}^r$ is used to denote the i-th row of $\mathcal{M}$.

Let $M = (\mathfrak{m}_1, \ldots, \mathfrak{m}_r)$ be an ordered set of monomials in $\mathcal{R}$ and let $V_M = \{\sum_{i=1}^{r} c_i \mathfrak{m}_i : c_i \in \mathbb{F}, 1 \leq i \leq r\}$ be a vector subspace of $\mathcal{R}$ generated by M. We consider the linear map

$$\phi_M : V_M \mapsto \mathbb{F}^r \tag{4.2}$$

where $\phi_M(\mathfrak{m}_i) = e_i$ and $e_1, \ldots, e_r$ are the elements of the canonical basis of $\mathbb{F}^r$. The map ϕ_M allows an interpretation of polynomials in V_M as vectors in $\mathbb{F}^r$. On the other hand, the inverse function ϕ_M^{-1} gives an interpretation of vectors in $\mathbb{F}^r$ as polynomials in $V_M \subseteq \mathcal{R}$. We will now extend ϕ_M for a set of polynomials.

Definition 4.15. *Let $F = \{f_1, \ldots, f_m\} \subseteq \mathcal{R}$ and $M = \mathsf{M}(F)$ be the ordered set of monomials of F in descending order w.r.t. the monomial ordering $<$ in $\mathcal{R}$.*

*The **coefficient matrix** of F is the $m \times r$ matrix $\mathcal{M}$ constructed by assigning*

$$\mathsf{Row}(\mathcal{M}, i) \leftarrow \phi_M(f_i), \quad i \in \{1, \ldots, m\},$$

where ϕ_M is the linear function defined in (4.2).

Conversely, given the matrix $\mathcal{M}$, we can reconstruct the set of polynomials F below

$$F = \{\phi_M^{-1}(\mathsf{Row}(\mathcal{M}, i)) : 1 \leq i \leq r\} \setminus \{0\}.$$

The intuition behind the coefficient matrix $\mathcal{M}$ of $F \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is that the element $\mathcal{M}_{i,j}$ at row i and column j is the coefficient of the monomial $\mathfrak{m}_j$ in f_i. Naturally, when no such monomial exists in f_i, then $\mathcal{M}_{i,j}$ is equal to zero. Equivalently, the rows and columns of $\mathcal{M}$ represent the polynomials and monomials of F respectively, as illustrated below:

$$\mathcal{M} = \begin{array}{c} \\ f_1 \\ f_2 \\ \vdots \\ f_m \end{array} \begin{array}{c} \begin{array}{cccc} \mathfrak{m}_1 & \mathfrak{m}_2 & \ldots & \mathfrak{m}_r \end{array} \\ \begin{bmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \cdots & \vdots \\ * & * & \cdots & * \end{bmatrix} \end{array}$$

This implies that row operations in $\mathcal{M}$, such as addition and scalar multiplication, are essentially polynomial operations. In practice one can take advantage of efficient implementations of matrix algorithms in order to perform computations on polynomials. The most important matrix operation used in Gröbner bases computations is the computation of (reduced) row echelon form (also known as *Gaussian elimination*).

Definition 4.16. *Let $F, |F| = m$ be a finite subset of $\mathcal{R}$. Let $\mathcal{M}$ be the coefficient matrix of F and $\widetilde{\mathcal{M}}$ be the (reduced) row echelon form of $\mathcal{M}$. We say that the set*

$$\widetilde{F} = \{\phi_M^{-1}(\mathsf{Row}(\widetilde{\mathcal{M}}, i)) : 1 \leq i \leq m\} \setminus \{0\}$$

is the (reduced) row echelon form of F where $M = \mathsf{M}(F)$ is the ordered set of monomials of F.

Definition 4.16 motivates the definition of a Gaussian elimination algorithm for a finite set of polynomials F: Algorithm 4.4. The algorithm essentially computes the (reduced) row echelon form of F by applying Gaussian elimination on the coefficient matrix corresponding to F.

An important property of row echelon matrices is described in the following theorem.

Theorem 4.17. *[Fau99, pg. 65] Let F be a finite subset of $\mathbb{F}[x_1, \ldots, x_n]$ and let $\widetilde{F}$ be the (reduced) row echelon form of F. Then F^+ is defined as the following set:*

$$\widetilde{F}^+ = \{g \in \widetilde{F} : \mathsf{LM}(g) \notin \mathsf{LM}(F)\}.$$

For any subset $H \subseteq F$ such that $|H| = |\mathsf{LM}(F)|$ and $\mathsf{LM}(H) = \mathsf{LM}(F)$, the set $G = \widetilde{F}^+ \cup H$ is a triangular basis of the $\mathbb{F}$-vector space V generated by F. That is, for all $f \in V$, there exist $c_i \in \mathbb{F}$ and $g_i \in G$ such that $f = \sum_i c_i g_i$ with $\mathsf{LM}(g_1) = \mathsf{LM}(f)$ and $\mathsf{LM}(g_i) > \mathsf{LM}(g_{i+1})$.

Input: $F = \{f_1, \ldots, f_{|F|}\}$ - a finite subset of $\mathcal{R}$
Input: $\widetilde{F}$ - a (reduced) row echelon form of F
$M \leftarrow \mathsf{M}(F)$; // ordered in descending order
$\mathcal{M} \leftarrow |F| \times |M|$ zero matrix
for $i = 1$ **to** $|F|$ **do**
  $\mathsf{Row}(\mathcal{M}, i) \leftarrow \phi_M(f_i)$
$\widetilde{\mathcal{M}} \leftarrow$ the (reduced) row echelon form of $\mathcal{M}$
$\widetilde{F} \leftarrow \{\phi_M^{-1}(\mathsf{Row}(\widetilde{\mathcal{M}}, i)) : 1 \leq i \leq |F|\} \setminus \{0\}$
return $\widetilde{F}$

Algorithm 4.4: GaussianElimination algorithm.

4.2.2 Basic F_4 Algorithm

Before giving the actual description of the F_4 algorithm, we first need the following definition.

Definition 4.18. *For a given critical pair $p = (f, g) \in \mathcal{R}^2$, we define the following functions*

$$\textsc{Left}(p) = (x^\gamma / \mathsf{LM}(f), f), \tag{4.3}$$
$$\textsc{Right}(p) = (x^\gamma / \mathsf{LM}(g), g) \tag{4.4}$$

where $x^\gamma = \mathsf{LCM}(\mathsf{LM}(f), \mathsf{LM}(g))$. This definition is extended for a set of critical pairs $P \subset \mathcal{R}^2$, that is:

$$\textsc{Left}(P) = \bigcup_{p \in P} \{\textsc{Left}(p)\},$$
$$\textsc{Right}(P) = \bigcup_{p \in P} \{\textsc{Right}(p)\}.$$

The equation (4.3) and (4.4) in Definition 4.18 above are seen as the "left" part and the "right" part of S-polynomial $\mathsf{Spoly}(f, g)$. Note that the inversion of the leading coefficients is omitted, since this will be taken care by the computation of the row echelon form of $\textsc{Left}(P) \cup \textsc{Right}(P)$.

Similar with Buchberger algorithm, the F_4 algorithm takes as input a finite set of polynomials in $\mathcal{R}$ together with a selection function Select. Unlike in the Buchberger algorithm where the selection function only chooses one pair at a time, the feature of the F_4 algorithm to simultaneously reduce many S-polynomials at once requires the selection function to choose a subset of critical pairs. The description of the F_4 algorithm does not strictly give a concrete implementation of selection function. However, it is recommended to use normal selection strategy.

Definition 4.19. *Let $P \subset \mathcal{R}^2$ be a set of critical pairs and let $d_{\min} = \min\{\deg(p) : p \in P\}$ be the minimal degree of critical pairs in P. The **normal selection strategy** chooses a subset $P' \subseteq P$ such that*

$$P' = \{p \in P : \deg(p) = d_{\min}\}.$$

We now have the necessary information to describe the basic version of the F_4 algorithm. The pseudo-code of the basic F_4 algorithm is given in Algorithm 4.5. During the initialization step, the only notable difference compared to the Buchberger algorithm is that the F_4 algorithm introduces a loop counter d by which intermediate sets of the algorithm are indexed. The advantage of this counter d will be obvious in the improved version of the algorithm later. We will now describe the steps in the main iteration of the F_4 algorithm.

The first step in the main iteration of the F_4 algorithm is the selection of critical pairs. This is relatively straightforward given the selection function Select as input. This will be followed by the removal of selected pairs from the set of critical pairs P. Then the algorithm calls the functions Left as well as Right on the selected critical pairs and joins the two resulting sets into L_d. The elements of the set L_d are unevaluated products of the components of S-polynomials. Although this does not seem to provide an obvious advantage in the basic F_4 algorithm, the strategy that improves the performance of the F_4 algorithm later relies heavily on having these unevaluated products.

Input: A finite subset $F = \{f_1, \ldots, f_m\} \subseteq \mathcal{R}$
Input: A selection function Select
Output: A Gröbner basis of the ideal $\langle f_1, \ldots, f_m \rangle$

```
1  G ← F, F̃_0^+ ← F
2  d ← 0
3  P ← {(f_i, f_j) : 1 ≤ i < j ≤ m}
4  while P ≠ {} do
5      d ← d + 1
6      P_d ← Select(P)
7      P ← P \ P_d
8      L_d ← Left(P_d) ∪ Right(P_d)
9      F_d ← SymbolicPreprocessingBasic(L_d, G)
10     F̃_d ← GaussianElimination(F_d)
11     F̃_d^+ ← {f ∈ F̃_d : LM(f) ∉ LM(F_d)}
12     for h ∈ F̃_d^+ do
13         P ← P ∪ {{h, g} : g ∈ G}
14         G ← G ∪ {h}
15 return G
```

Algorithm 4.5: Basic version of F_4 Algorithm.

The next step in the main iteration of the F_4 algorithm is the reduction of S-polynomials in L_d. This is achieved by computing the row echelon form of a suitable set of polynomials, c.f. SymbolicPreprocessingBasic and GaussianElimination in line 9 and 10 respectively. The description of the sub-routine SymbolicPreprocessingBasic is given in Algorithm 4.6.

SymbolicPreprocessingBasic (cf. Algorithm 4.6) starts by evaluating the products $(\mathfrak{m} \cdot f)$ for all $(\mathfrak{m}, f) \in L$, which were previously constructed by calling the function Left and Right on the selected critical pairs. The goal of the algorithm is to add missing "reduction" polynomials $u \cdot g$ to F with $u \in \mathsf{M}(\mathcal{R}), g \in G$ and $\mathsf{LM}(u \cdot g) \in \mathsf{M}(F) \setminus \mathsf{LM}(F)$, until no such polynomials can be added anymore.

Input: A finite subset L of $\mathsf{M}(\mathcal{R}) \times \mathcal{R}$
Input: A finite subset G of $\mathcal{R}$
Output: A finite subset F of $\mathcal{R}$
1 $F \leftarrow \{\mathfrak{m} \cdot f : (\mathfrak{m}, f) \in L\}$
2 $D \leftarrow \mathsf{LM}(F)$
3 **while** $\mathsf{M}(F) \neq D$ **do**
4 Choose an $\mathfrak{m} \in \mathsf{M}(F) \setminus D$
5 $D \leftarrow D \cup \{\mathfrak{m}\}$
6 **if** $\exists g \in G : \mathsf{LM}(g) \mid \mathfrak{m}$ **then**
7 $F \leftarrow F \cup \{\mathfrak{m}/\mathsf{LM}(g) \cdot g\}$
8 **return** F

Algorithm 4.6: Algorithm SymbolicPreprocessingBasic.

The set D plays the role of tracking which monomials in $\mathsf{M}(F)$ have already been checked in the main loop of SymbolicPreprocessingBasic. It is initially set to $\mathsf{LM}(F)$, since initially for each $\mathfrak{m} \in \mathsf{LM}(F)$ there are already two distinct $f, f' \in F$ with $\mathsf{LM}(f) = \mathsf{LM}(f') = \mathfrak{m}$.

Lemma 4.20. *[Fau99, pg. 67] Let G be a finite subset of $\mathcal{R}$. Let F be the output of* SymbolicPreprocessingBasic(L_d, G) *and $\widetilde{F}$ be the reduced row echelon form of F. For all $h \in \widetilde{F}^+ = \{f \in \widetilde{F} : \mathsf{LM}(f) \notin \mathsf{LM}(F)\}$ we have $\mathsf{LM}(h) \notin \langle \mathsf{LM}(G) \rangle$.*

Proof. Assume that there exists $h \in \widetilde{F}^+$ such that $\mathsf{LM}(h) \in \langle \mathsf{LM}(G) \rangle$. This implies that there exists $g \in G$ such that $\mathsf{LM}(g) \mid \mathsf{LM}(h)$. Moreover, we have $\mathsf{LM}(h) \in \mathsf{M}(\widetilde{F}^+) \subset \mathsf{M}(\widetilde{F}) \subset \mathsf{M}(F)$ and h is lead-reducible by G. Hence $\mathsf{LM}(h)/\mathsf{LM}(g) \cdot g$ is inserted to F by SymbolicPreprocessingBasic which by definition of $\widetilde{F}^+$ contradicts the fact that $\mathsf{LM}(h) \notin \mathsf{LM}(F)$. ∎

Lemma 4.21. *Let G be a subset of $\mathcal{R}$ and let L be a finite subset of $\mathsf{M}(\mathcal{R}) \times \mathcal{R}$. We set $F =$* SymbolicPreprocessingBasic(L, G) *and $\widetilde{F}^+ = \{f \in \widetilde{F} : \mathsf{LM}(f) \notin \mathsf{LM}(F)\}$. For all polynomials f in the set $L' = \{\sum_i \mathfrak{c}_i \mathfrak{m}_i f_i : \mathfrak{c}_i \in \mathbb{F},\ (\mathfrak{m}_i, f_i) \in L\}$ we have*

$$f \xrightarrow[\widetilde{F}^+ \cup G]{} 0$$

Proof. Let $f \in L'$. By construction of F we have $L' \subseteq F$. Theorem 4.17 implies that f is an $\mathbb{F}$-linear combination of the triangular basis $\widetilde{F}^+ \cup H$ for a suitable $H \subseteq F$. Polynomials in H are either elements of L' or polynomials of the form $\mathfrak{m} \cdot g$ for $\mathfrak{m} \in \mathsf{M}(\mathcal{R})$ and $g \in G$. Thus f can be written in the form

$$f = \sum_i \mathfrak{c}_i f_i + \sum_j \mathfrak{c}_j \mathfrak{m}_j g_j, \quad \mathfrak{c}_i, \mathfrak{c}_j \in \mathbb{F}$$

and $f_i \in \widetilde{F}^+, \mathfrak{m}_j \in \mathsf{M}(\mathcal{R}), g_j \in G$. Thus, $f \xrightarrow[\widetilde{F}^+ \cup G]{} 0$. ∎

Theorem 4.22. *[Fau99, pg. 68] Given a finite subset $F \subseteq \mathcal{R}$, the F_4 algorithm returns a Gröbner basis G of an ideal $\langle F \rangle$, with $F \subseteq G$ and $\langle G \rangle = \langle F \rangle$.*

Proof. **Termination.** Assume that the **while**-loop does not terminate. There exists an ascending sequence $d_i > 0$ such that $\widetilde{F}_{d_i}^+ \neq \{\}$ for all i. Let $q_i \in \widetilde{F}_{d_i}^+$ and let $U_i = U_{i-1} + \langle \mathsf{LM}(q_i) \rangle$ with $U_0 = \{0\}$ for $i > 1$. It is trivial to show that U_i is an ideal of $\mathcal{R}$. Because polynomials in $\widetilde{F}_{d_i}^+$ are added to G at the end of every loop, by Lemma 4.20 we have $U_{i-1} \not\subseteq U_i$. This contradicts the fact that $\mathcal{R}$ is a Noetherian ring.

Correctness. The set G is constructed by $G = \bigcup_{d \geq 0} \widetilde{F}_d^+$. We claim that both statements below are loop-invariants of the **while**-loop.

1. The set $G \subset \mathcal{R}$ satisfies $F \subseteq G \subseteq \langle F \rangle$.
2. For all $g_1, g_2 \in G$ such that $(g_1, g_2) \notin P$, we have $\mathsf{Spoly}(g_1, g_2) \underset{G}{\longrightarrow} 0$.

We will now prove the first claim. Since $\widetilde{F}_0^+ = F$, we have $F \subseteq G$. Now suppose that F_d is the output of SymbolicPreprocessingBasic. Polynomials in F_d are of the form $\mathfrak{m} \cdot g$ with $\mathfrak{m}$ a monomial and $g \in G$. Since $\widetilde{F}_d$ consists of $\mathbb{F}$-linear combination of polynomials in F_d, so does $\widetilde{F}_d^+$. This shows that $G = \bigcup_{d \geq 0} \widetilde{F}_d^+ \subseteq \langle F \rangle$.

To prove the second claim, note that for $g_1, g_2 \in G$ such that $(g_1, g_2) \notin P$ this means that the critical pair (g_1, g_2) has been selected in a previous iteration, say iteration i, by the function Select. This implies that $(\mathfrak{m}_1, g_1), (\mathfrak{m}_2, g_2) \in L_i$ for some monomials $\mathfrak{m}_1, \mathfrak{m}_2$. Thus $\mathsf{Spoly}(g_1, g_2)$ is an element of the $\mathbb{F}$-vector space generated by the set $\{\mathfrak{m} \cdot f : \forall (\mathfrak{m}, f) \in L_i\}$. Hence by Lemma 4.21, we have $\mathsf{Spoly}(g_1, g_2) \underset{G}{\longrightarrow} 0$.

■

4.2.3 The Improved F_4 Algorithm

While the description of the basic F_4 algorithm computes a Gröbner basis for a given set of polynomials, the algorithm's efficiency can be significantly improved. There are two important techniques used to improve the basic F_4 algorithm.

One can check that the previous description of the F_4 algorithm keeps all critical pairs and redundant elements in the intermediate basis. The first improvement of the F_4 algorithm is the incorporation of the Buchberger's first and the second criterion. Faugère suggested the standard implementation of Gebauer-Möller installation [GM88] (see Algorithm 4.3). Instead of adding all critical pairs to P, the only pairs added are the ones that remain unaffected by the Buchberger's first and the second criterion implemented in the Update function.

The second important improvement to the F_4 algorithm is to reuse the computational results from previous iterations. In the basic version of F_4, the polynomials in $\widetilde{F}_d$ with leading monomials in $\mathsf{LM}(F_d)$ are simply discarded. These polynomials are the fully-reduced polynomials from $\tilde{F}_d$ and discarding them may lead to their recomputation in the next iterations. In the improved version of the F_4 algorithm, these polynomial are kept to replace some products $\mathfrak{m} \cdot f$ with a new product $\mathfrak{m}' \cdot f'$ with $\mathfrak{m}, \mathfrak{m}'$ be monomials of $\mathcal{R}$ such that $\mathfrak{m}' \leq \mathfrak{m}$ and $\mathsf{LM}(\mathfrak{m} \cdot f) = \mathsf{LM}(\mathfrak{m}' \cdot f')$. This means that more reductions are applied to f' already. This is achieved by the subroutine Simplify (Algorithm 4.9). The inputs to the Simplify function are a monomial $\mathfrak{m}$, a polynomial $f \in \mathcal{R}$, and a

Input: A finite subset $F = \{f_1, \ldots, f_m\} \subseteq \mathcal{R}$
Input: A selection function SELECT
Output: A Gröbner basis of the ideal $\langle f_1, \ldots, f_m \rangle$

```
1  G ← {}, P ← {}
2  d ← 0
3  while F ≠ {} do
4      Choose an f ∈ F
5      F ← F \ {f}
6      (G, P) ← UPDATE(G, P, f)
7  while P ≠ {} do
8      d ← d + 1
9      P_d ← SELECT(P)
10     P ← P \ P_d
11     L_d ← LEFT(P_d) ∪ RIGHT(P_d)
12     F_d ← SYMBOLICPREPROCESSING(L_d, G, (F_1, ..., F_{d-1}))
13     F̃_d ← GAUSSIANELIMINATION(F_d)
14     F̃_d^+ = {f ∈ F̃_d : LM(f) ∉ LM(F_d)}
15     for h ∈ F̃_d^+ do
16         (G, P) ← UPDATE(G, P, h)
17 return G
```

Algorithm 4.7: An improved F_4 algorithm.

$(d-1)$-tuple of all previously constructed F, i.e. $(F_1, \ldots, F_{d-1})$. The SIMPLIFY function is called while constructing F_d from L_d, that is during the execution of symbolic preprocessing. Some necessary changes are applied to the previously SYMBOLICPREPROCESSINGBASIC into the SYMBOLICPREPROCESSING described in Algorithm 4.8.

Lemma 4.23. *[Fau99] Let $(\mathfrak{m}, f) \in \mathsf{M}(\mathcal{R}) \times \mathcal{R}$. If $(\mathfrak{m}', f') \in \mathsf{M}(\mathcal{R}) \times \mathcal{R}$ is an output of* SIMPLIFY$(\mathfrak{m}, f, (F_1, \ldots, F_{d-1}))$*, then $\mathsf{LM}(\mathfrak{m}' \cdot f') = \mathsf{LM}(\mathfrak{m} \cdot f)$. Furthermore, if V is an $\mathbb{F}$-vector space generated by $\left(\bigcup_{i=1}^{d-1} F_i\right) \bigcup \left(\bigcup_{i=1}^{d-1} \widetilde{F_i}\right)$, then there exists a nonzero $\mathfrak{c} \in \mathbb{F}$ and $r \in V$ such that $\mathfrak{m}f = \mathfrak{c}\mathfrak{m}'f' + r$ with $\mathsf{LM}(r) < \mathsf{LM}(\mathfrak{m} \cdot f)$*

Proof. See Lemma 2.3 in [Fau99, pg. 71]. ■

Theorem 4.24. *[Fau99] Let F be a finite subset of $\mathcal{R}$ and let $\mathcal{F} = (F_1, \ldots, F_{d-1})$ with $F_i \subseteq \mathcal{R}$ for $i = 1, \ldots, d-1$. Let $(g_1, g_2) \subset \mathcal{R}^2$ be a critical pair. For $i = 1, 2$ we define $\mathfrak{m}_i = x^\gamma / \mathsf{LM}(g_i)$ with $x^\gamma = \mathsf{LCM}(\mathsf{LM}(g_1), \mathsf{LM}(g_2))$. If the following conditions hold*

1. *For $k = 1, \ldots, d-1$, we have $F_k = \{\mathfrak{m} \cdot f : \forall (\mathfrak{m}, f) \in L_k\}$ where L_k is a finite subset of $\mathsf{M}(\mathcal{R}) \times \mathcal{R}$,*
2. *For $k = 1, \ldots, d-1$, $\widetilde{F_k} \subset G$,*
3. *$f_i = \mathfrak{m}_i' \cdot g_i'$ where $(\mathfrak{m}_i', g_i') =$ SIMPLIFY$(\mathfrak{m}_i, g_i, \mathcal{F})$ for $i = 1, 2$,*
4. *$\mathsf{Spoly}(f_1, f_2) \underset{F}{\longrightarrow} 0$*

then we have $\mathsf{Spoly}(g_1, g_2) \underset{F}{\longrightarrow} 0$.

Proof. See Theorem 2.4 in [Fau99, pg. 71]. ■

Input: A finite subset L of $\mathsf{M}(\mathcal{R}) \times \mathcal{R}$
Input: A finite subset G of $\mathcal{R}$
Input: A tuple $(F_1, \ldots, F_{d-1})$ where $F_i \subseteq \mathcal{R}$
Output: A finite subset $\tilde{F}$ of $\mathcal{R}$

```
F ← {m' · f' : (m, f) ∈ L, (m', f') ← SIMPLIFY(m, f, (F_1, ..., F_{d-1}))}
D ← LM(F)
while M(F) ≠ D do
    Choose an m ∈ M(F) \ D
    D ← D ∪ {m}
    if ∃g ∈ G : LM(g) | m then
        (m', g') ← SIMPLIFY(m/LM(g), g, (F_1, ..., F_{d-1}))
        F ← F ∪ {m' · g'}
return F
```

Algorithm 4.8: Algorithm SymbolicPreprocessing

Input: A monomial $\mathfrak{m} \in \mathsf{M}(R)$
Input: A polynomial $f \in \mathcal{R}$
Input: A tuple $(F_1, \ldots, F_{d-1})$, $F_i \subseteq \mathcal{R}$, $1 \leq i \leq d-1$
Output: An element of $\mathsf{M}(\mathcal{R}) \times \mathcal{R}$

```
for u ∈ M(R) : u | m do
    if ∃j ∈ {1, ..., d − 1} : uf ∈ F_j then
        F̃_j ← row echelon form of F_j
        There exists a p ∈ F̃_j s.t. LM(p) = LM(uf)
        if u ≠ m then
            return SIMPLIFY(m/u, p, (F_1, ..., F_{d-1}))
        else
            return (1, p)
return (m, f)
```

Algorithm 4.9: Algorithm Simplify

4.3 Extended Linearization (XL) Algorithm

In this subsection we discuss the *extended linearization* (XL) algorithm to solve a system of multivariate polynomial equations. The algorithm was proposed by Courtois *et al.* [CKPS00] and it received a lot of attention in the cryptographic community due to its relatively straightforward description as well as its simplicity. The XL algorithm takes into account that the systems of equations appear in cryptanalysis of cryptographic schemes are, in general, overdefined. The authors of XL argued that these cases are not well-utilized by Gröbner bases

algorithms and they claimed that the XL algorithm is a major improvement to solve overdefined system of equations.

The theoretical foundation of the XL algorithm is build upon the linearization technique. Let us now briefly discuss the idea of linearization. Consider a random homogeneous quadratic polynomial equations F over a field $\mathbb{F}$ in variables $x_1, \ldots, x_n$ with $\binom{n}{2} = n(n-1)/2$ equations. The main idea of linearization is to transform F into a system of linear equations by treating each monomial $x_i x_j$ in F as a new independent variable, say, y_{ij}. The new linear system has $n(n-1)/2$ variables with $n(n-1)/2$ equations. This can be efficiently solved using Gaussian elimination. As soon as all solutions for y_{ij} were found, two possible values for x_i are extracted by computing the square root of y_{ii} in $\mathbb{F}$. The values y_{ij} are used to remove pathological solutions by combining the correct square roots of y_{ii} and y_{jj}.

The linearization works well when there are sufficient number of linearly independent equations. To increase the number of equations, Kipnis and Shamir introduced a new method called relinearization [KS99]. The main principle of relinearization is that variables y_{ij} in the linear equations are related rather than independent. For instance, a relation that can be derived is based on the commutativity of multiplication. Let $i, j, k, \ell \in \{1, \ldots, n\}$, we have the following relation

$$(x_i x_j)(x_k x_\ell) = (x_i x_k)(x_j x_\ell) = (x_i x_\ell)(x_j x_k).$$

This implies that $y_{ij} y_{k\ell} = y_{ik} y_{j\ell} = y_{i\ell} y_{jk}$ and the following non-linear quadratic equations can be added to the set of linear equations

$$\begin{aligned} y_{ij} y_{k\ell} - y_{ik} y_{j\ell} &= 0, \\ y_{ij} y_{k\ell} - y_{i\ell} y_{jk} &= 0, \\ y_{ik} y_{j\ell} - y_{i\ell} y_{jk} &= 0. \end{aligned}$$

Different variants of relinearization technique based on other relations, such as recursive and commutativity of multiplication of higher degree monomials, were also discussed in [KS99].

Let $F = \{f_1, \ldots, f_m\}$ be a set of quadratic polynomials in $x_1, \ldots, x_n$ over the field $\mathbb{F}$. Given a positive integer D, the XL algorithm adds polynomials of the form $\mathfrak{m} \cdot f_i$ to F, where $\mathfrak{m}$ is a monomial, such that $\deg(\mathfrak{m} \cdot f_i) \leq D$. The algorithm proceeds to solve the new equation system using the linearization technique. The complete description is given below:

1. **Multiply** : Generate all polynomials of the form $\mathfrak{m} \cdot f_i$ where $\mathfrak{m}$ is a monomial such that $\deg(\mathfrak{m} \cdot f_i) \leq D$ for all $i \in \{1, \ldots, m\}$.

2. **Linearize** : Consider an ordering on monomials such that the univariate monomials are the smallest monomials, i.e., these will be eliminated last. We treat each monomial as a new variable independent from each other.

3. **Reduction** : Compute the (reduced) row echelon form of the corresponding coefficient matrix of the linearized system of equations.

4. **Solve** : Assume the previous step yields a univariate polynomial h in the variable x_i. Find all the roots of h in $\mathbb{F}$, e.g., by polynomial factorization.

5. **Repeat** : Simplify the original system of equations by substituting x_i and repeat similar process to obtain solutions for other variables.

Based on experimental observation [CKPS00], the authors suggested the following value for D.

	D
$m = n$	2^n
$m = n + 1$	n
$m = n + 2$	$\sqrt{n} + C$ (C is a constant)

Table 4: *Suggested values for D in XL.*

The XL algorithm was intended to be an efficient algorithm to solve a system of equations F with n variables over a finite field $\mathbb{F}_q$ in which F has only one solution in $\mathbb{F}_q^n$. Note that F may have another solution in $\mathbb{F}_{q^e}^n$ where $\mathbb{F}_{q^e}$ is the e-th degree extension of $\mathbb{F}_q$. Elimination of solutions in $\mathbb{F}_{q^e}^n \setminus \mathbb{F}_q^n$ is done by inserting polynomials $\{x_i^q - x_i : i = 1, \ldots, n\}$ to F.

4.3.1 XL as a Variant of the F_4 Algorithm

In this subsection, we briefly discuss the relation between the XL algorithm and the F_4 algorithm where the former is seen as a redundant variant of the latter. The description of the XL algorithm as a variant of the F_4 relies on the following definitions.

Definition 4.25. *For a critical pair $p = (f, g) \in \mathcal{R}^2$ and a nonnegative integer $d \in \mathbb{Z}_{\geq 0}$ we define the following function*

$$\textsc{Left}_{\text{XL}}(p, d) = \{(\mathfrak{m}, f) : \mathfrak{m} \in \mathsf{M}(\mathcal{R}), \deg(\mathfrak{m} \cdot f) \leq d\},$$
$$\textsc{Right}_{\text{XL}}(p, d) = \{(\mathfrak{m}, g) : \mathfrak{m} \in \mathsf{M}(\mathcal{R}), \deg(\mathfrak{m} \cdot g) \leq d\}.$$

For a set of critical pairs $P \subseteq \mathcal{R} \times \mathcal{R}$, we also define a natural extension of above definitions as follows

$$\textsc{Left}_{\text{XL}}(P, d) = \bigcup_{p \in P} \textsc{Left}_{\text{XL}}(p, d),$$
$$\textsc{Right}_{\text{XL}}(P, d) = \bigcup_{p \in P} \textsc{Right}_{\text{XL}}(p, d).$$

Definition 4.26. *Let $P \subseteq \mathcal{R} \times \mathcal{R}$ be a set of critical pairn. For nonnegative integer $d \in \mathbb{Z}_{\geq 0}$ we define the following selection function for XL*

$$\textsc{Select}_{\text{XL}}(P, d) = \{p \in P : \deg(p) \leq d\}.$$

The description of the XL algorithm in an F_4-like fashion is given in Algorithm 4.10. Note that the algorithm described differs slightly from the original description of the XL algorithm. While the original algorithm sets the degree bound D globally at once, Algorithm 4.10 determines the optimal value for D iteratively. The **Multiply** step of the XL algorithm corresponds to the computation of L_d in line 9. The **Reduction** step of the XL algorithm corresponds to the Gaussian elimination in line 10. Algorithm 4.10 clearly shows that the XL algorithm is a redundant variant of the F_4 algorithm since the function $\textsc{Left}_{\text{XL}}$ and $\textsc{Right}_{\text{XL}}$ collect more polynomials than the function $\textsc{Left}$ and $\textsc{Right}$ of F_4. This implies that there are more polynomials constructed in F_d in the XL algorithm compared to the same set in the F_4 algorithm.

Input: A finite subset $F = \{f_1, \ldots, f_m\} \subseteq \mathcal{R}$
Output: A finite subset $G \subseteq \langle f_1, \ldots, f_m \rangle$

$G \leftarrow F$, $\tilde{F}_0^+ \leftarrow F$
$d \leftarrow 0$
$P \leftarrow \{(f_i, f_j) : 1 \leq i < j \leq m\}$
while $P \neq \{\}$ **do**
  $d \leftarrow d + 1$
  $P_d \leftarrow \textsc{Select}_{\text{XL}}(P, d)$
  $P \leftarrow P \setminus P_d$
  $L_d \leftarrow \textsc{Left}_{\text{XL}}(P_d, d) \cup \textsc{Right}_{\text{XL}}(P_d, d)$
  $F_d \leftarrow \{\mathfrak{m} \cdot f : (\mathfrak{m}, f) \in L_d\}$
  $\widetilde{F}_d \leftarrow \textsc{GaussianElimination}(F_d)$
  $\widetilde{F}_d^+ \leftarrow \{f \in \widetilde{F}_d : \mathsf{LM}(f) \notin \mathsf{LM}(F_d)\}$
  for $h \in \widetilde{F}_d^+$ **do**
    $P \leftarrow P \cup \{(h, g) : g \in G\}$
    $G \leftarrow G \cup \{h\}$
return G

Algorithm 4.10: An F_4-like description of XL algorithm.

Theorem 4.27. *Let $F \subseteq \mathcal{R}$. Then Algorithm 4.10 computes a Gröbner basis G of the ideal $\langle F \rangle \subseteq \mathcal{R}$ such that $F \subseteq G$.*

Proof. See Theorem 3 in [AFI+04]. ■

4.3.2 The Advantage of XL Algorithm

Although the XL algorithm can be viewed as a variant of the F_4 algorithm, its distinct advantage is that it produces a sparse coefficient matrix. This sparsity arises from the redundant polynomials added to the rows of the coefficient matrix as a result of the multiply step. Consequently, the XL algorithm can leverage dedicated sparse linear algebra techniques, especially Wiedemann [Wie86] and Block Wiedemann [Cop94], which are tailored for matrices over finite fields. This property makes XL particularly appealing in cryptanalytic applications involving sparse and overdefined systems of equations.