



UNIVERSITY OF TRENTO
DEPARTMENT OF MATHEMATICS
MASTER'S DEGREE IN MATHEMATICS

~ . ~

ACADEMIC YEAR 2024–2025

Algebraic Algorithms for the Learning With Errors Problem

Supervisor
Irene VILLA, PhD

Co-Supervisor
Prof. Arnab ROY

Candidate
Massimo FUMIANI
247723

FINAL EXAMINATION DATE: March 20, 2026

Acknowledgments

This thesis is the result of a long journey, and it would not have been possible without the people who accompanied me along the way.

I would like to begin by thanking Professor Roy, whose warm welcome in Innsbruck and invaluable advice made this experience truly extraordinary. I am truly grateful to Irene for her support and for following me not only during this thesis, but throughout my master's studies, including my internship.

A special thank you goes to Luca, who has been an irreplaceable companion throughout these months abroad. Thank you for your support, your enthusiasm, and for making the work so much more enjoyable.

On a more personal note, I want to thank Manuel and Giovanni for being a constant presence and a shoulder to lean on. To the Lonigo group, now scattered across different cities, and to the whole mathematics group, for making this path worthwhile. To “ST7” and “Renk4”, thank you for turning both Trento and Innsbruck into places that felt like home.

To my family, thank you for your unconditional support and for always being my safe harbour, no matter how far I went.

And to Costanza, you never stopped believing in me, even across the distance and through the hardest moments. Thank you for being there, always. I would not be here without you.

Abstract

The Learning With Errors (LWE) problem is central to modern post-quantum cryptography. While standard lattice-based attacks are heavily studied, algebraic cryptanalysis, specifically polynomial system solving approaches inspired by Arora and Ge, offers a powerful alternative, providing valuable insights into the underlying algebraic structure of this fundamental problem. This algebraic perspective is particularly relevant for the **BinaryError**-LWE problem and its **BinarySecret** variant, which are increasingly utilized in lightweight IoT cryptography and Fully Homomorphic Encryption to minimize resource overhead.

Traditional algebraic techniques, such as the XL algorithm and standard Gröbner basis computations (e.g., F4/F5), serve as the classical foundation for solving multivariate systems. However, a fundamental limitation of these methods is the difficulty in deriving precise and accurate complexity estimates. Because predicting the exact number of operations required to compute a general Gröbner basis is notoriously difficult, state-of-the-art estimates rely on heuristic assumptions, most notably the degree of regularity (d_{reg}) and semi-regular sequences.

While this unpredictability holds true for general polynomial systems, we demonstrate that it is possible to design an *ad hoc* process for the LWE problem to accurately bound the complexity. To this end, we propose a novel, exact algebraic attack specifically designed to break **BinaryError**-LWE instances. By systematically combining linear-algebraic matrix diagonalization with targeted S -polynomial reductions, our algorithm forces a deterministic degree-fall. Crucially, this mechanism entirely confines the computation to degree-two polynomials, completely eliminating unpredictable degree inflation and allowing for an exact theoretical estimation of the algorithm’s computational cost.

We provide a rigorous mathematical complexity analysis that definitively establishes a strictly polynomial asymptotic time bound of $\mathcal{O}(n^6)$, successfully circumventing the need for the semi-regularity assumption. Finally, we provide a complete MAGMA implementation that practically executes the algorithm and empirically verifies our theoretical complexity bounds.

Contents

Introduction	xi
1 Mathematical Foundations	1
1.1 Notation	1
1.2 Affine Varieties and Ideals	2
1.3 Gröbner Basis	3
1.3.1 Monomial Orderings	3
1.3.2 Gröbner Basis	4
1.3.3 Quotient Space and Zero-Dimensional Ideals	5
1.4 Algorithms for Computing Gröbner Basis	6
1.4.1 S-Polynomials and Buchberger’s Criteria	6
1.4.2 Buchberger’s Algorithm	7
1.4.3 Faugère’s F4 and F5 Algorithms	8
1.4.4 Linearization and the Macaulay Matrix	9
1.4.5 The XL Algorithm	11
1.5 Basics on LWE	12
1.5.1 Applications of LWE	14
1.5.2 LWE Variants	14
2 Cryptanalysis of LWE	17
2.1 Fundamental Reductions and Strategies	17
2.2 Lattice-Based Attacks	18
2.3 Combinatorial Attacks	19
2.4 Algebraic Attacks	19
2.4.1 General Modeling Approach	20
2.4.2 The Linearization Technique	20
2.4.3 The Polynomial System Solving Pipeline	21
2.4.4 Gröbner Basis and the Degree of Regularity	22
2.4.5 Refined Analysis of the Degree of Regularity	23
2.4.6 The Bottleneck of Existing Analyses	24

3	A Novel Algebraic Algorithm for LWE with Binary Errors	27
3.1	From Linear Algebra to Gröbner Basis	27
3.2	Overview of the Algorithm	29
3.3	The Algorithm	30
3.3.1	Phase 1: Initial System Diagonalization	32
3.3.2	Phase 2: S-Polynomial Generation and Reduction	33
3.3.3	Phase 3: Final Diagonalization and Secret Recovery	36
3.4	A Comprehensive Toy Example	36
3.5	Pseudocode	40
3.6	Algorithm Termination and Correctness	41
3.7	Complexity Analysis	43
3.7.1	Phase 1: Initial System Diagonalization	43
3.7.2	Phase 2: S-Polynomial Generation and Reduction	47
3.7.3	Phase 3: Final Diagonalization and Secret Recovery	49
3.7.4	Total Asymptotic Complexity	51
3.8	Adaptation for Binary Errors and Binary Secrets	53
3.8.1	New basis dimension	53
3.8.2	Selection of S -Polynomial Pairs	53
3.8.3	Complexity Re-Evaluation for Binary Secrets	55
4	Experimental Results	57
4.1	Methodology	57
4.2	Performance and Asymptotic Behavior	58
4.3	Matrix-Optimized Implementation	59
4.4	Theoretical vs. Practical Performance	60
4.5	Success Probability and Prime Modulus	61
5	Conclusions	63
5.1	Summary of Contributions	63
5.2	Further Work	63
A	Probability of Linear Independence for LWE Samples	65
A.1	Case 1: Binary Errors	65
A.2	Case 2: Binary Errors and Binary Secrets	66
A.3	Numerical Example: Practical Parameter Sizes	67
B	MAGMA implementation	69
B.1	Sample Generation	69
B.2	Polynomial System Reduction	70
B.3	Proposed Algorithm	71
B.4	Implementation and Code Availability	73
	Bibliography	81

List of Figures	84
List of Tables	85

Introduction

In recent years, research in post-quantum cryptography (PQC) has explored several promising directions to secure communication systems against quantum adversaries. Among these, lattice-based cryptography has attracted the most significant attention due to its strong worst-case to average-case reductions [19, 42, 45, 46] and its practical efficiency. Lattice-based schemes enable the construction of a remarkably versatile array of cryptographic primitives, ranging from foundational public-key encryption to advanced protocols like Fully Homomorphic Encryption (FHE).

Within this context, the Learning With Errors (LWE) problem, introduced by Oded Regev in 2005 [45], has emerged as the central building block of modern lattice-based cryptography. At a high level, LWE can be viewed as a noisy variant of solving linear systems. While a standard system of linear equations $\mathbf{Ax} = \mathbf{b}$ over a finite field can be efficiently solved using Gaussian elimination, injecting a small error term to each equation to yield $\mathbf{Ax} + \mathbf{e} = \mathbf{b}$ transforms the recovery of the secret vector $\mathbf{x}$ into a strictly hard problem for both classical and quantum algorithms.

Over the years, variants such as Ring-LWE and Module-LWE have been proposed to exploit algebraic structures for efficiency improvements while preserving these hardness properties, underpinning standard post-quantum schemes like CRYSTALS-Kyber [8]. Conversely, conservative schemes like FrodoKEM [14, 34] deliberately avoid these structured variants, relying entirely on the unstructured, plain LWE problem to maximize security guarantees.

Alongside the development of these schemes, the LWE problem has become the subject of extensive cryptanalysis. In this thesis, we focus specifically on the **search-LWE** problem from an algorithmic cryptanalysis perspective. While lattice reduction algorithms often dominate practical attack scenarios, algebraic methods offer crucial insights into the mathematical structure of the problem. These methods transform LWE into a system of non-linear polynomial equations, which are typically tackled using tools from computational algebra, notably Gröbner basis approaches.

While Gröbner basis provide a natural, deterministic framework for solving such polynomial systems, a fundamental challenge lies in accurately estimating the computational complexity. Predicting the exact number of operations required

to compute a Gröbner basis for a general system is notoriously difficult. To establish theoretical bounds, state-of-the-art complexity estimates rely heavily on heuristic assumptions, most notably the degree of regularity and the assumption that the system forms a semi-regular sequence. While these heuristics provide useful asymptotic approximations, they often yield loose complexity bounds, leaving significant room for improvement in theoretical estimations. This lack of exactness makes it difficult to rigorously assess the concrete security of specific LWE instantiations under general algebraic attacks.

The primary objective of this thesis is to bridge this gap by designing an exact algebraic solver tailored specifically for the LWE problem, thereby eliminating the reliance on general-purpose heuristics. In the context of lightweight IoT cryptography, the adoption of binary errors has been highly successful in bypassing computationally expensive Gaussian sampling, notably in structured variants like **BinarySecret**-RLWE [21]. However, the strictly unstructured, plain **BinaryError**-LWE case remains the fundamental theoretical benchmark for algebraic cryptanalysis [7].

Rather than applying standard Gröbner basis algorithms, we propose a novel, *ad hoc* algebraic attack designed specifically to recover the secret vector from **BinaryError**-LWE samples. By systematically combining linear-algebraic matrix diagonalization with targeted S -polynomial reductions, our algorithm forces a deterministic degree-fall, strictly confining the computation to degree-two polynomials. Crucially, this approach allows us to provide a rigorous, direct mathematical complexity analysis, establishing a strictly polynomial asymptotic time bound of $\mathcal{O}(n^6)$ without invoking the semi-regularity assumption or relying on heuristic estimates of the degree of regularity.

To achieve this, the remainder of the thesis is structured as follows. Chapter 1 establishes the mathematical background necessary for the study, introducing foundational concepts from computational algebra, polynomial rings, ideal theory, and Gröbner basis. Chapter 2 provides a comprehensive overview of the LWE problem and its cryptanalysis. We outline both lattice-based approaches and algebraic attacks, detailing the mechanics of the XL algorithm and F4/F5, while highlighting the theoretical limitations imposed by the semi-regularity assumption and reviewing existing attacks.

Building on this foundation, Chapter 3 introduces the proposed exact algebraic attack for the **BinaryError**-LWE problem. We detail the degree-fall mechanism and present a rigorous theoretical proof establishing the $\mathcal{O}(n^6)$ computational complexity bound. Subsequently, we adapt the proposed algorithm to address the **BinarySecret** variant, demonstrating how the additional field equations of a binary secret can be naturally integrated into our algebraic framework while maintaining the robust asymptotic complexity.

Chapter 4 then presents our experimental results, where we empirically validate these theoretical bounds and benchmark our attack against native MAGMA

implementations of F4. Finally, Appendices A and B contain the explicit mathematical probability bounds for matrix full-rank conditions, alongside the complete MAGMA source code of the proposed algorithm.

1 | Mathematical Foundations

This chapter introduces the fundamental mathematical concepts used throughout this thesis and establishes the notation adopted in the remainder of the manuscript. Its purpose is to provide the theoretical background necessary for the algebraic and cryptanalytic techniques developed in the subsequent chapters.

We begin with an overview of computational algebra, with particular emphasis on Gröbner basis, which constitute a central tool in algebraic cryptanalysis. After recalling the basic notions concerning polynomial rings and ideals, we introduce Gröbner basis, discuss their main structural properties, and explain how they can be employed to solve systems of polynomial equations. We also review the principal algorithms for computing Gröbner basis from a given set of generators, highlighting both their theoretical complexity and their practical performance.

We then turn our attention to the Learning With Errors (LWE) problem. We provide a formal definition of LWE, fix the notation that will be used throughout the thesis, and discuss the hardness assumptions underlying its security. Finally, we present several relevant variants of the LWE problem and outline their main characteristics, with particular emphasis on the settings that are most relevant to the algebraic techniques studied in this work.

The material presented in this chapter serves as the theoretical foundation for the cryptanalytic investigations developed in the remainder of the thesis.

1.1 Notation

In this section we fix the notation and recall basic algebraic notions that will be used throughout the thesis. We denote by $\mathbb{K}$ a field and by

$$R := \mathbb{K}[x_1, \dots, x_n]$$

the polynomial ring in the variables $x_1, \dots, x_n$ over $\mathbb{K}$.

Unless otherwise stated, we will mainly consider the case $\mathbb{K} = \mathbb{F}_q$, the finite field of q elements, where q is an odd prime. In this setting, we sometimes denote the corresponding polynomial ring by R_q . For simplicity of notation, we occasionally write $\mathbb{K}[x]$ when the exact set of variables is clear from the context.

Given a polynomial $p \in R$ and a fixed monomial ordering (which will be clear from the context), we denote by $\text{LM}_k(p)$ the k -th leading monomial of p .

In particular, $\text{LM}(p)$ denotes the leading monomial of p , $\text{LC}(p) \in \mathbb{K}$ its leading coefficient, and

$$\text{LT}(p) = \text{LC}(p) \cdot \text{LM}(p)$$

its leading term.

For a finite set B , we denote by $|B|$ its cardinality. Throughout the thesis, with a slight abuse of notation, $\deg(\cdot)$ may denote either the degree of a variable or of a polynomial; the intended meaning will be clear from the context.

Most of the algebraic background presented in this chapter follows standard references, in particular [25].

1.2 Affine Varieties and Ideals

Systems of multivariate polynomial equations naturally arise in computational algebra and in the algebraic approach to cryptanalysis. A fundamental task is to describe and study their solutions over a field $\mathbb{K}$. This can be approached from two complementary viewpoints: a geometric one, through affine varieties, and an algebraic one, through ideals in polynomial rings.

We begin by formalizing the geometric perspective.

Definition 1.2.1 (Affine variety). *Let $f_1, \dots, f_s \in \mathbb{K}[x_1, \dots, x_n]$. Then, the set*

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

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

While affine varieties describe solution sets geometrically, it is often more convenient to manipulate polynomial systems algebraically. This motivates the introduction of ideals in the polynomial ring.

Definition 1.2.2 (Ideal). *A subset $I \subseteq R$ is an ideal if it satisfies the following conditions:*

- $0 \in I$;
- if $f, g \in I$, then $f + g \in I$;
- if $f \in I$ and $h \in \mathbb{K}[x_1, \dots, x_n]$, then $h \cdot f \in I$.

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

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

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

The relationship between these two viewpoints becomes explicit by associating to an ideal the set of its common zeros.

Definition 1.2.3 (Variety of ideal). *Let $s, n \in \mathbb{N}$ and let $I = \langle f_1, \dots, f_s \rangle$ be an ideal in $\mathbb{K}[x_1, \dots, x_n]$. Then, the set*

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

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

Conversely, to any affine variety one can associate the set of all polynomials vanishing on it.

Definition 1.2.4 (Ideal of variety). *If $\mathbb{V} \in \mathbb{K}^n$ is an affine variety, then $I(\mathbb{V}) \subseteq \mathbb{K}[x_1, \dots, x_n]$ where*

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

is an ideal, and we call it the ideal of the variety $\mathbb{V}$.

This duality between solution sets and polynomial ideals is fundamental in computational algebraic geometry and underlies the algorithmic techniques developed in the following sections, most notably Gröbner basis methods.

1.3 Gröbner Basis

Gröbner basis constitute the main algorithmic tool for handling systems of multivariate polynomial equations. Their definition relies on the notion of monomial ordering, which allows one to identify leading terms in multivariate polynomials.

1.3.1 Monomial Orderings

To define Gröbner basis, we first need the concept of a *monomial ordering*.

Definition 1.3.1 (Monomial ordering). *A monomial ordering $\succ$ on $\mathbb{K}[x_1, \dots, x_n]$ is a relation $\succ$ on $\mathbb{Z}_{\geq 0}^n$, or rather a relation on the set of monomials x^α where $\alpha \in \mathbb{Z}_{\geq 0}^n$ satisfying the following conditions:*

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

This ordering allows us to define the *leading monomial* of a polynomial, which in turn is essential to describe the structure of an ideal with respect to computational algorithms. The most commonly used monomial orderings are defined as follows (letting $\alpha, \beta \in \mathbb{Z}_{\geq 0}^n$):

- **Lexicographic (LEX):** We say $\alpha \succ_{lex} \beta$ if the leftmost non-zero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is positive. We will write $x^\alpha \succ_{lex} x^\beta$ if $\alpha \succ_{lex} \beta$.
- **Reverse Lexicographic (RLEX):** We say $\alpha \succ_{rlex} \beta$ if the rightmost non-zero entry of the vector difference $\alpha - \beta \in \mathbb{Z}^n$ is negative. We will write $x^\alpha \succ_{rlex} x^\beta$ if $\alpha \succ_{rlex} \beta$.
- **Graded Lexicographic (GRLEX):** We say $\alpha \succ_{grlex} \beta$ if $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, or $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and $\alpha \succ_{lex} \beta$. We will write $x^\alpha \succ_{grlex} x^\beta$ if $\alpha \succ_{grlex} \beta$.
- **Graded Reverse Lexicographic (GREVLEX/DRL):** We say $\alpha \succ_{grevlex} \beta$ if $\sum_{i=1}^n \alpha_i > \sum_{i=1}^n \beta_i$, or $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and $\alpha \succ_{rlex} \beta$. We will write $x^\alpha \succ_{grevlex} x^\beta$ if $\alpha \succ_{grevlex} \beta$.

1.3.2 Gröbner Basis

We now introduce Gröbner basis, which provide a canonical representation of an ideal with respect to a monomial ordering.

Theorem 1.3.2 (Hilbert Basis theorem). *Every ideal $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ has a finite generating set. In other words, $I = \langle g_1, \dots, g_s \rangle$ for some $g_1, \dots, g_s \in I$.*

Before introducing Gröbner bases, we must formalize the concept of the ideal generated by the leading terms. Once a monomial ordering is chosen, every non-zero polynomial $f \in \mathbb{K}[x_1, \dots, x_n]$ has a unique leading term $\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f)$, where $\text{LC}(f)$ is the leading coefficient and $\text{LM}(f)$ is the leading monomial.

Definition 1.3.3 (Ideal of Leading Terms). *Let $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ be an ideal other than $\{0\}$, and fix a monomial ordering on $\mathbb{K}[x_1, \dots, x_n]$.*

- We denote by $\text{LT}(I)$ the set of leading terms of all non-zero elements of I :

$$\text{LT}(I) = \{\text{LT}(f) \mid f \in I \setminus \{0\}\}.$$

- We denote by $\langle \text{LT}(I) \rangle$ the ideal generated by the elements of $\text{LT}(I)$.

This definition brings up a subtle but fundamental point. If we are given a finite generating set for I , say $I = \langle f_1, \dots, f_s \rangle$, it is always true by definition that $\text{LT}(f_i) \in \text{LT}(I)$, which implies $\langle \text{LT}(f_1), \dots, \text{LT}(f_s) \rangle \subseteq \langle \text{LT}(I) \rangle$. However, the ideal $\langle \text{LT}(I) \rangle$ can be strictly larger. A Gröbner basis is precisely a generating set that forces this subset relation to become an equality.

Definition 1.3.4 (Gröbner basis [25]). *Fix a monomial ordering on the polynomial ring $\mathbb{K}[x_1, \dots, x_n]$. A finite subset $G = \{g_1, \dots, g_t\}$ of an ideal $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ different from $\{0\}$ is said to be a Gröbner basis (or standard basis) if*

$$\langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle = \langle \text{LT}(I) \rangle.$$

Using the convention that $\langle \emptyset \rangle = \{0\}$, we define the empty set $\emptyset$ to be the Gröbner basis of the zero ideal $\{0\}$.

Definition 1.3.5 (Reduced Gröbner basis). *Let G be a Gröbner basis for the ideal $I \subset \mathbb{K}[x_1, \dots, x_n]$ w.r.t. a monomial ordering $\succ$. If*

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

G is said to be a reduced Gröbner basis.

Reduced Gröbner basis are particularly useful because polynomial division modulo a reduced Gröbner basis yields unique remainders. Consequently, every residue class in the quotient ring R/I can be uniquely represented using the basis monomials.

1.3.3 Quotient Space and Zero-Dimensional Ideals

The quotient ring R/I is a $\mathbb{K}$ -vector space, sometimes called the *quotient space*. A standard basis for this space is given by the set of all monomials that are not divisible by any leading monomial in the ideal:

$$B_I = \{x^\alpha \mid x^\alpha \notin \langle \text{LM}(I) \rangle\}.$$

Definition 1.3.6 (Zero-dimensional ideal). *Let $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ be a non-zero ideal. If the quotient space $\mathbb{K}[x_1, \dots, x_n]/I$ is finite-dimensional as $\mathbb{K}$ -vector space, that is,*

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

then I is called a Zero-dimensional ideal.

The property of being zero-dimensional is of profound importance in algebraic cryptanalysis because it characterizes systems of equations that have a finite number of solutions. The common roots to all polynomials in an ideal I are referred to as the *zeros of the ideal I* . These zeros precisely constitute the points of the affine variety $\mathbb{V}_{\overline{\mathbb{K}}}(I)$. Specifically, an ideal I is zero-dimensional if and only if it has finitely many zeros in the algebraic closure $\overline{\mathbb{K}}$, meaning its affine variety is a finite set.

Theorem 1.3.7. *Let $I \subseteq \mathbb{K}[x_1, \dots, x_n]$ be an ideal and let G be a Gröbner basis of I with respect to some monomial ordering. The ideal I is zero-dimensional if and only if for every variable x_i , $1 \leq i \leq n$, there exists a polynomial $g_i \in G$ such that $\text{LT}(g_i)$ is a pure power of x_i .*

Moreover, if I is zero-dimensional, the total number of zeros of I in $\overline{\mathbb{K}}^n$ (counted with multiplicity) is exactly equal to $\dim_{\mathbb{K}}(\mathbb{K}[x_1, \dots, x_n]/I)$.

In the context of algebraic cryptanalysis against the LWE problem, we routinely deal with overdetermined systems of polynomial equations. By collecting a sufficient number of LWE samples (denoted by m), the resulting algebraic system contains significantly more equations than unknown variables. When processing such an overdetermined system under an appropriately chosen monomial ordering, the abundance of algebraically independent relations ensures that the resulting Gröbner basis will naturally contain polynomials whose leading monomials are pure powers of each variable x_i . Consequently, because we require a sufficient number of samples to start the attack, we can always safely assume throughout this thesis that the polynomial systems generated by our LWE instances define Zero-dimensional ideals.

For Zero-dimensional ideals, the computation of their variety reduces to factoring a univariate polynomial, making the structure of the quotient space particularly relevant for algorithmic applications. Linear transformations corresponding to multiplication by variables in the quotient space can then be defined as

$$T_j : R/I \rightarrow R/I, \quad 1 \leq j \leq n,$$

and play a central role in algebraic cryptanalysis techniques.

1.4 Algorithms for Computing Gröbner Basis

Having introduced Gröbner basis and their properties, we now discuss the computational aspects: how Gröbner basis can be effectively computed from a given set of polynomials, and which algorithmic techniques are commonly used in practice.

1.4.1 S-Polynomials and Buchberger's Criteria

The classical method for computing a Gröbner basis is the *Buchberger algorithm*, which relies on the concept of *S-polynomials*.

Definition 1.4.1 (*S-polynomial*). *Let $f, g \in \mathbb{K}[x_1, \dots, x_n]$ be non-zero polynomials.*

- *Let $\text{LM}(f) = x^\alpha$ and $\text{LM}(g) = x^\beta$. We define their least common multiple as $x^\gamma = \text{lcm}(\text{LM}(f), \text{LM}(g))$, where $\gamma = (\gamma_1, \dots, \gamma_n)$ with $\gamma_i = \max(\alpha_i, \beta_i)$ for $1 \leq i \leq n$.*

- The S -polynomial of f and g is defined as

$$S(f, g) = \frac{x^\gamma}{\text{LT}(f)} \cdot f - \frac{x^\gamma}{\text{LT}(g)} \cdot g.$$

Buchberger introduced two criteria to reduce the number of S -polynomial pairs that need to be considered:

Theorem 1.4.2 (First Buchberger criterion or S -pair criterion). *A set of polynomials $G = \{g_1, \dots, g_s\}$ is a Gröbner basis of an ideal I if and only if for all pairs $i \neq j$, the remainder of $S(g_i, g_j)$ upon division by G is zero:*

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

Theorem 1.4.3 (Second Buchberger criterion). *Let $G = \{g_1, \dots, g_s\}$ and suppose there exist distinct elements $g_i, g_j, g_\ell \in G$ such that*

$$\text{LT}(g_\ell) \mid \text{lcm}(\text{LT}(g_i), \text{LT}(g_j)).$$

Then the S -polynomial $S(g_i, g_j)$ can be omitted from consideration, as it can be expressed as a combination of $S(g_i, g_\ell)$ and $S(g_j, g_\ell)$.

These two criteria allow the algorithm to skip many unnecessary reductions, significantly improving efficiency in practice.

1.4.2 Buchberger's Algorithm

Buchberger's algorithm [20], first introduced in 1965, iteratively adds polynomials obtained from S -polynomials to the current basis until all S -polynomials reduce to zero. The algorithm is guided by two key criteria (Buchberger's first and second criteria) that allow skipping certain S -pairs and reducing the number of computations. This ensures that the resulting set is indeed a valid Gröbner basis.

At a high level, the deterministic procedure can be expressed as follows:

Algorithm 1 Buchberger's Algorithm (Basic Version) [26]

```

1: procedure BUCHBERGER( $F = \{f_1, \dots, f_s\}$ )
2:    $G := F$ 
3:   repeat
4:      $G' := G$ 
5:     for each pair  $p, q \in G'$  with  $p \neq q$  do
6:        $S := \overline{S(p, q)}^{G'}$  ▷ Remainder of  $S(p, q)$  on division by  $G'$ 
7:       if  $S \neq 0$  then
8:          $G := G \cup \{S\}$ 

```

```

9:         end if
10:    end for
11:    until  $G = G'$ 
12:    return  $G$ 
13: end procedure

```

While conceptually simple, the computational complexity of Buchberger’s algorithm grows extremely quickly. The theoretical worst-case complexity for Gröbner basis computation is doubly exponential in the number of variables, bounded by:

$$2 \left(\frac{d^2}{2} + d \right)^{2^{n-2}}$$

where d is the maximum degree of the input polynomials.

In practice, depending on the specific system of equations, ad-hoc complexity bounds can be derived. However, the generation of intermediate S -polynomials often triggers severe degree inflation, creating an exponential spike in memory consumption long before theoretical time limits are reached. Advanced strategies and specialized improvements, such as the F4 algorithm or structured degree-fall mechanisms, are therefore strictly necessary for practical computations in algebraic cryptanalysis. However, it is crucial to note that algorithm efficiency is intrinsically tied to the specific system. In certain particular cases, the heavy linear algebra overhead of F4 becomes a bottleneck, making the standard Buchberger’s algorithm a more efficient approach.

1.4.3 Faugère’s F4 and F5 Algorithms

While Buchberger’s algorithm provides the foundational mathematics for computing Gröbner basis, its sequential nature (reducing a single S -polynomial at a time via multivariate division) renders it highly inefficient for large cryptographic systems. To overcome these practical limitations, Jean-Charles Faugère introduced the F4 algorithm in 1999 [29] and the F5 algorithm in 2002 [30]. These algorithms revolutionized Polynomial System Solving (PoSSo) by replacing sequential polynomial division with highly optimized linear algebra.

The F4 Algorithm. The primary innovation of the F4 algorithm is the use of linear algebra to perform the simultaneous computation of several S -polynomial remainders. Instead of selecting and reducing a single pair at a time, F4 selects a batch of pairs, typically using a normal selection strategy that prioritizes pairs of minimal degree.

Crucially, the algorithm does not explicitly compute the S -polynomials by taking their differences upfront. Instead, it extracts the two “halves” of the S -polynomials for the selected pairs and collects them into an initial set. It then systematically adjoins necessary “reducer” polynomials, constructed by multiplying

elements of the current basis by appropriate monomials, to ensure that generated terms can be canceled. This entire collection of raw material forms the rows of a large coefficient matrix, known as the Macaulay matrix.

By computing the row reduced echelon form of this matrix via Gaussian elimination, F4 effectively reduces the selected pairs simultaneously. The rows of the resulting echelon matrix that yield new leading monomials correspond to the non-zero remainders, which are then added to the Gröbner basis. This matrix-based approach allows the algorithm to leverage highly optimized, low-level linear algebra libraries and modern CPU architectures. Because of its massive speedup over traditional division algorithms, F4 forms the backbone of Gröbner basis computation in almost all modern computer algebra systems, including MAGMA.

The F5 Algorithm. Despite the linear-algebraic speedups of F4, a fundamental theoretical bottleneck remains: a vast majority of S -polynomials ultimately reduce to zero. These zero-reductions contribute nothing to the final Gröbner basis but consume enormous amounts of computational time and memory. These redundant computations are caused by trivial syzygies (algebraic relations between the polynomials themselves, such as $f_i \cdot f_j - f_j \cdot f_i = 0$).

The F5 algorithm was developed to explicitly prevent these useless computations. It does so by attaching a mathematical *signature* (or label) to every polynomial, which tracks its algebraic history and generation path. By applying two new rules (the F5 criterion and the rewritten criterion) the algorithm can inspect the signatures of an S -pair and mathematically prove whether it will reduce to zero before performing any matrix operations. For regular and semi-regular polynomial systems, F5 mathematically guarantees that there will be exactly zero useless reductions, making it the most optimal Gröbner basis algorithm known for such systems.

Together, F4 and F5 represent the state-of-the-art in exact algebraic solving. However, in the specific context of overdetermined systems over finite fields (such as those derived from the LWE problem), the construction of massive matrices in F4 can still lead to fatal memory exhaustion due to degree inflation. This limitation motivates the search for specialized algebraic algorithms (like the one proposed in this thesis) that restrict the matrix degree and prevent exponential memory growth.

1.4.4 Linearization and the Macaulay Matrix

To introduce algebraic solving techniques like the XL algorithm or Gröbner basis, we first require the fundamental concept of the *Macaulay matrix*. This structure allows us to reduce complex, non-linear polynomial systems into linear algebra problems through a process called *linearization*, where every distinct monomial is treated as an independent linear variable.

Let $\mathcal{F} = \{f_1, \dots, f_m\}$ denote a system of polynomials in the ring $\mathbb{K}[x_1, \dots, x_n]$, and let $\prec$ represent a chosen generic monomial ordering. For any target degree $D \in \mathbb{Z}_+$, the Macaulay matrix $M_{\leq D}$ associated with $\mathcal{F}$ is constructed such that its columns are indexed by all possible monomials in the ring of degree at most D . To facilitate proper elimination, these columns are sorted in strictly decreasing order with respect to $\prec$.

The rows of $M_{\leq D}$ are indexed by the expanded polynomials $\mu \cdot f$, where μ is a monomial multiplier ensuring that the total degree satisfies $\deg(\mu \cdot f) \leq D$. Consequently, the scalar entry located at the intersection of the i -th row and the j -th column of $M_{\leq D}$ is exactly the coefficient of the j -th monomial within the expanded polynomial corresponding to that row.

Example 1.4.4 (Constructing a Macaulay Matrix [22]). To illustrate this construction in practice, let us consider the polynomial ring $\mathbb{K}[x, y]$ equipped with the descending Degree Reverse Lexicographic (DRL) ordering. Let our initial system consist of two polynomials:

$$\begin{aligned} f_1 &= 2x^2 + 3xy + 7, \\ f_2 &= 4x + 1. \end{aligned}$$

Suppose we wish to build the Macaulay matrix $M_{\leq 3}$, meaning our target degree is $D = 3$. First, we identify all monomials up to degree 3 and sort them in decreasing order with respect to our chosen ordering. These form the columns of our matrix:

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

Next, for each polynomial, we determine the set of monomials μ such that $\deg(\mu \cdot f_i) \leq 3$. For f_1 (which has degree 2), the valid multipliers are 1, x , and y :

$$\begin{aligned} 1 \cdot f_1 &= 2x^2 + 3xy + 7, \\ x \cdot f_1 &= 2x^3 + 3x^2y + 7x, \\ y \cdot f_1 &= 2x^2y + 3xy^2 + 7y. \end{aligned}$$

For f_2 (which has degree 1), the valid multipliers are 1, x , y , x^2 , xy , and y^2 :

$$\begin{aligned} 1 \cdot f_2 &= 4x + 1, \\ x \cdot f_2 &= 4x^2 + x, \\ y \cdot f_2 &= 4xy + y, \\ x^2 \cdot f_2 &= 4x^3 + x^2, \\ xy \cdot f_2 &= 4x^2y + xy, \\ y^2 \cdot f_2 &= 4xy^2 + y^2. \end{aligned}$$

Finally, we extract the coefficients of these expanded polynomials and place them into the matrix grid. The resulting Macaulay matrix $M_{\leq 3}$ is:

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

1.4.5 The XL Algorithm

Up to this point, our algebraic definitions have been stated over a general field $\mathbb{K}$. However, for the remainder of this thesis, specifically in the context of algebraic cryptanalysis and the LWE problem, we strictly restrict our attention to finite fields, setting $\mathbb{K} = \mathbb{F}_q$.

With the Macaulay matrix defined over $\mathbb{F}_q$, we can now introduce the *eXtended Linearization (XL)* algorithm [24]. Developed to solve overdetermined systems of multivariate polynomial equations over finite fields, XL operates by reducing the highly complex problem of polynomial system solving into a straightforward linear algebra problem.

The algorithm proceeds through three primary phases:

1. **Multiplication (Extension):** The original polynomials $f_i \in \mathcal{F}$ are multiplied by all possible monomials μ such that the total degree of the resulting polynomial $\mu \cdot f_i$ does not exceed a chosen target degree D .
2. **Linearization:** The expanded system is represented as a massive Macaulay matrix $M_{\leq D}$.
3. **Solving:** Gaussian elimination is performed on the Macaulay matrix. Because the system is overdetermined, the goal is to produce step-wise reductions that eventually yield univariate equations (e.g., $x_i - c_i = 0$), completely revealing the variable assignments.

The algorithmic flow of XL can be concisely represented as follows:

Algorithm 2 XL Algorithm

Require: Overdetermined system $\mathcal{F}$, target degree D

Ensure: Solution vector for the variables $x_1, \dots, x_n$

1: $\mathcal{F}_{\text{ext}} \leftarrow \mathcal{F}$

2:

▷ Phase 1: Extension

```

3: for each  $f_i \in \mathcal{F}$  do
4:   for each monomial  $\mu$  where  $\deg(\mu) + \deg(f_i) \leq D$  do
5:      $\mathcal{F}_{\text{ext}} \leftarrow \mathcal{F}_{\text{ext}} \cup \{\mu \cdot f_i\}$ 
6:   end for
7: end for
8:                                     ▷ Phase 2: Linearization
9:  $\mathcal{M}_{\leq D} \leftarrow \text{BuildMacaulayMatrix}(\mathcal{F}_{\text{ext}})$ 
10:                                     ▷ Phase 3: Solving
11:  $\mathcal{M}_{\text{RREF}} \leftarrow \text{GaussianElimination}(\mathcal{M})$ 
12:  $\mathcal{S} \leftarrow \text{ExtractUnivariateRelations}(\mathcal{M}_{\text{RREF}})$ 
13: return  $\mathcal{S}$ 

```

Relevance to LWE and Algebraic Cryptanalysis

Because of its strictly linear-algebraic structure, XL appears naturally in the cryptanalysis of problems like LWE. It can be viewed as a specialized, highly structured approximation of Gröbner basis computation that is particularly intuitive for systems over small finite fields.

However, the standard XL algorithm suffers from a severe theoretical bottleneck: to find enough linearly independent rows to fully solve the system, the operating degree D must often be set quite high. The size of the Macaulay matrix grows proportionally to $\binom{n+D}{D}$, leading to exponential memory exhaustion. Modern exact algebraic algorithms targeting specialized LWE instances (such as the algorithm proposed in this work) attempt to bypass this limitation by carefully orchestrating polynomial combinations that induce a degree-fall, thereby extracting the secret variables without the memory expansion required by traditional XL extension.

1.5 Basics on LWE

The Learning With Errors problem was introduced by Oded Regev in 2005 [45] as a noisy generalization of the learning parity with noise problem over finite fields.

One of the main features that makes LWE particularly attractive from a cryptographic perspective is the existence of reductions from worst-case lattice problems to average-case instances of LWE. More precisely, Regev showed that an efficient algorithm for solving LWE on average would yield an efficient quantum algorithm for approximating lattice problems such as GapSVP and SIVP in the worst case, for polynomial approximation factors [45, 46]. Subsequently, Brakerski, Langlois, Peikert, Regev, and Stehlé [19] established a fully classical worst-case to average-case reduction for LWE, showing that solving LWE efficiently would also imply efficient classical algorithms for approximating standard lattice problems such as GapSVP and BDD in the worst case. This worst-case to average-case

connection provides strong evidence for the hardness of LWE and distinguishes it from many classical cryptographic assumptions.

At a high level, LWE can be viewed as a noisy variant of solving linear systems. Given a system of linear equations

$$\mathbf{A}\mathbf{s} = \mathbf{b} \quad \text{over } \mathbb{Z}_q,$$

where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is a known matrix, $\mathbf{b} \in \mathbb{Z}_q^m$ is a known vector and $\mathbf{s} \in \mathbb{Z}_q^n$ is an unknown secret vector, the secret $\mathbf{s}$ can be efficiently recovered using standard Gaussian elimination. However, when a small error vector $\mathbf{e} \in \mathbb{Z}_q^m$ is added to the system, producing

$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b},$$

recovering $\mathbf{s}$ becomes computationally intractable. LWE exploits this difficulty: cryptographic schemes are constructed so that an attacker must either recover the secret $\mathbf{s}$ (search-LWE) or distinguish LWE samples from uniform random samples (decision-LWE).

We formally define the problem following Regev's original formulation [45], where the rows of the matrix $\mathbf{A}$ are generated as independent vectors $\mathbf{a} \in \mathbb{Z}_q^n$.

Definition 1.5.1 (LWE [4, 45]). *Let n, q be positive integers, χ be a probability distribution on $\mathbb{Z}$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s}, \chi}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}$ according to χ and considering it in $\mathbb{Z}_q$, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

- *Decision-LWE is the problem of deciding whether pairs $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $L_{\mathbf{s}, \chi}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*
- *Search-LWE is the problem of recovering $\mathbf{s}$ from $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s}, \chi}$.*

In the context of algebraic cryptanalysis, stacking m independent samples drawn from $L_{\mathbf{s}, \chi}$ naturally reconstructs the matrix formulation $\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b}$, where the vectors $\mathbf{a}_i$ form the rows of $\mathbf{A}$, and the scalar elements b_i and e_i form the column vectors $\mathbf{b}$ and $\mathbf{e}$, respectively.

The distinction between the search and decision versions of LWE is mainly technical, as the two problems are polynomially equivalent under standard parameter choices. In practice, the hardness of LWE relies on the presence of the error distribution χ , which prevents the recovery of the secret vector $\mathbf{s}$ through linear algebra techniques [46].

However, the dimension n required to achieve a desired security level may lead to large public keys and significant computational costs. For this reason, several structured variants of LWE have been introduced, aimed at improving efficiency while preserving comparable hardness guarantees, most notably Ring-LWE [40]

and Module-LWE [17]. These variants exploit algebraic structures such as rings or modules, leading to reduced key sizes and faster computations, and underlie practical schemes such as CRYSTALS-Kyber [8].

1.5.1 Applications of LWE

Since its introduction, the LWE problem has emerged as a remarkably rich and versatile foundation in cryptography. It has been used as the basis for a wide range of cryptographic constructions, including:

- secure public-key encryption schemes resilient to both chosen-plaintext [39, 43, 45] and chosen-ciphertext attacks [41, 42, 44],
- oblivious transfer protocols [43],
- identity-based encryption [1, 2, 23, 33],
- leakage-resilient cryptography [3, 6, 35],
- fully homomorphic encryption [15, 16, 18],

among many others. These applications demonstrate the versatility of LWE and its central role in modern and post-quantum cryptography.

1.5.2 LWE Variants

To optimize computational efficiency, reduce key sizes, and adapt to constrained hardware environments, a rich body of structural variants has been developed from the standard LWE problem. While these variants introduce additional structure or restrict the parameter space, they are designed to maintain robust security guarantees based on worst-case lattice problems. In this section, we present the most relevant LWE variants, focusing particularly on those targeted by the algebraic techniques developed in this thesis.

Bounded-Error LWE. In Bounded-Error LWE, the error term is restricted to a known bounded interval rather than being drawn from a full probability distribution. This variant allows for simplified analyses and can be used to model practical implementations where the error magnitude is controlled, while maintaining the essential hardness assumptions of standard LWE.

Binary-Error LWE. Binary-Error LWE is a variant where each error component is sampled from $\{0, 1\}$. This discrete and sparse error distribution reduces computational overhead and has been used in lightweight cryptographic schemes, particularly for resource-constrained devices.

Binary-Secret LWE. Binary-Secret LWE restricts the secret vector $\mathbf{s}$ to binary values (e.g., $\{0, 1\}^n$). This restriction is useful in constructions where compact representation or efficient computation is required, while preserving the hardness of the underlying problem under standard LWE assumptions.

Ring-LWE. One of the main drawbacks of the standard LWE problem is that its parameters grow linearly with the dimension n , which leads to large keys and relatively expensive arithmetic operations. To address these limitations, Lyubashevsky, Peikert, and Regev introduced the Ring Learning With Errors problem (Ring-LWE, denoted RLWE) [40].

In RLWE, vectors over $\mathbb{Z}_q$ are replaced by elements of a polynomial ring $R_q = \mathbb{Z}_q[x]/(f(x))$, where $f(x)$ is typically a cyclotomic polynomial. Samples are then given as pairs $(a, c) \in R_q \times R_q$ of the form $c = a \cdot s + e$, where a is sampled uniformly at random from R_q , s is the secret element, and e is drawn from a suitable error distribution over the ring.

The additional algebraic structure allows more compact representations and faster arithmetic through polynomial multiplication, making RLWE particularly attractive for practical implementations. At the same time, the presence of structure requires additional care in security analyses, since the hardness assumption is no longer purely unstructured as in standard LWE.

Module-LWE. The Module Learning With Errors problem (Module-LWE, denoted MLWE) [17] can be viewed as an intermediate construction between standard LWE and Ring-LWE. Instead of working with vectors over $\mathbb{Z}_q$ or single elements of a ring, Module-LWE considers vectors over a ring R_q of small rank k , that is, elements of R_q^k .

This formulation allows a finer trade-off between efficiency and security. For $k = 1$, MLWE coincides with RLWE, while for large values of k it approaches the structure of standard LWE. As a result, Module-LWE retains most of the efficiency advantages of Ring-LWE while reducing the amount of algebraic structure, and it is therefore widely used in modern lattice-based cryptographic constructions and standardization efforts.

Applications of MLWE. MLWE underlies several practical key encapsulation mechanisms (KEMs) and public-key encryption schemes. The most notable example is CRYSTALS-Kyber [8], which relies on MLWE to achieve a highly efficient trade-off between execution speed, key size, and security. By taking advantage of the reduced algebraic structure of MLWE compared to RLWE, Kyber allows for implementations that are both faster and more compact while retaining worst-case lattice security guarantees.

Learning With Rounding. Introduced by Banerjee, Peikert, and Rosen in 2012 [11], the Learning With Rounding (LWR) problem replaces the additive Gaussian error $\mathbf{e}$ of standard LWE with a deterministic rounding operation. Instead of computing $\mathbf{As} + \mathbf{e} \pmod{q}$, the LWR problem computes $\lfloor (p/q) \cdot \mathbf{As} \rfloor \pmod{p}$ for some modulus $p < q$. This deterministic rounding effectively drops the low-order bits, creating an implicit “error” that provides similar security guarantees to LWE. LWR is highly advantageous in lightweight cryptography because it completely eliminates the need for complex, secure Gaussian error samplers.

Module-LWR. By combining the algebraic efficiency of modules with the deterministic error generation of LWR, the Module Learning With Rounding (denoted MLWR) problem is constructed. MLWR serves as the underlying hardness assumption for the cryptographic suite SABER [28] (and its lightweight parameter set, LightSaber). SABER was a prominent finalist in the NIST Post-Quantum Cryptography standardization process, demonstrating that MLWR achieves a highly practical trade-off between execution speed, key size, and security in hardware-constrained environments.

2 | Cryptanalysis of LWE

During the past years, a variety of attacks targeting LWE have been introduced and extensively analyzed. A systematic overview of the state of the art is given in the survey by Albrecht, Player, and Scott [4]. Based on the techniques used to exploit the structure of LWE instances, these attacks can be broadly divided into three main categories:

1. **Lattice-Based Attacks:** These attacks exploit the underlying geometric structure of the LWE problem by framing it as a computational problem over integer lattices. Utilizing powerful lattice reduction algorithms like LLL [38] and BKZ [47], they are consistently the most efficient and practical methods for solving typical LWE instances, especially those featuring the large moduli required in homomorphic encryption.
2. **Combinatorial Attacks:** Originating from the cryptanalysis of the Learning Parity with Noise (LPN) problem, combinatorial methods primarily rely on searching for collisions among the sample vectors to systematically eliminate noise and isolate the secret components. The archetypal algorithm in this family is the Blum-Kalai-Wasserman (BKW [13]) algorithm.
3. **Algebraic Attacks:** Unlike geometric or combinatorial approaches, algebraic attacks treat the LWE noise as a root of a known, noise-free polynomial. By generating systems of non-linear equations, these attacks attempt to recover the secret through advanced algebraic solving techniques, such as linearization or the computation of Gröbner basis.

2.1 Fundamental Reductions and Strategies

Regardless of the specific algorithmic family employed, most attacks on LWE operate by transforming the problem into one of three fundamental target strategies. These reductions bridge the gap between the algebraic definition of LWE and the computational subroutines used by the attacker.

Short Integer Solutions (SIS)

The first strategy solves decision-LWE by reducing it to the Short Integer Solutions (SIS) problem. Given m samples represented as a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{c} = \mathbf{As} + \mathbf{e}$, the attacker attempts to find a suitably short vector $\mathbf{v}$ in the scaled dual lattice $\Lambda^\perp = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{wA} \equiv \mathbf{0} \in \mathbb{Z}_q^n \pmod{q}\}$. Computing the inner product yields $\langle \mathbf{v}, \mathbf{c} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle$. Because both $\mathbf{v}$ and $\mathbf{e}$ are short, their inner product remains small, following a Gaussian distribution. This allows the attacker to distinguish valid LWE samples from a uniform distribution. To achieve a target success probability ϵ , the required norm of the short vector is approximated by $\|\mathbf{v}\| \approx \frac{1}{\alpha} \sqrt{\ln(1/\epsilon)/\pi}$ [4], where $\sigma = \alpha q$ is the standard deviation of the LWE error distribution (α being the relative noise rate), and the vector norm is defined as $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$.

Bounded Distance Decoding (BDD)

Alternatively, an LWE instance can be directly viewed as a Bounded Distance Decoding (BDD) problem on the primal lattice. The target vector $\mathbf{c}$ lies close to a point $\mathbf{As}$ within the lattice spanned by the columns of $\mathbf{A}$. The distance between the target and the lattice point is precisely the error vector $\mathbf{e}$. Finding the closest lattice point to $\mathbf{c}$ naturally recovers the secret $\mathbf{s}$, directly solving the search-LWE problem.

Solving for the Secret Directly

The final strategy bypasses geometric formulations to search for a candidate secret $\mathbf{s}'$ such that the discrepancy $\|\mathbf{As}' - \mathbf{c}\|$ is small, directly satisfying the bounds of the discrete Gaussian noise. This approach is the conceptual foundation for brute-force searches, meet-in-the-middle optimizations, and purely algebraic formulations like the Arora-Ge algorithm [7].

2.2 Lattice-Based Attacks

Lattice-based attacks represent the most formidable and efficient threat to standard LWE parameterizations. These attacks utilize lattice basis reduction algorithms, most notably LLL (Lenstra-Lenstra-Lovász) and BKZ (Block Korkine-Zolotarev), to uncover short vectors in either the primal or dual lattice representations of the LWE instance.

When applying lattice reduction to distinguish via the SIS strategy, the attacker must find a short vector $\mathbf{v}$ in an m -dimensional scaled dual lattice. The success of the attack hinges entirely on the quality of the reduced basis, parameterized by the root-Hermite factor δ_0 . Because the attacker generally has access to an arbitrary number of LWE samples m , they can optimize the lattice dimension

to minimize the required δ_0 . The optimal number of samples (subdimension) to process is $m = \sqrt{\frac{n \log q}{\log \delta_0}}$.

Alternatively, attackers solve the BDD representation using variants of Babai's Nearest Plane algorithm [9, 39] or by utilizing Kannan's embedding technique [37] to reduce the BDD instance into a unique Shortest Vector Problem (uSVP). The time complexity of these attacks is generally dominated by the underlying exact SVP oracle called by BKZ in dimension k . While lattice sieving reduces this subroutine cost asymptotically to $2^{\mathcal{O}(k)}$, lattice reduction remains the unequivocally most efficient cryptanalytic approach for large moduli q , heavily outperforming combinatorial and algebraic equivalents.

2.3 Combinatorial Attacks

Combinatorial attacks, primarily built upon the Blum-Kalai-Wasserman (BKW) algorithm [13], adapt techniques originally designed for the Learning Parity with Noise problem to the larger moduli of LWE.

The core idea behind BKW is to split the n components of the sample vectors $\mathbf{a}$ into blocks, and systematically search for collisions on these subsets. By adding or subtracting samples that collide on specific coordinates, the algorithm effectively clears those components, gradually constructing a short vector $\mathbf{v}$ in the dual lattice (solving SIS).

While BKW achieves a subexponential time complexity of $\mathcal{O}(2^{n/(2-o(1))})$ when table creation and evaluation phases are carefully balanced, it suffers from severe practical limitations. The algorithm requires an immense sample complexity and memory storage overhead that scales as $2^{\mathcal{O}(n)}$, making it highly uncompetitive against lattice reduction when applied to instances with large cryptographic moduli q .

2.4 Algebraic Attacks

Algebraic attacks model LWE instances as deterministic systems of polynomial equations, subsequently attempting to solve them using advanced algebraic geometry techniques. It is important to note a fundamental algebraic distinction: while the standard LWE problem is formally defined over the integer quotient ring $\mathbb{Z}_q$, exact algebraic cryptanalysis, such as the computation of Gröbner basis, requires arithmetic over a field to avoid zero-divisors. Consequently, in algebraic attacks, the modulus q is almost universally assumed to be prime, establishing the canonical isomorphism $\mathbb{Z}_q \cong \mathbb{F}_q$. Thus, for the remainder of this exposition, all polynomial systems will be modeled over the finite field $\mathbb{F}_q$.

Unlike geometric approaches that treat the discrete Gaussian error as a distance metric, algebraic attacks treat this error as a root of a deterministic polyno-

mial equation. By exploiting the fact that the Gaussian error e is bounded within a small integer interval $[-t, t]$ with overwhelming probability (where the bound t is typically a function of the standard deviation σ), attackers can construct a noise-free, non-linear system of multivariate polynomials over $\mathbb{F}_q$.

Although these attacks do not currently threaten the most practical, high-noise parameter sets as effectively as lattice reduction methods, they provide profound mathematical insight into the fundamental algebraic structure of LWE. In particular, they define the problem's theoretical limits in low-noise or highly overdetermined regimes.

While early formulations of this strategy utilized high-degree linearization to resolve the resulting systems, modern refinements employ more sophisticated algebraic geometry tools, relying specifically on Gröbner basis. In the following detailed exposition, we will examine the specific mechanics, degrees of regularity, and sample complexity trade-offs inherent to this algebraic modeling.

2.4.1 General Modeling Approach

The conceptual foundation of an algebraic attack lies in translating a geometric or probabilistic problem into a Polynomial System Solving (PoSSo) instance. Given m LWE samples:

$$b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \pmod{q}$$

one can rewrite them as polynomial constraints involving the unknown secret variables $\mathbf{s} = (s_1, \dots, s_n)$ and the unknown error terms e_i . Because the exact value of the error variables is unknown, pure linear algebra cannot isolate the secret. However, algebraic attacks exploit the fundamental characteristic of LWE: the error distribution has a highly concentrated, small known support.

By encoding the finite set of possible values for the error into low-degree polynomial relations, the LWE samples can be transformed into a multivariate polynomial system strictly over $\mathbb{Z}_q$. The resulting non-linear system can then be attacked using linearization paradigms or Gröbner basis computations.

2.4.2 The Linearization Technique

The first dedicated algebraic attack on LWE was introduced by Arora and Ge in 2011 [7]. Their methodology rigorously models each LWE sample as a polynomial equation exclusively in the secret variables, completely eliminating the error terms algebraically.

Let the discrete Gaussian error e_i fall within a strictly bounded symmetric interval $[-t, t]$ with overwhelming probability, as done in [4, 7]. The probability of an error falling outside this bound decays exponentially, allowing the attacker to safely assume $e_i \in \{-t, \dots, t\}$. The number of possible error values is $d = 2t + 1$.

The attacker constructs an error-annihilating polynomial $P(x)$ whose roots are precisely the elements of this bounded interval:

$$P(x) = x \prod_{j=1}^t (x + j)(x - j).$$

Because the error e_i is assumed to fall within this specific range, it is then a root of this polynomial, yielding $P(e_i) \equiv 0 \pmod{q}$. By substituting the relation $e_i = b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle \pmod{q}$ into this identity, the error term is completely eliminated. This substitution yields a non-linear polynomial equation of total degree d that depends exclusively on the known public sample $(\mathbf{a}_i, b_i)$ and the unknown secret vector $\mathbf{s}$:

$$P(b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle) \equiv 0 \pmod{q}.$$

The resulting system is highly non-linear. The Arora and Ge framework resolves this via *polynomial linearization* [7]. By introducing a fresh, independent variable for every possible monomial up to the maximum occurring degree d , the non-linear system is flattened into a massive linear system.

However, a sufficient number of samples is required to obtain a full-rank matrix. A system in n variables of degree d contains $\mathcal{O}(n^d)$ monomials, dictating a sample complexity of $m = \mathcal{O}(n^d)$ [4]. As m increases, the probability of an error term falling outside $[-t, t]$ also increases, forcing the attacker to widen the interval. This increases d , thereby requiring even more samples.

While practically prohibitive for secure cryptographic bounds, this attack is a foundational theoretical result demonstrating that LWE possesses an exploitable algebraic structure.

2.4.3 The Polynomial System Solving Pipeline

The standard strategy for modern algebraic attacks against polynomial systems usually consists of three primary stages:

1. **Gröbner Basis Computation:** A Gröbner basis for the initial system is computed using efficient algorithms such as Faugère’s F4 [29] or F5 [30]. To optimize computational performance, this step is usually performed with respect to a degree-compatible ordering, typically the GREVLEX order.
2. **Change of Ordering:** Although a GREVLEX basis is highly efficient to compute, its structure does not isolate variables, making it difficult to extract solutions directly. To resolve this, a change of ordering algorithm like FGLM [31] is applied to convert the basis into LEX order. This crucial transformation reorganizes the system into a solvable triangular form.
3. **Back-Substitution:** The triangular structure of the LEX basis isolates at least one variable in a univariate polynomial. Finding its roots allows for the sequential recovery of the remaining variables via simple back-substitution.

This sequence of algebraic operations is formally illustrated in Figure 2.1.

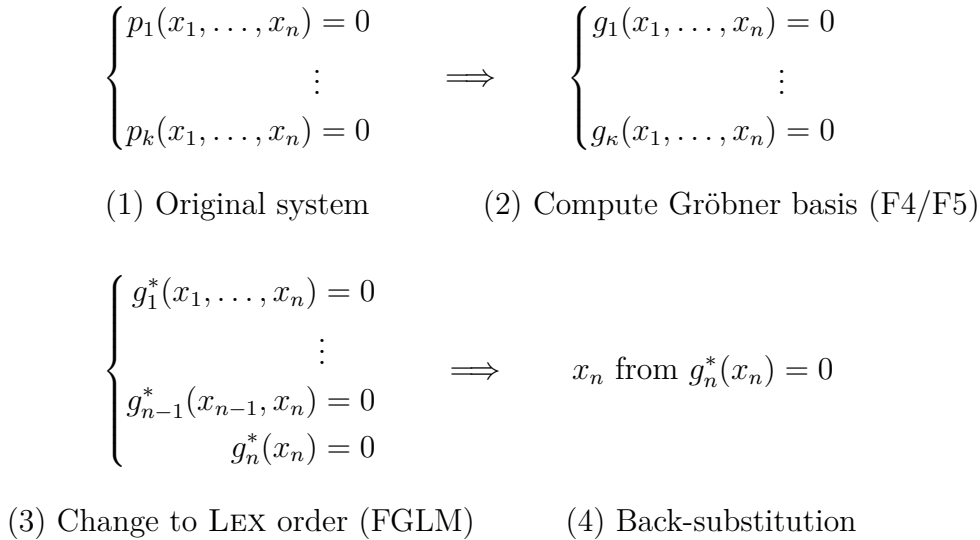


Figure 2.1: The standard PoSSo pipeline for algebraic cryptanalysis.

2.4.4 Gröbner Basis and the Degree of Regularity

To circumvent the sample complexity required by pure linearization, subsequent work focused on solving these polynomial systems using Gröbner basis algorithms, such as Faugère’s F4 [29] and F5 [30]. Gröbner basis allow attackers to find common solutions to multivariate polynomial equations even when the system is underdetermined with respect to the total number of monomials.

The time and memory complexity of computing a Gröbner basis is dominated by a central parameter known as the *degree of regularity*, denoted d_{reg} . Heuristically, d_{reg} corresponds to the highest total polynomial degree reached during the Gröbner basis reduction process before the algebraic system collapses to a basis that easily yields a solution.

Determining d_{reg} analytically for an arbitrary system is notoriously difficult. To derive precise complexity bounds, state-of-the-art estimates, such as those by Albrecht et al. [5], analyze LWE-derived systems under the heuristic assumption that they behave as *semi-regular sequences* [36]. While a rigorous algebraic treatment of semi-regularity is beyond the scope of this work, intuitively, a semi-regular sequence is a system of equations that behaves as “randomly” or “generically” as possible. In such systems, the polynomials share no hidden algebraic relations other than the trivial ones, meaning the computation will not benefit from any unexpected degree-falls.

For these generic systems, the degree of regularity can be explicitly extracted by expanding the Hilbert series of the ideal [32]. Specifically, for a semi-regular sequence of m polynomials of degree d in n variables, the Hilbert series is given by the generating function:

$$H(z) = \left[\frac{(1 - z^d)^m}{(1 - z)^n} \right]_+$$

where d_{reg} is defined as the index of the first non-positive coefficient in this power series expansion [4].

By utilizing this heuristic, it is possible to demonstrate that Gröbner basis attacks drastically reduce the sample requirement compared to pure linearization. To evaluate the precise computational cost, we must account for the noise distribution and the cost of the underlying matrix operations.

Assuming a standard LWE parameterization where the discrete Gaussian error has a standard deviation of $\sigma = \alpha q = \sqrt{n}$, an attacker can choose an optimal sample size to recover the secret with a time complexity of: $\mathcal{O}(2^{2.35\omega n + 1.13n})$ where ω represents the linear algebra constant, namely the exponent of matrix multiplication ($2 \leq \omega \leq 3$) [4].

This shifts the algebraic attack into a strictly exponential $\mathcal{O}(2^n)$ class. While it outperforms the Arora-Ge linearization technique, it still remains practically inferior to lattice reduction for standard cryptographic parameters.

2.4.5 Refined Analysis of the Degree of Regularity

Following the foundational estimates by Albrecht et al. [5], subsequent works have attempted to refine the complexity bounds of algebraic attacks. Under the same semi-regular assumption, Sun, Tibouchi, and Abe [49] provided tighter asymptotic estimates for d_{reg} by directly analyzing the associated Hilbert series.

However, the fundamental reliance on the semi-regularity assumption for LWE-derived systems remains mathematically controversial. While generic, random polynomial systems are widely expected to be semi-regular, Fröberg [32] demonstrated that there are specific classes of polynomial systems that can never achieve semi-regularity. Furthermore, Hodges et al. [36] proved that for certain parameter choices and variable counts, semi-regular sequences simply cannot exist.

To address these limitations, recent literature has sought to bypass or weaken these requirements. Steiner [48] revisited the algebraic approach by using the mathematical notions of *generic coordinates* and *solving degree*. This methodology yields an accurate estimate of the algebraic complexity that reflects the structural nuances of the LWE noise distribution. However, while this result successfully drops the strict semi-regularity assumption, it still fundamentally relies on heuristic assumptions regarding the degree of regularity itself (specifically, assuming d_{reg} to be bounded by the initial degree of the polynomials).

More recently, the work in [27] showed that d_{reg} can be considered equal to the initial degree d with high probability.

While these refinements significantly advance our theoretical understanding and bridge the gap between heuristic predictions and actual algorithmic performance, they all share a fundamental reliance on heuristic or probabilistic assumptions to successfully bound the complexity of Gröbner basis computations.

2.4.6 The Bottleneck of Existing Analyses

In essentially all Gröbner-basis-based attacks on LWE, the dominant computational cost arises from the initial computation of the Gröbner basis. Because an exact, deterministic complexity analysis of the F4 and F5 algorithms is notoriously difficult, most works rely on heuristic estimates based on the degree of regularity d_{reg} and the semi-regularity assumption.

While this methodology provides useful asymptotic predictions for random systems, it leaves a significant gap between heuristic complexity estimates and provable, concrete bounds when applied to the structured systems arising from LWE. Table 2.1 summarizes the main algebraic approaches to LWE and explicitly highlights the underlying assumptions driving their complexity analyses.

Work	No. Samples	Time Complexity	Using assumptions?	Without d_{reg} ?
[5]	$\mathcal{O}(n \log \log n)$	$\mathcal{O}\left(n^2 \cdot 2^{\frac{\omega n \log \log \log n}{8 \log \log n}}\right)$	YES	×
[5]	$c \cdot n$	$\mathcal{O}\left(n^2 \cdot 2^{\omega n(1+\beta)H_2(\beta/(1+\beta))}\right)$	YES	×
[49]	$c \cdot n^2$	$n^{\mathcal{O}(1/c)}$	YES	×
[49]	$n^{1+\alpha}$	$2^{\tilde{\mathcal{O}}(m^{1-\alpha})}$	YES	×
[48]	$\mathcal{O}(n^2)$	$\mathcal{O}\left(n^2 \cdot \binom{n+3}{3}^{\omega+2}\right)$	YES	×
[27]	$\binom{n+1}{2}$	$\mathcal{O}\left(2n^{1+2\omega}\right)$	NO	×

Table 2.1: An overview of the complexities of algebraic attacks on LWE with binary errors, a secret $\mathbf{s}$ of length n and m samples. $H_2(x) = -x \log x - (1-x) \log(1-x)$ is the binary entropy function, $\ln$ is the natural logarithm, ω is the linear algebra constant ($2 \leq \omega \leq 3$), and $\beta = c - 0.5 - \sqrt{c(c-1)}$. The last two columns indicate whether the bounds rely on unproven theoretical assumptions (such as semi-regularity or probabilistic estimates) and/or heuristic estimates of the degree of regularity d_{reg} were utilized in formulating the complexity bounds.

Motivation for a New Approach. As synthesized in Table 2.1, existing algebraic attacks on LWE share a common feature: they rely on unproven theoretical assumptions (such as semi-regularity) or on heuristic estimates of the degree of regularity. While these probabilistic frameworks provide convenient asymptotic predictions, their validity when applied to the highly structured polynomial systems such as those derived from LWE remains mathematically unproven. This reliance on heuristics creates a gap between theoretical estimations and provable, concrete complexity bounds.

Motivated by the analytical gaps highlighted in Table 2.1, the primary objective of this thesis is to introduce a novel algebraic attack on the **search**-LWE problem that fundamentally departs from standard heuristic paradigms. Unlike previous works, our goal is to design a Gröbner-basis-based solving strategy that does not rely on the degree of regularity d_{reg} , generic structural assumptions, or asymptotic heuristics.

Instead, the following chapters will present an explicit, deterministic computation of the algebraic complexity, derived exclusively from the known parameters of the LWE instance. By removing the analytical bottleneck of estimating d_{reg} , this approach aims to replace heuristics with exact, polynomial-time expressions, offering a fully transparent and rigorous perspective on the algebraic hardness of LWE.

3 | A Novel Algebraic Algorithm for LWE with Binary Errors

As established in Chapter 2, LWE instances naturally reduce to Polynomial System Solving (PoSSo) problems, making them susceptible to Gröbner basis techniques. However, as previously discussed, relying on general-purpose algorithms like F4 or F5 complicates rigorous security evaluations, since their theoretical complexity bounds depend heavily on heuristic assumptions such as the degree of regularity, and often yield loose estimates.

Motivated by these limitations, this chapter introduces a novel, exact algebraic algorithm tailored specifically for the **BinaryError-LWE** problem. Our primary objective is to design an algorithm whose computational complexity can be evaluated deterministically, entirely bypassing asymptotic heuristics. By introducing a targeted selection criterion for S -polynomial pairs, our algorithm forces a strict degree-fall mechanism that confines the entire computation to degree-two polynomials. This structural bounding allows us to derive a precise, mathematically rigorous complexity bound relying exclusively on the explicit parameters of the system.

To build the necessary intuition for this optimized methodology, Section 3.1 first explores a straightforward linear-algebraic approach. Analyzing the dimensional bottlenecks and failure points of this naive method will clearly motivate the algebraic mechanics and Gröbner basis techniques underpinning our main proposed algorithm.

3.1 From Linear Algebra to Gröbner Basis

In the specific setting of Learning With Errors with binary errors (where $e_i \in \{0, 1\}$), the error polynomial simplifies to:

$$f(x) = x(x - 1) = x^2 - x \in \mathbb{F}_q[x].$$

As previously established, rewriting the available LWE samples as polynomial equations yields constraints of degree at most two. Consequently, the resulting system consists exclusively of multivariate polynomials whose monomials have a total degree bounded by two.

In an idealized scenario, this problem reduces to a direct linear-algebraic computation. Let N denote the number of possible monomials of total degree at most two in n variables, given by:

$$N = \binom{n+2}{2}.$$

If an attacker possesses exactly N independent LWE samples, they can map these polynomials into a coefficient matrix using an ordered monomial basis. Diagonalizing this matrix via standard Gaussian elimination would explicitly reveal the linear relations, and thus the unknowns.

In Figure 3.1 and following figures, coloured blocks denote dense matrices, white blocks with a filled diagonal denote diagonal matrices, and upper-triangular coloured blocks denote matrices in Row Echelon Form (REF).

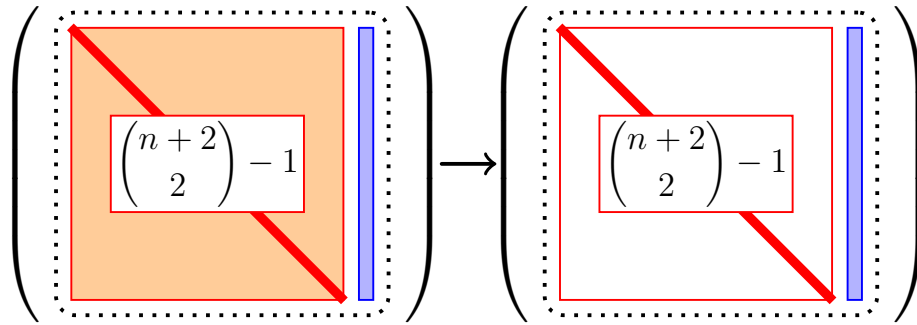


Figure 3.1: Graphical representation of the straightforward linear-algebraic approach for recovering the LWE secret vector. As a general visual convention for this and following figures: coloured blocks denote dense matrices, white blocks with a filled diagonal denote diagonal matrices, and upper-triangular blocks indicate matrices in Row Echelon Form (REF).

Strictly speaking, isolating the secret via equations of the form $s_i - c_i = 0$ does not require a perfectly square matrix. Rather, the coefficient matrix must achieve full column rank, which typically requires dimensions of exactly $(N - 1) \times N$. When an attacker has access to an abundance of samples, the full-rank picture serves as an accurate, intuitive guideline for the attack.

In practical cryptographic applications, however, the number of initially available samples is typically smaller than $\binom{n+2}{2}$. As a result, the associated matrix is rectangular with far fewer rows than columns, rendering the straightforward linear-algebraic approach heavily underdetermined.

This dimensional bottleneck is precisely where Gröbner basis techniques become essential. When the initial LWE samples are insufficient to construct a full-rank matrix, Gröbner basis algorithms systematically generate new, exact polynomial relations (most notably via S -polynomials). From a linear-algebraic perspective, this process compensates for the initial lack of equations by computing and appending new, linearly independent rows to the coefficient matrix. By

expanding the system while strictly preserving the underlying ideal, the algorithm effectively recovers the full-rank condition necessary to solve the system, without requiring additional LWE samples.

3.2 Overview of the Algorithm

Building upon the conceptual framework introduced in the previous section, we now outline the high-level mechanics of our proposed algorithm.

To provide concrete intuition, we establish a baseline sample requirement. Unlike the linear-algebraic approach, which demands full rank across all monomials up to degree two, our optimized method requires only a number of polynomial samples equal to:

$$m = \binom{n+1}{2} = \frac{n(n+1)}{2}.$$

Crucially, this value coincides exactly with the number of monomials of total degree *exactly* two in n variables.

The core of our methodology relies on fixing a graded monomial ordering and performing a targeted diagonalization on the leading principal submatrix of the associated coefficient matrix (i.e., the submatrix corresponding to the degree-two monomials). We operate under the assumption that this specific submatrix achieves full rank, a condition that holds with overwhelming probability for random LWE instances (a formal probabilistic justification is provided in Appendix A).

This initial diagonal form is highly advantageous. By strategically selecting specific pairs of polynomials from this diagonalized set, we guarantee that the computation and subsequent reduction of their S -polynomials can be executed without invoking general multivariate polynomial division. Instead, the reduction is natively absorbed into the linear-algebraic row operations. This represents a fundamental structural simplification: it allows the algorithm to operate deterministically on the polynomial set, completely circumventing unpredictable degree inflation and enabling an exact theoretical estimation of the computational complexity.

The attack proceeds systematically: following the initial diagonalization, the algorithm isolates n strategic pairs of polynomials and computes their corresponding S -polynomials. These n newly generated relations are then adjoined to the existing system. Finally, the diagonalization procedure is applied once more to this augmented matrix. Because the new S -polynomials introduce linearly independent constraints at lower degrees, this final reduction forces the system to collapse, directly revealing the explicit solution to the original LWE problem.

Figure 3.2 provides a graphical intuition of the algorithm’s core mathematical mechanism, while Figure 3.3 details a sequential overview of the execution phases.

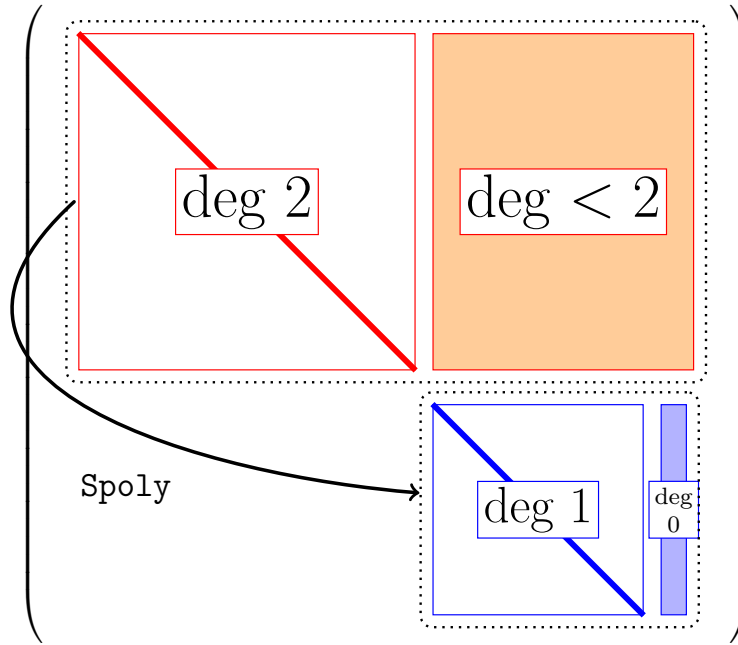


Figure 3.2: A visual overview of the proposed algorithm's core mathematical phases. Block colour conventions follow those established in Figure 3.1.

In the subsequent sections, we provide a rigorous mathematical description of each procedural step. Following the formalization of the algorithm, Section 3.7 is dedicated to a comprehensive analysis of the associated exact computational complexities.

3.3 The Algorithm

Having established the high-level intuition, we now formally detail the proposed algebraic algorithm. The execution of the algorithm can be broadly outlined in three main steps:

1. **Initial System Diagonalization:** The matrix constructed from the available LWE samples is reduced to a fully diagonalized form.
2. **S -Polynomial Generation and Reduction:** A targeted set of n new polynomials (S -polynomials) is generated. Thanks to the prior diagonalization, these can be reduced deterministically without requiring multivariate polynomial division.
3. **Final Diagonalization and Secret Recovery:** The reduced set of S -polynomials are appended to the system, and a final diagonalization reveals the explicit linear relations that yield the secret vector.

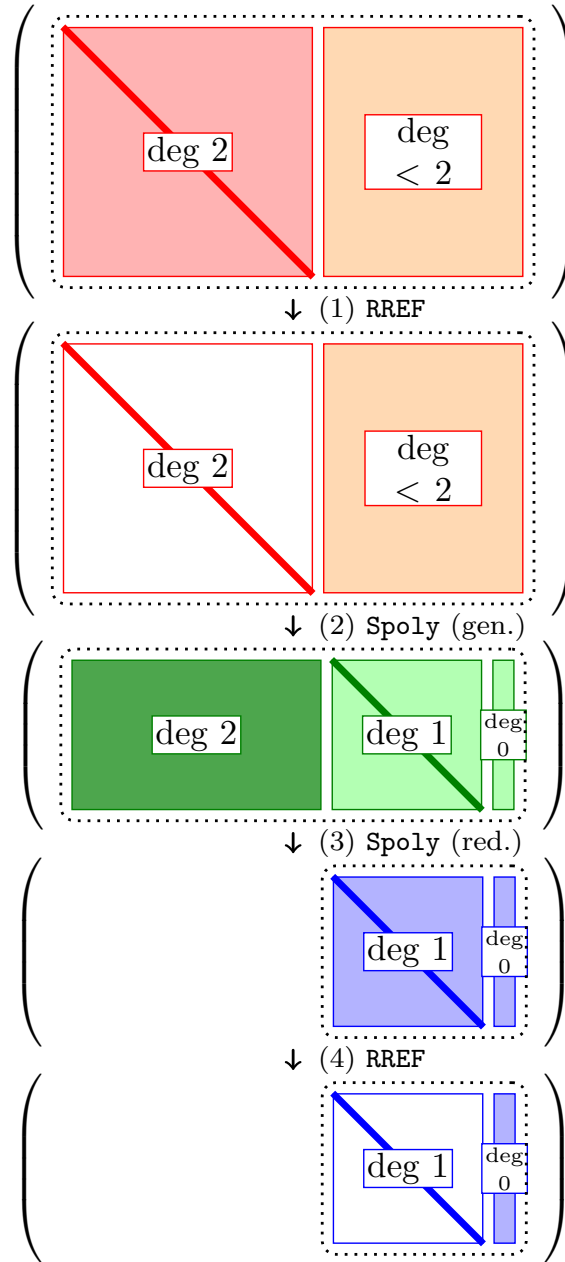


Figure 3.3: Comprehensive visual summary of the presented algorithm. The diagram illustrates four main steps of the workflow: (1) The initial coefficient matrix representing the polynomial system is transformed into Reduced Row-Echelon Form (RREF). (2) A targeted set of n S -polynomials is generated based on the diagonalized structure. (3) These polynomials are deterministically reduced against the system basis to yield linear relations. (4) A final RREF step is performed on the augmented system to explicitly recover the secret vector $\mathbf{s}$. Block colour conventions follow those established in Figure 3.1.

For the sake of clarity, particularly to facilitate the complexity analysis presented in Section 3.7, it is highly convenient to decompose the diagonalization processes (used in both the first and third steps) into two distinct linear-algebraic subroutines: Forward Elimination and Backward Substitution, also called **DiagA** and **DiagB** respectively.

3.3.1 Phase 1: Initial System Diagonalization

The initial diagonalization operates on the coefficient matrix constructed from the m available polynomial samples. This step relies strictly on the linear independence of the rows, allowing us to isolate specific monomials. We divide this phase into two distinct operational steps.

Step 1A: Forward Elimination (Reduction to REF).

The first step consists of reducing the coefficient matrix to a Row-Echelon Form (REF).

The procedure consists of iteratively selecting a pivot (which corresponds to the leading monomial of the current polynomial under the chosen graded ordering) and eliminating this pivot from all subsequent rows via elementary row operations.

Prior to the elimination step, the pivot row is normalized by the inverse of its leading coefficient to ensure the pivot becomes monic, thus preserving the algebraic integrity of the polynomial ideal over $\mathbb{F}_q$. This process is executed sequentially, progressing from the top row downward, until all matrix entries strictly below the main diagonal are annihilated.

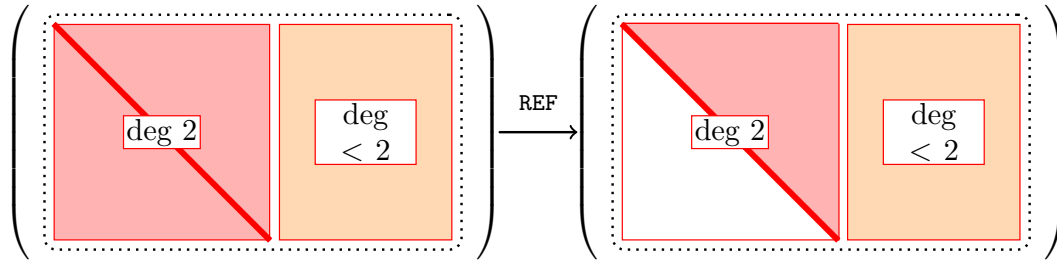


Figure 3.4: Overview of the Forward Elimination step, reducing the initial system to Row-Echelon Form. Block colour conventions follow those established in Figure 3.1.

Step 1B: Backward Substitution (Reduction to RREF).

The second step completes the diagonalization by transforming the REF matrix into Reduced Row-Echelon Form (RREF).

This procedure is computationally analogous to the first step but is executed in the reverse direction. Starting from the bottom-most non-zero row, the algorithm selects the monic pivots in reverse order. Each pivot is then used to eliminate the corresponding non-zero entries from all rows strictly *above* it via linear combinations.

At the conclusion of this backward step, all elements above the main diagonal corresponding to the pivot columns have been removed. The combination of the forward and backward elimination steps produces a fully diagonalized leading principal submatrix. This RREF structure is mathematically vital, as it allows for the immediate, division-free reduction of the S -polynomials generated in the subsequent phase.

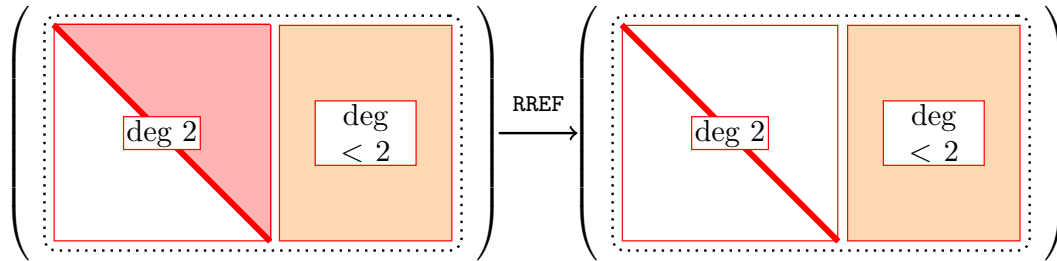


Figure 3.5: Overview of the Backward Substitution step, yielding the fully diagonalized leading submatrix. Block colour conventions follow those established in Figure 3.1.

3.3.2 Phase 2: S-Polynomial Generation and Reduction

In a standard Gröbner basis algorithm, computing S -polynomials for all possible pairs inevitably leads to unpredictable degree inflation. To strictly bound the complexity, our algorithm selectively computes the S -polynomial of two polynomials f and g if and only if the greatest common divisor of their leading terms has exactly degree one:

$$\deg(\gcd(\text{LT}(f), \text{LT}(g))) = 1.$$

This targeted selection guarantees that the corresponding S -polynomial takes a highly controlled form. Because the initial system consists of polynomials of total degree two, their leading terms $\text{LT}(f)$ and $\text{LT}(g)$ are quadratic. Under the condition above, their least common multiple, $\text{lcm}(\text{LM}(f), \text{LM}(g))$, has exactly degree three. Consequently, the multiplier terms used in the S -polynomial definition,

$$\frac{\text{lcm}(\text{LM}(f), \text{LM}(g))}{\text{LT}(f)} \quad \text{and} \quad \frac{\text{lcm}(\text{LM}(f), \text{LM}(g))}{\text{LT}(g)},$$

are strictly of degree one.

This is where the structure of the Reduced Row Echelon Form becomes crucial. Because f and g are selected from the diagonalized system (according to the given criterion), they are monic and their *only* degree-two terms are their respective leading monomials. We can thus decompose them as:

$$f = \text{LM}(f) + f_{\leq 1} \quad \text{and} \quad g = \text{LM}(g) + g_{\leq 1},$$

where $f_{\leq 1}$ and $g_{\leq 1}$ consist entirely of monomials of degree at most one. When we multiply f and g by their respective degree-one multipliers (e.g., individual variables x_u and x_v), the only terms that momentarily reach degree three are $x_u \cdot \text{LM}(f)$ and $x_v \cdot \text{LM}(g)$. By the definition of the S -polynomial, these degree-three leading terms perfectly cancel each other out:

$$\begin{aligned} S(f, g) &= x_u f - x_v g \\ &= x_u (\text{LM}(f) + f_{\leq 1}) - x_v (\text{LM}(g) + g_{\leq 1}) \\ &= x_u f_{\leq 1} - x_v g_{\leq 1}. \end{aligned}$$

Since $f_{\leq 1}$ and $g_{\leq 1}$ have degree at most one, their multiplication by x_u and x_v guarantees that the resulting S -polynomial drops back to a total degree of at most two. For a concrete example of this generation step, we refer the reader to Step 4 of the comprehensive toy example in Section 3.4.

Once this degree-two S -polynomial is formed, the power of the initial diagonalization becomes clear once again. Because the leading monomials of the diagonalized sample matrix span the *entire* set of degree-two monomials in n variables, every newly generated degree-two S -polynomial can be completely reduced with respect to this basis. This reduction process does not require multivariate polynomial division: it is achieved entirely via deterministic linear combinations (row operations) against the diagonal form. This structural property completely eliminates all degree-two terms from the S -polynomials, yielding purely *linear* relations.

To recover the secret, we require n such linear relations. We can systematically extract n distinct S -polynomials by selecting specific pairs whose leading terms satisfy the GCD constraint. Let $x_1, \dots, x_n$ denote the variables, and consider the indices cyclically (setting $i + 1 = 1$ whenever $i = n$). For each $i \in \{1, \dots, n\}$, we select the pair of polynomials whose respective leading monomials are:

$$(x_i^2, x_i x_{i+1}).$$

These pairs are distinctly defined and involve distinct monomials. It is trivial to verify that

$$\gcd(x_i^2, x_i x_{i+1}) = x_i,$$

which satisfies the degree-one requirement.

Feasibility Condition ($n \geq 3$). To successfully extract the secret, the algorithm requires exactly n distinct S -polynomials. To prevent redundant relations, no initial polynomial should be reused across the n computed S -polynomial pairs. Therefore, the algorithm strictly requires $2n$ distinct polynomials from the diagonalized set. Since our diagonalized set provides exactly $\binom{n+1}{2}$ independent polynomials, the algebraic feasibility condition to form enough disjoint pairs is:

$$\binom{n+1}{2} \geq 2n.$$

Expanding the binomial coefficient yields:

$$\frac{n(n+1)}{2} \geq 2n.$$

Since the dimension n is strictly positive, we obtain the relation:

$$n+1 \geq 4 \implies n \geq 3.$$

Furthermore, the successful recovery of the secret vector relies on the condition that the n generated linear relations are themselves linearly independent. Similarly to the initial sample matrix, this property is expected to hold with the same overwhelming probability discussed in Appendix A.

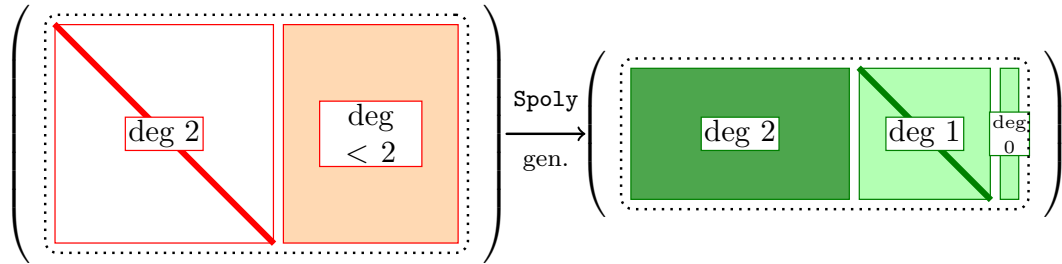


Figure 3.6: Overview of targeted S -polynomial generation.

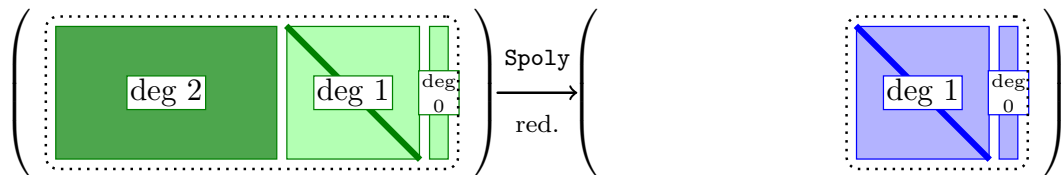


Figure 3.7: Overview of the division-free S -polynomial reduction against the diagonal system. Block colour conventions follow those established in Figure 3.1.

3.3.3 Phase 3: Final Diagonalization and Secret Recovery

Once the n targeted S -polynomials have been generated and reduced against the degree-two diagonal form, we are left with exactly n linearly independent polynomials of degree one.

We append this linear subsystem to our coefficient matrix. To extract the secret, we proceed with a final diagonalization step, which coincides precisely with performing standard Gaussian elimination (final REF and RREF steps) strictly on the linear and constant columns of the augmented matrix.

Because we have essentially isolated a full-rank linear system of n equations in n unknowns, this final reduction explicitly solves for the variables. The resulting system trivially simplifies to the canonical form:

$$x_i - c_i = 0, \quad \text{for } i = 1, \dots, n.$$

The constants $c_i \in \mathbb{F}_q$ derived from this final reduced row-echelon form constitute the exact components of the secret vector $\mathbf{s}$, effectively terminating the attack and recovering the LWE secret.

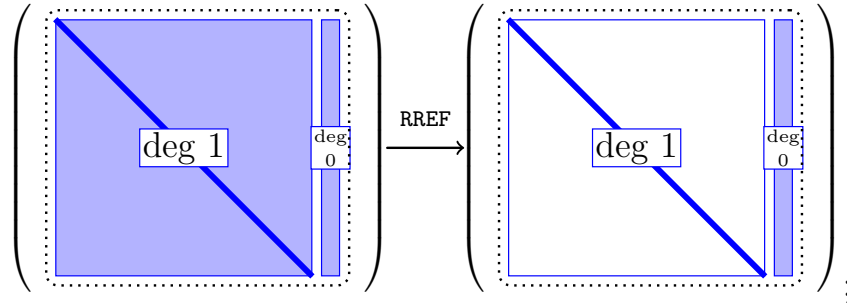


Figure 3.8: Overview of the final diagonalization. The linear subsystem undergoes final REF and RREF steps to explicitly recover the secret vector $\mathbf{s}$. Block colour conventions follow those established in Figure 3.1.

3.4 A Comprehensive Toy Example

To concretely illustrate the complete execution of the proposed algorithm, we walk through a highly scaled-down, end-to-end instance of the LWE problem. We establish the parameters $n = 3$ and $q = 11$. Suppose the unknown secret vector is

$$\mathbf{s} = (7, 2, 10) \in (\mathbb{F}_{11})^3$$

and the binary error vector is

$$\mathbf{e} = (1, 0, 1, 0, 1, 0)^\top \in \{0, 1\}^6.$$

We are given $m = \frac{n(n+1)}{2} = 6$ samples, represented by the public matrix A :

$$A = \begin{pmatrix} 2 & 3 & 5 \\ 4 & 6 & 7 \\ 8 & 9 & 10 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \\ 5 & 2 & 3 \end{pmatrix} \in (\mathbb{F}_{11})^{6 \times 3}.$$

We compute the corresponding target vector $\mathbf{b} = A\mathbf{s} + \mathbf{e} \pmod{11}$:

$$\mathbf{b} = \begin{pmatrix} 5 \\ 0 \\ 10 \\ 1 \\ 2 \\ 3 \end{pmatrix} \in (\mathbb{F}_{11})^{6 \times 1}.$$

Step 1: System Generation. For each LWE sample $(\mathbf{a}_i, b_i)$, we define the linear relation:

$$T_i := b_i - \langle \mathbf{a}_i, \mathbf{x} \rangle.$$

By substituting this into the binary error polynomial $f(x) = x^2 - x$, we generate the initial equations:

$$f_i(\mathbf{x}) = f_i(T_i) = T_i^2 - T_i \equiv 0 \pmod{11}.$$

For completeness, let us illustrate the computations for the first sample in full detail. The first row is $\mathbf{a}_1 = (2, 3, 5)$ and $b_1 = 5$, we have:

$$T_1 = 5 - 2x_1 - 3x_2 - 5x_3.$$

Expanding $f_1(\mathbf{x}) = T_1^2 - T_1$ and strictly reducing all coefficients modulo 11 yields the quadratic polynomial:

$$f_1 = 4x_1^2 + x_1x_2 + 9x_1x_3 + 9x_2^2 + 8x_2x_3 + 3x_3^2 + 4x_1 + 6x_2 + 10x_3 + 9.$$

Applying this identical construction to all six samples yields the initial polynomial system $\mathcal{F}_{\text{LWE}} = \{f_1, \dots, f_6\}$:

$$\begin{aligned} f_1 &= 4x_1^2 + x_1x_2 + 9x_1x_3 + 9x_2^2 + 8x_2x_3 + 3x_3^2 + 4x_1 + 6x_2 + 10x_3 + 9, \\ f_2 &= 5x_1^2 + 4x_1x_2 + x_1x_3 + 3x_2^2 + 7x_2x_3 + 5x_3^2 + 4x_1 + 6x_2 + 7x_3 + 0, \\ f_3 &= 9x_1^2 + x_1x_2 + 6x_1x_3 + 4x_2^2 + 4x_2x_3 + x_3^2 + 2x_1 + 5x_2 + 8x_3 + 2, \\ f_4 &= 9x_1^2 + 2x_1x_2 + 3x_1x_3 + 5x_2^2 + 4x_2x_3 + 3x_3^2 + 8x_1 + 7x_2 + 5x_3 + 0, \\ f_5 &= 5x_1^2 + 2x_1x_2 + 5x_1x_3 + 9x_2^2 + x_2x_3 + 4x_3^2 + x_1 + 9x_2 + 6x_3 + 2, \\ f_6 &= 3x_1^2 + 9x_1x_2 + 8x_1x_3 + 4x_2^2 + x_2x_3 + 9x_3^2 + 8x_1 + x_2 + 7x_3 + 6. \end{aligned}$$

Step 2: Phase 1A (Forward Elimination to REF). We fix the graded reverse lexicographic ordering $x_1 \succ x_2 \succ x_3$, ordering the monomials as:

$$x_1^2 \succ x_1x_2 \succ x_1x_3 \succ x_2^2 \succ x_2x_3 \succ x_3^2 \succ x_1 \succ x_2 \succ x_3 \succ 1.$$

We represent the polynomial system using the coefficient matrix M_{init} :

$$M_{\text{init}} = \begin{pmatrix} 4 & 1 & 9 & 9 & 8 & 3 & 4 & 6 & 10 & 9 \\ 5 & 4 & 1 & 3 & 7 & 5 & 4 & 6 & 7 & 0 \\ 9 & 1 & 6 & 4 & 4 & 1 & 2 & 5 & 8 & 2 \\ 9 & 2 & 3 & 5 & 4 & 3 & 8 & 7 & 5 & 0 \\ 5 & 2 & 5 & 9 & 1 & 4 & 1 & 9 & 6 & 2 \\ 3 & 9 & 8 & 4 & 1 & 9 & 8 & 1 & 7 & 6 \end{pmatrix}.$$

We perform the Forward Elimination step over $\mathbb{F}_{11}$. The goal of this step is to iteratively eliminate the leading monomials from the rows below each pivot, producing a triangular structure. The resulting Row-Echelon Form matrix is:

$$M_{\text{REF}} = \begin{pmatrix} 1 & 0 & 6 & 10 & 8 & 4 & 4 & 2 & 4 & 3 \\ 0 & 1 & 7 & 2 & 9 & 9 & 10 & 9 & 5 & 8 \\ 0 & 0 & 1 & 0 & 7 & 9 & 6 & 9 & 0 & 7 \\ 0 & 0 & 0 & 1 & 5 & 5 & 10 & 5 & 8 & 6 \\ 0 & 0 & 0 & 0 & 1 & 3 & 7 & 7 & 6 & 8 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 10 & 5 & 7 \end{pmatrix}.$$

and the corresponding polynomial system takes the form:

$$\begin{aligned} \tilde{f}_1 &= x_1^2 + 6x_1x_3 + 10x_2^2 + 8x_2x_3 + 4x_3^2 + 4x_1 + 2x_2 + 4x_3 + 3, \\ \tilde{f}_2 &= x_1x_2 + 7x_1x_3 + 2x_2^2 + 9x_2x_3 + 9x_3^2 + 10x_1 + 9x_2 + 5x_3 + 8, \\ \tilde{f}_3 &= x_1x_3 + 7x_2x_3 + 9x_3^2 + 6x_1 + 9x_2 + 7, \\ \tilde{f}_4 &= x_2^2 + 5x_2x_3 + 5x_3^2 + 10x_1 + 5x_2 + 8x_3 + 6, \\ \tilde{f}_5 &= x_2x_3 + 3x_3^2 + 7x_1 + 7x_2 + 6x_3 + 8, \\ \tilde{f}_6 &= x_3^2 + 3x_1 + 10x_2 + 5x_3 + 7. \end{aligned}$$

Step 3: Phase 1B (Backward Substitution to RREF). Operating on M_{REF} , we perform the Backward Substitution step. This eliminates all entries strictly above the monic pivots in the quadratic block, yielding the fully diagonalized matrix M_{RREF} :

$$M_{\text{RREF}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 4 \\ 0 & 1 & 0 & 0 & 0 & 0 & 7 & 9 & 2 & 9 \\ 0 & 0 & 1 & 0 & 0 & 0 & 4 & 3 & 7 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 5 & 4 & 6 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 9 & 10 & 2 & 9 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 10 & 5 & 7 \end{pmatrix}.$$

This matrix directly translates to our diagonalized basis of quadratic polynomials, which we denote as $\{\tilde{f}'_1, \dots, \tilde{f}'_6\}$:

$$\begin{aligned}\tilde{f}'_1 &= \underline{x_1^2} + (9x_3 + 4), \\ \tilde{f}'_2 &= \underline{x_1x_2} + (7x_1 + 9x_2 + 2x_3 + 9), \\ \tilde{f}'_3 &= \underline{x_1x_3} + (4x_1 + 3x_2 + 7x_3 + 2), \\ \tilde{f}'_4 &= \underline{x_2^2} + (5x_1 + 4x_2 + 6x_3 + 3), \\ \tilde{f}'_5 &= \underline{x_2x_3} + (9x_1 + 10x_2 + 2x_3 + 9), \\ \tilde{f}'_6 &= \underline{x_3^2} + (3x_1 + 10x_2 + 5x_3 + 7).\end{aligned}$$

Step 4: Phase 2 (S -Polynomial Generation and Reduction). We construct $n = 3$ distinct S -polynomials using the pairs (x_1^2, x_1x_2) , (x_2^2, x_2x_3) , and (x_3^2, x_1x_3) . Because the diagonalized basis spans all quadratic monomials, these S -polynomials are deterministically reduced into purely linear relations via elementary combinations:

1. **Pair $(\tilde{f}'_1, \tilde{f}'_2)$ with $\text{lcm} = x_1^2x_2$:**

$$\begin{aligned}S(\tilde{f}'_1, \tilde{f}'_2) &= x_2\tilde{f}'_1 - x_1\tilde{f}'_2 \\ &\equiv 4x_1^2 + 2x_1x_2 + 9x_1x_3 + 9x_2x_3 + 2x_1 + 4x_2 \pmod{11}.\end{aligned}$$

We strictly eliminate the degree-two terms by subtracting the linear combination:

$$4\tilde{f}'_1 + 2\tilde{f}'_2 + 9\tilde{f}'_3 + 9\tilde{f}'_5.$$

This yields the purely linear relation:

$$g_1 = 3x_1 + x_2 + 10.$$

2. **Pair $(\tilde{f}'_4, \tilde{f}'_5)$ with $\text{lcm} = x_2^2x_3$:**

$$\begin{aligned}S(\tilde{f}'_4, \tilde{f}'_5) &= x_3\tilde{f}'_4 - x_2\tilde{f}'_5 \\ &\equiv x_2^2 + 2x_2x_3 + 5x_1x_3 + 6x_3^2 + 2x_2 + 3x_3 \pmod{11}.\end{aligned}$$

We eliminate the quadratic terms via subtraction of:

$$2\tilde{f}'_2 + 5\tilde{f}'_3 + \tilde{f}'_4 + 2\tilde{f}'_5 + 6\tilde{f}'_6.$$

This yields the second linear relation:

$$g_2 = 2x_1 + 6x_2 + x_3 + 8.$$

3. **Pair** $(\tilde{f}'_6, \tilde{f}'_3)$ **with** $\text{lcm} = x_1x_3^2$:

$$\begin{aligned} S(\tilde{f}'_6, \tilde{f}'_3) &= x_1\tilde{f}'_6 - x_3\tilde{f}'_3 \\ &\equiv 3x_1^2 + 10x_1x_2 + x_1x_3 + 8x_2x_3 + 4x_3^2 + 7x_1 + 9x_3 \pmod{11}. \end{aligned}$$

We eliminate the quadratics via subtraction of:

$$3\tilde{f}'_1 + 10\tilde{f}'_2 + \tilde{f}'_3 + 8\tilde{f}'_5 + 4\tilde{f}'_6.$$

This yields the final linear relation:

$$g_3 = 3x_1 + 7x_2 + 7x_3 + 5.$$

Step 5: Phase 3 (Final Diagonalization and Recovery). Collecting the newly generated linear relations gives the following 3×4 linear subsystem:

$$\begin{aligned} g_1 &= 3x_1 + x_2 + 10 \\ g_2 &= 2x_1 + 6x_2 + x_3 + 8 \\ g_3 &= 3x_1 + 7x_2 + 7x_3 + 5 \end{aligned}$$

We now perform a final diagonalization step on this augmented subsystem. Since the system is purely linear, this corresponds to a standard Gaussian elimination over $\mathbb{F}_{11}$. This yields a triangular canonical form:

$$\begin{aligned} x_1 + 4 &= 0 \implies x_1 \equiv 7 \pmod{11} \\ x_2 + 9 &= 0 \implies x_2 \equiv 2 \pmod{11} \\ x_3 + 1 &= 0 \implies x_3 \equiv 10 \pmod{11} \end{aligned}$$

From this reduced form, the variables can be recovered directly. The original LWE secret is successfully recovered exactly:

$$\boxed{\mathbf{s} = (7, 2, 10)}.$$

3.5 Pseudocode

Having introduced the mathematical mechanics underlying our approach, we now present the high-level algorithmic framework. To facilitate both theoretical analysis and practical implementation, the algorithm is modularized into distinct sub-routines: **DiagA**, **DiagB**, and **Spoly**.

These subroutines directly map to the conceptual phases discussed in Section 3.3. Specifically, **DiagA** handles the forward elimination to REF, **DiagB** performs the backward substitution to RREF, and **Spoly** executes the targeted generation and deterministic reduction of the S -polynomials.

Algorithm 3 provides the overarching pseudocode that orchestrates these sub-routines to solve the LWE instance.

This high-level description demonstrates how the algorithm progressively reduces a dense quadratic polynomial system derived from LWE samples into a fully diagonalized linear system. By strictly controlling the reduction process through these three deterministic subroutines, we prevent the unpredictable degree inflation that plagues standard Gröbner basis algorithms.

Algorithm 3 Proposed Algorithm

Require: A system of quadratic polynomials $\mathcal{F} = \{f_1, \dots, f_m\}$ over $\mathbb{F}_q$, derived from $m = \frac{n(n+1)}{2}$ LWE samples.

Ensure: The recovered secret vector $\mathbf{s} \in \mathbb{F}_q^n$.

```

1: /* Phase 1: Initial System Diagonalization */
2:  $\mathcal{F}_{\text{REF}} \leftarrow \text{DiagA}(\mathcal{F})$  ▷ Forward elimination to REF
3:  $\mathcal{F}_{\text{RREF}} \leftarrow \text{DiagB}(\mathcal{F}_{\text{REF}})$  ▷ Backward substitution to RREF (Diagonal Form)

4: /* Phase 2: Targeted  $S$ -Polynomial Generation and Reduction */
5:  $\mathcal{L} \leftarrow \text{Spoly}(\mathcal{F}_{\text{RREF}})$  ▷ Generates and reduces to  $n$  linear relations

6: /* Phase 3: Final Diagonalization and Secret Recovery */
7:  $\mathcal{L}_{\text{final}} \leftarrow \text{DiagA}(\mathcal{L})$  ▷ Forward elimination on the linear subsystem
8:  $\mathcal{L}_{\text{diag}} \leftarrow \text{DiagB}(\mathcal{L}_{\text{final}})$  ▷ Yields linear relations  $x_i - c_i = 0$ 
9: Extract the constants  $(c_1, \dots, c_n)$  from  $\mathcal{L}_{\text{diag}}$ .
10: return  $\mathbf{s} = (c_1, \dots, c_n)$ 
    
```

We will detail the precise inner workings, linear-algebraic bounds, and implementation specifics of `DiagA`, `DiagB`, and `Spoly` in the subsequent sections. Furthermore, a complete, executable implementation of this procedure written in MAGMA is provided in Appendix B for reproducibility and practical verification.

3.6 Algorithm Termination and Correctness

In this section, we formally establish the termination and correctness of the proposed algebraic algorithm. We prove that the algorithm strictly executes in a finite number of steps and deterministically recovers the unique LWE secret vector $\mathbf{s}$.

Termination

Termination is guaranteed by the finite algebraic structure of the algorithm's discrete phases:

- Phase 1 (**DiagA** and **DiagB**) performs standard Gaussian elimination on a system of m polynomials. Over the finite field $\mathbb{F}_q$, this requires a strictly bounded sequence of elementary row operations.
- Phase 2 (**Spoly**) deterministically computes n targeted S -polynomials. Each is subsequently reduced against the finite degree-two basis via a bounded number of linear combinations.
- Phase 3 (**DiagA** and **DiagB**) applies a final Gaussian elimination step on a strictly bounded linear subsystem of size $n \times (n + 1)$.

Since every step avoids arbitrary polynomial division and relies exclusively on a bounded number of finite matrix operations, the algorithm inherently and deterministically terminates.

Correctness

Let $\mathbf{s} \in \mathbb{F}_q^n$ denote the true secret vector, which by definition is a common root of the initial polynomial system $\mathcal{F}_{\text{LWE}}$ derived from the LWE samples. To prove correctness, we must show two things: first, that algebraic manipulations strictly preserve the underlying ideal (so $\mathbf{s}$ remains a valid solution), and second, that the algorithm isolates $\mathbf{s}$ as the *unique* solution.

1. **Initial Diagonalization:** These subroutines consist entirely of elementary row operations (linear combinations) on the coefficient matrix representing $\mathcal{F}_{\text{LWE}}$. It is a fundamental property of polynomial rings that linear combinations of polynomials generate elements within the same algebraic ideal, $\langle \mathcal{F}_{\text{LWE}} \rangle$. Consequently, the zero locus is invariant, and $\mathbf{s}$ continues to satisfy all polynomials in the diagonalized basis $\mathcal{F}_{\text{REF}}$.
2. **S -Polynomial Generation and Reduction:** An S -polynomial, by definition, is constructed via the addition of two polynomials multiplied by specific monomial terms. Because multiplication by a monomial and subsequent addition keep the resulting polynomial securely within the ideal $\langle \mathcal{F}_{\text{LWE}} \rangle$, the secret $\mathbf{s}$ perfectly satisfies the newly generated relations. Furthermore, the deterministic reduction against the diagonalized basis $\mathcal{F}_{\text{REF}}$ operates strictly via linear combinations, preserving $\mathbf{s}$ as a root while successfully forcing the system to collapse into n distinct linear relations.
3. **Final Diagonalization and Final Recovery:** The final step consists of re-applying **DiagA** and **DiagB** to the augmented linear subsystem $\mathcal{L}$. As before, these linear operations do not alter the solution set. Crucially, because the cyclic S -polynomial generation yields n linearly independent relations in n unknowns, the coefficient matrix of $\mathcal{L}$ inherently achieves full rank.

Because a full-rank linear system of n equations in n variables over $\mathbb{F}_q$ has exactly one unique solution, the final triangular system must simplify to the canonical form $x_i - c_i = 0$ for $i = 1, \dots, n$. Therefore, the extracted constants strictly correspond to the original secret vector, yielding $\mathbf{s} = (c_1, \dots, c_n)$ and concluding the proof of correctness.

3.7 Complexity Analysis

In this section, we provide a rigorous, exact complexity analysis of the proposed algebraic algorithm. We evaluate the computational cost of each procedural phase, measuring the complexity in terms of the total number of elementary field operations (additions and multiplications) over $\mathbb{F}_q$.

Throughout this analysis, let n denote the dimension of the LWE secret vector. The number of variables is n , and we assume that the polynomials involved have a total degree of at most 2. The total number of monomials of degree *at most* two in n variables is exactly

$$\binom{n+2}{2}.$$

The algorithm operates on an initial coefficient matrix constructed from independent LWE polynomial samples. As established, the number of samples (and thus the number of rows of the matrix) is exactly

$$\binom{n+1}{2}.$$

3.7.1 Phase 1: Initial System Diagonalization

The first phase consists of reducing the initial coefficient matrix into the degree-two diagonal form. We analyze the forward elimination and backward substitution steps independently.

Theorem 3.7.1 (Complexity of Forward Elimination). *The forward elimination step (reduction to Row-Echelon Form) requires at most*

$$\mathcal{C}_{\text{REF}} = \mathcal{O}(n^6)$$

field operations.

Proof. We estimate the exact worst-case cost of the elimination procedure.

Normalization of the pivot. Each pivot polynomial contains at most $\binom{n+2}{2}$ non-zero coefficients. To make the pivot monic, all coefficients in the row are multiplied by the modular inverse of the leading coefficient. This requires at most $\binom{n+2}{2}$ field multiplications.

Elimination below the pivot. For a given pivot row j , where $1 \leq j \leq \binom{n+1}{2}$, there are exactly $\binom{n+1}{2} - j$ rows below it. To eliminate the leading entry from each of these rows, we:

- multiply the normalized pivot row by a scalar (at most $\binom{n+2}{2}$ field multiplications),
- subtract the result from the target row (at most $\binom{n+2}{2}$ additions).

Thus, the elimination for a single target row costs at most $2\binom{n+2}{2}$ operations. For pivot j , clearing all rows below it requires:

$$2\binom{n+2}{2} \left(\binom{n+1}{2} - j \right)$$

operations.

Total cost. Summing the normalization and elimination costs over all pivots $j = 1, \dots, \binom{n+1}{2}$, we obtain the total number of field operations:

$$\begin{aligned} \mathcal{C}_{\text{REF}} &= \sum_{j=1}^{\binom{n+1}{2}} \left[\binom{n+2}{2} + 2\binom{n+2}{2} \left(\binom{n+1}{2} - j \right) \right] \\ &= \binom{n+2}{2} \sum_{j=1}^{\binom{n+1}{2}} \left[1 + 2 \left(\binom{n+1}{2} - j \right) \right] \\ &= \binom{n+2}{2} \sum_{j=1}^{\binom{n+1}{2}} \left[1 + 2\binom{n+1}{2} - 2j \right] \\ &\stackrel{*}{=} \binom{n+2}{2} \binom{n+1}{2} + 2\binom{n+2}{2} \binom{n+1}{2}^2 - \\ &\quad - \frac{2}{2} \binom{n+2}{2} \binom{n+1}{2} \left(\binom{n+1}{2} + 1 \right) \\ &= \binom{n+2}{2} \binom{n+1}{2} + \binom{n+2}{2} \binom{n+1}{2}^2 - \binom{n+2}{2} \binom{n+1}{2} \\ &= \binom{n+2}{2} \binom{n+1}{2}^2 \end{aligned}$$

where in $\stackrel{*}{=}$ we have used the relation

$$\sum_{\ell=1}^k \ell = \frac{k(k+1)}{2}.$$

Using the standard combinatorial identity

$$\binom{n+2}{2} = \frac{(n+2)}{n} \binom{n+1}{2},$$

the total number of field operations is strictly bounded by:

$$\begin{aligned} \mathcal{C}_{\text{REF}} &= \mathcal{O} \left(\frac{(n+2)}{n} \binom{n+1}{2}^3 \right) \\ &= \mathcal{O} \left(\frac{(n+2)}{n} \left(\frac{(n+1)n}{2} \right)^3 \right) \\ &= \mathcal{O} \left(\frac{(n+2)(n+1)^3 n^2}{8} \right) \\ &= \mathcal{O}(n^6). \end{aligned}$$

□

Theorem 3.7.2 (Complexity of Backward Substitution). *The backward substitution step (reduction to Reduced Row-Echelon Form) requires at most*

$$\mathcal{C}_{\text{RREF}} = \mathcal{O}(n^5)$$

field operations.

Proof. We evaluate the cost of eliminating the entries strictly above the main diagonal of the quadratic block.

Effective row size. A crucial structural advantage of performing backward substitution *after* forward elimination is that all quadratic terms below the main diagonal have been annihilated. Furthermore, the pivot row j is monic in a single quadratic monomial, and its remainder consists entirely of linear terms and a constant. Therefore, the effective number of non-zero coefficients in any pivot row is at most:

$$\binom{n+1}{1} + 1 = n + 2.$$

This number corresponds to the n linear variables plus the constant, alongside the quadratic leading term.

Elimination above the pivot. Because the pivots are already monic after the forward pass, no normalization step is required. We process the pivots in reverse order (from bottom to top). Let j denote the step index of the backward elimination, where $j = 1$ corresponds to the bottom-most row and $j = \binom{n+1}{2}$ corresponds to the top-most row.

At step j , the current pivot has exactly

$$\binom{n+1}{2} - j$$

rows above it that have not yet been cleared in the current column. To eliminate the entry in the pivot column for one such target row, we:

- multiply the pivot row by a scalar (at most $n + 2$ multiplications),
- subtract the scaled row from the target row (at most $n + 2$ additions).

Hence, clearing one target row requires exactly $2(n + 2)$ operations. For the pivot processed at step j , clearing all rows above it costs:

$$2(n + 2) \left(\binom{n+1}{2} - j \right)$$

operations.

Total cost. Summing this cost over all $\binom{n+1}{2}$ pivots, we explicitly evaluate the total number of operations:

$$\begin{aligned} \mathcal{C}_{\text{RREF}} &= \sum_{j=1}^{\binom{n+1}{2}} 2(n + 2) \left(\binom{n+1}{2} - j \right) \\ &= 2(n + 2) \left[\sum_{j=1}^{\binom{n+1}{2}} \binom{n+1}{2} - \sum_{j=1}^{\binom{n+1}{2}} j \right] \\ &= 2(n + 2) \left[\binom{n+1}{2}^2 - \frac{1}{2} \binom{n+1}{2} \left(\binom{n+1}{2} + 1 \right) \right] \\ &= (n + 2) \left[\binom{n+1}{2}^2 - \binom{n+1}{2} \right]. \end{aligned}$$

To determine the precise asymptotic complexity, we explicitly substitute the binomial coefficient $\binom{n+1}{2} = \frac{n(n+1)}{2}$ into the derived expression:

$$\begin{aligned} \mathcal{C}_{\text{RREF}} &= \mathcal{O} \left((n + 2) \left[\left(\frac{n(n+1)}{2} \right)^2 - \frac{n(n+1)}{2} \right] \right) \\ &= \mathcal{O} \left((n + 2) \left(\frac{n(n+1)}{2} \right)^2 \right) \\ &= \mathcal{O} \left(\frac{(n + 2)n^2(n+1)^2}{4} \right) \\ &= \mathcal{O} \left(n^5 \right). \end{aligned}$$

□

3.7.2 Phase 2: S-Polynomial Generation and Reduction

We now analyze the computational cost of forming and deterministically reducing the n required S -polynomials. To maintain consistency with the preceding analysis, all computations are performed over the finite field $\mathbb{F}_q$, and complexity is measured in the total number of elementary field operations (both additions and multiplications).

We rely on the structural combinatorial quantities established in Phase 1. Specifically, the number of quadratic monomials (and thus the number of rows in the diagonalized basis) is $\binom{n+1}{2}$. Following the backward substitution pass, each diagonalized pivot row contains at most

$$\binom{n+1}{1} + 1 = n + 2$$

non-zero coefficients (the single leading quadratic monomial, the n linear variables, and the constant term).

Theorem 3.7.3 (Complexity of S -Polynomials). *Constructing and deterministically reducing the n targeted S -polynomials requires a worst-case total of exactly*

$$\mathcal{C}_{\text{Spoly}} = \mathcal{O}(n^3)$$

field operations.

Proof. We estimate the exact operational cost in two distinct stages: the initial construction of the S -polynomials, and their subsequent linear-algebraic reduction against the diagonalized basis.

Generation of an S -polynomial. Our algorithm systematically selects predefined pairs of diagonalized polynomials, f and g , whose leading monomials share a greatest common divisor of degree exactly one. Consequently, the multipliers required to form the S -polynomial are strictly monomials of degree one.

Multiplying a polynomial by a degree-one monomial merely shifts the algebraic degrees of its terms, requiring only coefficient re-indexing rather than explicit field multiplications. Thus, the scaling step incurs zero computational cost. However, constructing the S -polynomial requires subtracting the shifted g from the shifted f :

$$S(f, g) = x_f \cdot f - x_g \cdot g.$$

Because both f and g originally contain at most $n+2$ terms, their shifted versions also contain at most $n+2$ terms. The element-wise subtraction of these two polynomials therefore requires at most

$$2 \left(\binom{n+1}{1} + 1 \right) = 2(n+2)$$

field additions.

Reduction of an S -polynomial. The newly constructed S -polynomial has a total degree of at most two. The degree-two terms in $S(f, g)$ arise exclusively from the degree-one terms in f and g being multiplied by the respective degree-one multipliers. Because there are exactly n linear variables, there are at most n quadratic terms contributed by f and at most n contributed by g . Assuming the strict worst-case scenario where all generated quadratic monomials are distinct, the unreduced S -polynomial contains at most $2n$ quadratic terms.

To reduce the S -polynomial to a linear relation, each of these quadratic monomials must be eliminated by subtracting an appropriate scalar multiple of the corresponding diagonalized pivot row. Since a diagonalized pivot row contains up to $n + 2$ non-zero entries, eliminating a single quadratic monomial requires:

- 1 scalar multiplication applied across the pivot row (at most $n + 2$ multiplications),
- 1 vector subtraction from the S -polynomial (at most $n + 2$ additions).

Thus, eliminating one quadratic monomial costs exactly $2(n + 2)$ operations. Because there are at most $2n$ quadratic monomials to eliminate, the total reduction cost for a single S -polynomial is tightly bounded by:

$$(2n) \cdot 2(n + 2) = 4n(n + 2)$$

operations.

Total cost for n S -polynomials. Combining the construction cost and the reduction cost, the total number of field operations to generate and reduce a single S -polynomial is:

$$\mathcal{C}_{\text{single}} = 2(n + 2) + 4n(n + 2) = (4n + 2)(n + 2).$$

Because the algorithm strictly requires n such S -polynomials (one for each pair), we multiply this worst-case cost by n to derive the total asymptotic complexity:

$$\begin{aligned} \mathcal{C}_{\text{Spoly}} &= \mathcal{O}(n \cdot \mathcal{C}_{\text{single}}) \\ &= \mathcal{O}(n(4n + 2)(n + 2)) \\ &= \mathcal{O}(4n^3) \\ &= \mathcal{O}(n^3). \end{aligned}$$

□

3.7.3 Phase 3: Final Diagonalization and Secret Recovery

The final step of the algorithm diagonalizes the augmented linear subsystem of n equations in n unknowns obtained after the S -polynomial reduction stage. The analysis and combinatorial counting methods follow the precise conventions established in Phase 1: all costs are measured in total elementary field operations (both additions and multiplications) over $\mathbb{F}_q$.

Theorem 3.7.4 (Complexity of Final Diagonalization). *The complete diagonalization of the final linear subsystem, comprising the forward elimination step followed by the backward substitution step, requires at most*

$$\mathcal{C}_{\text{final_REF}} + \mathcal{C}_{\text{final_RREF}} = \mathcal{O}(n^3)$$

field operations.

Proof. The augmented linear subsystem is represented by an $n \times (n+1)$ coefficient matrix. Each row contains at most n variable coefficients plus the constant vector entry, totaling up to $n+1$ entries. We analyze the forward and backward passes separately.

Forward Elimination. For a pivot row j , where $1 \leq j \leq n$:

- **Normalization:** Because the variables before column j have already been eliminated, the row contains exactly $(n+1) - j + 1 = n+2-j$ non-zero entries. Making the pivot monic requires scaling all these entries, costing $n+2-j$ multiplications.
- **Elimination:** There are exactly $n-j$ rows below the pivot. To eliminate the entry in the pivot column for each target row, we multiply the pivot row by a scalar (costing $n+2-j$ multiplications) and subtract it (costing $n+2-j$ additions). Thus, a single elimination step costs $2(n+2-j)$ operations.

The total cost for processing pivot j is therefore:

$$(n+2-j) + 2(n+2-j)(n-j) = (n+2-j)(1+2n-2j).$$

Expanding this polynomial in terms of n and j , we get:

$$2n^2 + 5n + 2 - j(4n+5) + 2j^2.$$

Summing this exact cost over all n pivots and isolating the leading terms within asymptotic notation, we obtain the total complexity for the final forward step:

$$\mathcal{C}_{\text{final_REF}} = \mathcal{O}\left(\sum_{j=1}^n [2n^2 + 5n + 2 - j(4n+5) + 2j^2]\right)$$

$$\begin{aligned}
 &= \mathcal{O} \left(n(2n^2 + 5n + 2) - (4n + 5) \sum_{j=1}^n j + 2 \sum_{j=1}^n j^2 \right) \\
 &\stackrel{*}{=} \mathcal{O} \left(2n^3 - 4n \left(\frac{n(n+1)}{2} \right) + 2 \left(\frac{n(n+1)(2n+1)}{6} \right) \right) \\
 &= \mathcal{O} \left(2n^3 - 2n^3 + \frac{2}{3}n^3 \right) \\
 &= \mathcal{O} \left(n^3 \right),
 \end{aligned}$$

where in $\stackrel{*}{=}$ we explicitly utilized the standard power sum identities

$$\sum_{k=1}^n k = \frac{n(n+1)}{2} \quad \text{and} \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}.$$

Because the remaining leading term is cubic, this step requires $\mathcal{O}(n^3)$ operations.

Backward Substitution. After the forward elimination is complete, the pivots are already monic. We process the pivots in reverse order (from bottom to top). Let j denote the step index of the backward elimination, where $j = 1$ corresponds to the bottom-most row and $j = n$ corresponds to the top-most row.

Because the row is already in Row-Echelon Form from the left, it contains exactly 2 non-zero entries: the monic pivot variable x_{n-j+1} and the constant vector entry on the far right. At step j , there are exactly:

$$n - j$$

rows strictly above the current pivot that must be cleared. Because the row only contains 2 elements, eliminating it from a single target row requires exactly 2 multiplications and 2 additions. Each row elimination therefore costs exactly 4 operations.

Summing this cost over all n backward steps and applying asymptotic notation, we obtain:

$$\begin{aligned}
 \mathcal{C}_{\text{final_RREF}} &= \mathcal{O} \left(\sum_{j=1}^n 4(n - j) \right) \\
 &= \mathcal{O} \left(4 \left(n \sum_{j=1}^n 1 - \sum_{j=1}^n j \right) \right) \\
 &= \mathcal{O} \left(4 \left(n^2 - \frac{n(n+1)}{2} \right) \right) \\
 &= \mathcal{O} \left(4n^2 - 2n^2 \right) \\
 &= \mathcal{O} \left(n^2 \right).
 \end{aligned}$$

Because the dominant term is quadratic, the backward substitution step strictly requires $\mathcal{O}(n^2)$ operations.

Total Final Cost. Adding the respective linear-algebraic contributions yields the overarching complexity of Phase 3:

$$\mathcal{C}_{\text{final_REF}} + \mathcal{C}_{\text{final_RREF}} = \mathcal{O}(n^3) + \mathcal{O}(n^2) = \mathcal{O}(n^3).$$

□

This completes the complexity analysis of the entire algorithm.

3.7.4 Total Asymptotic Complexity

To conclude our complexity analysis, we aggregate the exact computational costs derived in the preceding sections. Table 3.1 summarizes the asymptotic bounds for each distinct phase of the proposed algorithm.

Phase	Step	Operation	Cost
Phase 1	Initial REF	Forward Elimination	$\mathcal{O}(n^6)$
Phase 1	Initial RREF	Backward Substitution	$\mathcal{O}(n^5)$
Phase 2	Spoly	Generation & Reduction	$\mathcal{O}(n^3)$
Phase 3	Final REF	Forward Elimination	$\mathcal{O}(n^3)$
Phase 3	Final RREF	Backward Substitution	$\mathcal{O}(n^2)$

Table 3.1: Summary of worst-case computational complexities per phase.

Theorem 3.7.5 (Main Result: Total Complexity). *The total worst-case computational complexity of the proposed algorithm presented in Algorithm 3 is bounded by:*

$$\mathcal{C}_{\text{total}} = \mathcal{O}(n^6).$$

Proof. By collecting the asymptotic bounds rigorously proven in Theorems 3.7.1, 3.7.2, 3.7.3, and 3.7.4, we have the following discrete contributions:

$$\begin{aligned} \mathcal{C}_{\text{REF}} &= \mathcal{O}(n^6), \\ \mathcal{C}_{\text{RREF}} &= \mathcal{O}(n^5), \\ \mathcal{C}_{\text{Spoly}} &= \mathcal{O}(n^3), \\ \mathcal{C}_{\text{final_REF}} &= \mathcal{O}(n^3), \\ \mathcal{C}_{\text{final_RREF}} &= \mathcal{O}(n^2). \end{aligned}$$

Summing the respective contributions of the individual algorithmic stages yields the overarching complexity:

$$\begin{aligned}\mathcal{C}_{\text{total}} &= \mathcal{C}_{\text{REF}} + \mathcal{C}_{\text{RREF}} + \mathcal{C}_{\text{Spoly}} + \mathcal{C}_{\text{final_REF}} + \mathcal{C}_{\text{final_RREF}} \\ &= \mathcal{O}(n^6) + \mathcal{O}(n^5) + \mathcal{O}(n^3) + \mathcal{O}(n^3) + \mathcal{O}(n^2) \\ &= \mathcal{O}(n^6).\end{aligned}$$

Because the computational cost is strictly dominated by the $\mathcal{O}(n^6)$ operations required during the initial forward elimination step over the coefficient matrix, the total time complexity of the algorithm is $\mathcal{O}(n^6)$ field operations over $\mathbb{F}_q$. This demonstrates that our algebraic algorithm recovers the binary-error LWE secret in deterministic polynomial time, provided $m = \binom{n+1}{2}$ samples are available. $\square$

Summary of Complexity for Binary Errors. Table 3.2 summarizes the complexity of the proposed algorithm in the binary error case, comparing it with the main results in the literature. As shown, our approach provides a polynomial-time bound, achieving a complexity of $\mathcal{O}(n^6)$ without relying on the heuristic assumptions or degree of regularity estimates required by previous works.

Work	No. Samples	Time Complexity	Using assumptions?	Without d_{reg} ?
[5]	$\mathcal{O}(n \log \log n)$	$\mathcal{O}\left(n^2 \cdot 2^{\frac{\omega n \log \log \log n}{8 \log \log n}}\right)$	YES	$\times$
[5]	$c \cdot n$	$\mathcal{O}\left(n^2 \cdot 2^{\omega n(1+\beta)H_2(\beta/(1+\beta))}\right)$	YES	$\times$
[49]	$c \cdot n^2$	$n^{\mathcal{O}(1/c)}$	YES	$\times$
[49]	$n^{1+\alpha}$	$2^{\tilde{\mathcal{O}}(m^{1-\alpha})}$	YES	$\times$
[48]	$\mathcal{O}(n^2)$	$\mathcal{O}\left(n^2 \cdot \binom{n+3}{3}^{\omega+2}\right)$	YES	$\times$
[27]	$\binom{n+1}{2}$	$\mathcal{O}\left(2n^{1+2\omega}\right)$	NO	$\times$
This Thesis	$\mathcal{O}\left(\frac{n(n+1)}{2}\right)$	$\mathcal{O}\left(\binom{n+1}{2}^3\right) = \mathcal{O}(n^6)$	NO	$\checkmark$

Table 3.2: Comparative overview of algebraic attacks on LWE with binary errors. Refer to Table 2.1 for detailed definitions of the notation and parameters used.

3.8 Adaptation for Binary Errors and Binary Secrets

Having extensively analyzed the algorithmic workflow for the case of binary errors and uniformly random secrets, we now turn our attention to an equally critical cryptographic scenario: LWE instances where the secret vector $\mathbf{s}$ is also drawn from a binary distribution, $\mathbf{s} \in \{0, 1\}^n$.

The assumption of a binary secret introduces n additional constraints to our algebraic system, specifically the Boolean field equations:

$$x_i^2 - x_i = 0, \quad \text{for all } i = 1, \dots, n.$$

These equations fundamentally “alter” the structure of the multivariate polynomial ring by allowing us to iteratively reduce any variable raised to a power greater than one. Consequently, the basis of our polynomial system is strictly confined to *square-free* monomials.

3.8.1 New basis dimension

In the previous scenario, the coefficient matrix operated over the complete set of degree-two monomials, which numbered $\binom{n+2}{2}$. Under the binary secret assumption, all squared variables x_i^2 are immediately reduced to linear terms x_i during the initial system generation. Thus, the degree-two block of our matrix now exclusively consists of square-free cross-terms of the form $x_i x_j$ for $i < j$.

The number of degree-two square-free monomials is exactly $\binom{n}{2}$. Combining these with the n linear variables $\binom{n}{1}$ and the single constant term $\binom{n}{0}$, the total width of our new square-free basis is bounded by:

$$N_{\text{bin}} = \binom{n}{2} + n + 1 = \frac{n(n-1)}{2} + n + 1 = \frac{n^2 + n + 2}{2}.$$

To achieve full column rank over the quadratic block during the initial Phase 1 diagonalization, the required number of LWE samples (and thus the number of rows in our coefficient matrix) is reduced to exactly:

$$m_{\text{bin}} = \binom{n}{2}.$$

This reduction in both matrix width and row count accelerates the initial forward elimination and backward substitution passes.

3.8.2 Selection of S -Polynomial Pairs

Following the diagonalization of the coefficient matrix via **DiagA** and **DiagB**, the system contains a set of polynomials whose leading terms are distinct, square-free

monomials of degree exactly two. Because the squared terms x_i^2 no longer exist in our basis, we cannot reuse the $(x_i^2, x_i x_{i+1})$ pairs utilized in Phase 2 of the previous algorithm.

To proceed with the degree-fall mechanism, we must select exactly n new pairs of polynomials from this square-free set such that the leading terms of each pair share exactly one variable. To ensure a deterministic and computationally efficient selection, we employ again a cyclic pairing strategy based on modular arithmetic. Let the variables be indexed as $x_1, \dots, x_n$. We define the arithmetic on the indices modulo n , with the convention that any computed index maps to the strictly positive representative in the set $\{1, 2, \dots, n\}$.

For each index $i \in \{1, 2, \dots, n\}$, we select the i -th pair of polynomials, denoted as (f_i, g_i) , by choosing the unique polynomials in the diagonalized system whose leading terms are:

$$\begin{aligned} \text{LM}(f_i) &= x_i x_{i+1} \\ \text{LM}(g_i) &= x_i x_{i+2} \end{aligned}$$

By construction, the greatest common divisor of the leading monomials for the i -th pair is exactly the variable x_i . Consequently, the optimal condition for avoiding degree inflation during the S -polynomial computation is intrinsically satisfied for all selected pairs:

$$\deg(\gcd(\text{LM}(f_i), \text{LM}(g_i))) = \deg(x_i) = 1.$$

Furthermore, this selection strategy guarantees that all $2n$ polynomials involved in the n pairs are strictly distinct, provided that $n \geq 5$. Thus, no polynomial is reused across the n generated S -polynomials.

Feasibility Condition ($n \geq 5$). To successfully extract the secret, the algorithm requires exactly n distinct S -polynomials. To prevent redundant relations, no initial polynomial should be reused across the n computed S -polynomial pairs. Therefore, the algorithm strictly requires $2n$ distinct polynomials from the diagonalized set. Since our diagonalized set provides exactly $\binom{n}{2}$ independent square-free polynomials, the algebraic feasibility condition to form enough disjoint pairs is:

$$\binom{n}{2} \geq 2n.$$

Expanding the binomial coefficient yields:

$$\frac{n(n-1)}{2} \geq 2n.$$

Since the dimension n is strictly positive, we obtain the relation:

$$n - 1 \geq 4 \implies n \geq 5.$$

3.8.3 Complexity Re-Evaluation for Binary Secrets

Because the dimensions of the coefficient matrix have shifted from $m = \binom{n+1}{2}$ to $m = \binom{n}{2}$, we must formally re-evaluate the exact computational complexity. We maintain the metric of total field operations over $\mathbb{F}_q$. Because Phase 1 (the initial system diagonalization) inherently dominates the algorithmic runtime, formally evaluating the worst-case bounds of this first phase under the binary constraints is sufficient to rigorously establish the total asymptotic complexity of the entire algorithm.

Theorem 3.8.1 (Complexity of Binary Secret Diagonalization). *The forward elimination step (REF) on the square-free system requires at most*

$$\mathcal{C}_{\text{REF_bin}} = \mathcal{O}(n^6)$$

field operations, and the backward substitution step (RREF) requires at most

$$\mathcal{C}_{\text{RREF_bin}} = \mathcal{O}(n^5)$$

field operations.

Proof. We track the algorithmic cost using the updated parameters $m = \binom{n}{2}$ and $N_{\text{bin}} = \frac{n^2+n+2}{2}$.

Forward Elimination (DiagA). For a given pivot row $1 \leq j \leq m$, normalization costs at most N_{bin} operations. There are $m - j$ rows below it, each requiring $2N_{\text{bin}}$ operations to clear. Summing over all m pivots:

$$\begin{aligned} \mathcal{C}_{\text{REF_bin}} &= \sum_{j=1}^m [N_{\text{bin}} + 2N_{\text{bin}}(m - j)] \\ &= N_{\text{bin}} \sum_{j=1}^m [1 + 2m - 2j] \\ &= N_{\text{bin}} \left[m + 2m^2 - 2 \frac{m(m+1)}{2} \right] \\ &= N_{\text{bin}} [m + 2m^2 - m^2 - m] \\ &= N_{\text{bin}} m^2. \end{aligned}$$

Substituting $N_{\text{bin}} = \frac{n^2+n+2}{2}$ and $m = \binom{n}{2} = \frac{n(n-1)}{2}$ into the derived expression and applying asymptotic notation, we obtain:

$$\begin{aligned} \mathcal{C}_{\text{REF_bin}} &= \mathcal{O} \left(\left(\frac{n^2 + n + 2}{2} \right) \left(\frac{n(n-1)}{2} \right)^2 \right) \\ &= \mathcal{O}(n^2 \cdot n^4) \\ &= \mathcal{O}(n^6). \end{aligned}$$

Backward Substitution (DiagB). After the forward step, the rows are cleared below the main diagonal. Crucially, each pivot row is monic in a single square-free quadratic monomial and contains at most n linear variables and 1 constant. The effective width of the pivot row is exactly $n + 2$. We process the m pivots from bottom to top. At step j , there are $m - j$ rows above to clear. Each row requires $2(n + 2)$ operations:

$$\begin{aligned} \mathcal{C}_{\text{RREF_bin}} &= \sum_{j=1}^m 2(n + 2)(m - j) \\ &= 2(n + 2) \left[m^2 - \frac{m(m + 1)}{2} \right] \\ &= (n + 2) [m^2 - m]. \end{aligned}$$

Substituting $m = \frac{n(n-1)}{2}$ explicitly and applying asymptotic notation to isolate the dominant terms:

$$\begin{aligned} \mathcal{C}_{\text{RREF_bin}} &= \mathcal{O} \left((n + 2) \left[\left(\frac{n(n-1)}{2} \right)^2 - \frac{n(n-1)}{2} \right] \right) \\ &= \mathcal{O} \left((n + 2) \left(\frac{n(n-1)}{2} \right)^2 \right) \\ &= \mathcal{O}(n \cdot n^4) \\ &= \mathcal{O}(n^5). \end{aligned}$$

□

Despite the matrix dimensions scaling down by a constant fraction due to the Boolean constraints, the asymptotic polynomial bounds for the binary secret scenario perfectly match those of the general case. Because the subsequent S -polynomial reduction and final diagonalization stages are inherently bounded by lower-degree complexities (specifically $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$), this $\mathcal{O}(n^6)$ initial forward elimination step acts as the absolute computational bottleneck. Therefore, evaluating this first diagonalization is entirely sufficient to conclude that the total asymptotic complexity of our algebraic algorithm remains bounded by $\mathcal{O}(n^6)$.

4 | Experimental Results

In this chapter, we evaluate the practical execution times and asymptotic scaling of our algorithm, comparing its performance directly against MAGMA’s native F4 implementation. All experiments were executed on a dedicated compute server equipped with an Intel Core i9-14900KF processor and 128 GB of RAM, running MAGMA version V2.28-23. To ensure statistical reliability, the recorded CPU times for each parameter set were averaged over 100 independent instantiations of the LWE problem. The complete MAGMA source code used to conduct all tests is provided in Appendix B.

4.1 Methodology

To benchmark our novel algorithm against established attacks, we compared its execution time against MAGMA’s native `GroebnerBasis()` function. MAGMA is a widely used computer algebra system in cryptographic research, and its native solver implements a highly optimized, C-compiled version of Faugère’s F4 algorithm, making it a natural baseline for our experiments.

To ensure a fair comparison, the native F4 algorithm was fed the exact same initial polynomial system $\mathcal{F}$ generated by the LWE samples. Time was recorded strictly for the solving phase, excluding the generation of the initial samples. When evaluating the results, it is crucial to acknowledge an inherent software-level disparity: MAGMA’s native F4 utilizes low-level compiled routines and advanced sparse linear algebra libraries, whereas our proposed algorithm is currently implemented as a high-level MAGMA script. Consequently, F4 naturally benefits from a massive constant-factor speedup in basic operations.

To thoroughly analyze this software gap and evaluate the practical performance of our approach, we implemented two variants of our algorithm: a *direct implementation* that manipulates MAGMA’s polynomial objects exactly as mathematically described in Chapter 3, and a *matrix-optimized implementation* specifically tailored to bypass the polynomial overhead by directly leveraging MAGMA’s highly efficient native linear algebra routines.

4.2 Performance and Asymptotic Behavior

In our first series of experiments, we fixed the prime modulus $q = 257$, working over the finite field $\mathbb{F}_{257}$, and varied the secret dimension n . To ensure statistical significance and mitigate the impact of system background processes, each configuration was executed 100 times. Table 4.1 details the average execution time per single instance for our direct implementation and MAGMA's F4 as the dimension increases, while Figure 4.1 illustrates the asymptotic growth of these average times as the dimension scales.

Parameters		Proposed Algorithm	MAGMA F4
n	m	Average Time (ms)	Average Time (ms)
10	55	4.00	0.80
15	120	22.60	3.10
20	210	84.50	5.10
25	325	255.60	10.30
30	465	657.80	20.60
35	630	1517.40	38.70
40	820	3204.30	72.70

Table 4.1: Average execution time (ms) per instance of our direct implementation versus MAGMA's native F4 over $\mathbb{F}_{257}$.

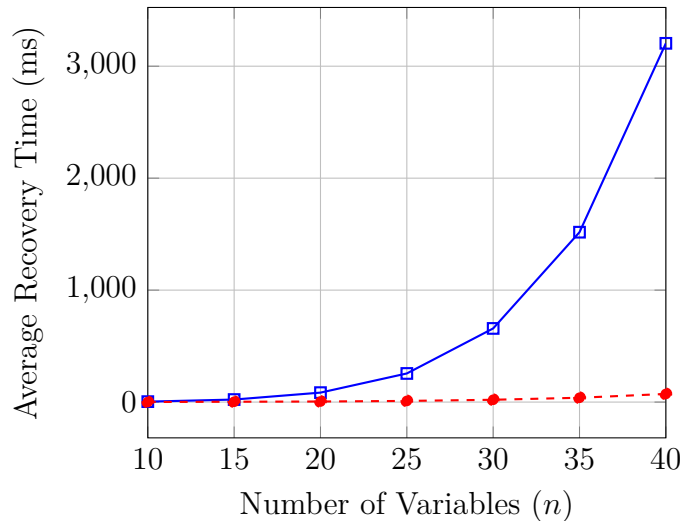


Figure 4.1: Average recovery time (ms) as a function of the number of variables n over $\mathbb{F}_{257}$. The plot compares our direct approach ($\text{--}\square\text{--}$) with MAGMA's F4 ($\text{--}\bullet\text{--}$).

4.3 Matrix-Optimized Implementation

While the direct implementation perfectly mirrors the mathematical theory, it is heavily penalized by the MAGMA interpreter. Operations such as polynomial multiplication, term cancellation, and memory reallocation inside iterative loops introduce a massive computational overhead.

To bridge this software gap, we developed a matrix-optimized variant of our algorithm. Since MAGMA is exceptionally optimized for dense linear algebra, this variant entirely avoids polynomial operations. Instead, it maps all possible monomials to columns of a matrix, representing polynomials as dense coefficient vectors in $\mathbb{F}_q$. The S -polynomial generation and reduction phases are executed as purely linear algebraic operations (vector-matrix multiplications) using MAGMA's natively compiled `EchelonForm()` routine.

The algorithm remains mathematically identical, but shifts the computational burden from the interpreter to the underlying C-compiled linear algebra engine. To evaluate the true asymptotic behavior and provide a fairer comparison, we tested both the matrix-optimized algorithm and MAGMA's F4 on significantly larger dimensions, scaling up to $n = 100$ over $\mathbb{F}_{257}$. The average execution times, recorded over 100 independent trials for each dimension, are detailed in Table 4.2. To better visualize the scaling behavior of both algorithms, Figure 4.2 plots these execution times against the system dimension, highlighting the trajectory of their asymptotic growth.

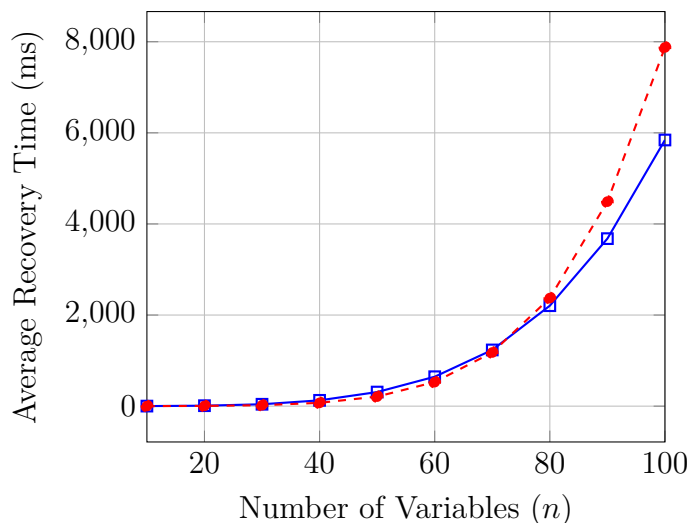


Figure 4.2: Asymptotic scaling of the average recovery time (ms) over $\mathbb{F}_{257}$. The plot compares the matrix-optimized implementation ($\text{--}\square\text{--}$) against MAGMA's F4 ($\text{--}\bullet\text{--}$).

Parameters		Matrix-Optimized Algorithm	MAGMA F4
n	m	Average Time (ms)	Average Time (ms)
10	55	1.20	1.00
20	210	10.00	5.10
30	465	41.30	20.40
40	820	127.40	72.70
50	1275	309.30	209.60
60	1830	645.90	528.00
70	2485	1233.80	1178.10
80	3240	2205.30	2372.20
90	4095	3677.60	4490.40
100	5050	5843.10	7875.30

Table 4.2: Performance comparison between the matrix-optimized algorithm and MAGMA’s F4. As the dimension scales, the $\mathcal{O}(n^6)$ complexity of our algorithm overcomes the initial constant-factor efficiency of F4, showing better performance for $n \geq 80$.

4.4 Theoretical vs. Practical Performance

As observed in the previous sections, while the direct implementation is initially penalized by the software environment, the matrix-based optimization drastically closes the performance gap. Even more significantly, Table 4.2 shows that for $n \geq 80$, our proposed algorithm overtakes MAGMA’s native F4, becoming strictly faster.

To formally explain this crossover, we must analyze the asymptotic complexity of computing a Gröbner basis via F4. In algebraic cryptanalysis, this complexity is fundamentally governed by the degree of regularity, d_{reg} . Instead of relying on theoretical estimates, we explicitly verified this metric by isolating the homogeneous quadratic components of our original system and evaluating them using MAGMA’s native `HilbertSeries()` function. Because our setup produces a highly overdetermined system of $m = \binom{n+1}{2} = \mathcal{O}(n^2)$ independent quadratic equations, the Hilbert series consistently shows that the ideal collapses at degree 2, experimentally showing that $d_{\text{reg}} = 2$.

According to the established bounds for semi-regular systems provided by Albrecht et al. [5], based on the foundational work of Bardet et al. [12], the general number of operations required to compute a Gröbner basis is bounded by:

$$\mathcal{O}\left(m d_{\text{reg}} \binom{n + d_{\text{reg}}}{d_{\text{reg}}}\right), \quad \text{as } d_{\text{reg}} \rightarrow \infty, \quad (4.1)$$

where $2 \leq \omega \leq 3$ is the linear algebra constant. This generic bound implicitly assumes that the rows of the Macaulay matrix are generated by multiplying the m initial equations by monomials to reach higher degrees.

However, because our specific parameterization strictly bounds the degree of regularity to $d_{\text{reg}} = 2$, and our initial m equations are already of degree 2, no polynomial multiplications are performed. F4 simply constructs a single Macaulay matrix using the exact $m = \mathcal{O}(n^2)$ initial equations as rows, and the $\binom{n+2}{2} = \mathcal{O}(n^2)$ monomials up to degree 2 as columns.

Applying the generic bound formula directly to this specific case would yield an overestimated complexity of $\mathcal{O}(n^{2\omega+2})$, falsely inflating the row count by a factor of m . Instead, the true computational cost is strictly defined by the row reduction of an $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ matrix, yielding an asymptotic complexity of $\mathcal{O}(n^{2\omega})$.

Even if MAGMA’s highly optimized backend employs sub-cubic matrix algorithms ($\omega \leq 3$) to lower F4’s theoretical bound, this advantage does not translate to practical superiority in our context. As derived in Chapter 3, our targeted reduction algorithm operates with a strict worst-case complexity of $\mathcal{O}(n^6)$.

Despite MAGMA potentially possessing a better asymptotic bound for its linear algebra ($\omega \leq 3$), our matrix-based approach empirically overtakes the native F4 solver for $n \geq 80$. This crossover proves that the symbolic overhead inherent to general-purpose Gröbner basis solvers completely dominates the execution time. Unlike F4, which has to dynamically manage, sort, and reduce a vast and unpredictable set of S -polynomials, our algorithm derives its efficiency from a mathematically sound targeted selection of n specific S -polynomials. By bypassing generalized polynomial structures and directly performing the reduction via dense matrix operations, our approach reduces the hidden constant factors. This precision allows a strictly $\mathcal{O}(n^6)$ algorithm to definitively outperform the proprietary standard in practice.

4.5 Success Probability and Prime Modulus

To further validate our theoretical model, we conducted a final series of experiments focusing on the algorithm’s reliability. Since the direct and matrix-optimized implementations are mathematically identical, their success rate is fundamentally the same and relies on the initial LWE samples generating a full-rank coefficient matrix. As established in Appendix A, the probability of achieving this full-rank condition is overwhelmingly high, bounded theoretically by approximately $1 - \mathcal{O}(q^{-(n+2)})$.

To empirically verify this probabilistic bound, we fixed the secret dimension at $n = 10$ and varied the prime modulus q , starting from small primes up to standard cryptographic sizes. For each selected modulus, we executed the proposed algorithm over 10000 independent, randomly generated LWE instantiations and recorded the success rate (defined as the exact recovery of the true secret vector).

Modulus q	Trials	Successful Recoveries	Success Rate
257	10000	9963	99.63%
443	10000	9975	99.75%
743	10000	9986	99.86%
1021	10000	9989	99.89%
3329	10000	9998	99.98%
4093	10000	9998	99.98%
65537	10000	10000	100%

Table 4.3: Empirical success rates for a fixed dimension $n = 10$ across varying prime moduli. The algorithm recovers the secret vector consistently, demonstrating that the full-rank condition is easily satisfied even for minimal values of q .

As demonstrated in Table 4.3, the empirical data strictly aligns with our theoretical expectations. While low moduli (such as $q = 257$) might introduce a marginal risk of linear dependency, the success rate rapidly converges to 100% as q increases. This confirms that for any practical cryptographic parameter choice, the proposed algorithm is highly robust and recovers the secret with virtual certainty.

5 | Conclusions

5.1 Summary of Contributions

In this thesis, we presented a novel algebraic attack for the LWE problem parameterized with binary errors. By framing the cryptographic instance as a PoSSo problem, we developed an exact algorithm that successfully bypasses the heuristic assumptions and unpredictable complexity estimates typically associated with standard Gröbner basis computations.

The core contribution of this work is the meticulous construction of a division-free, linear-algebraic reduction pipeline. We demonstrated that given $m = \binom{n+1}{2}$ samples, the initial polynomial system can be systematically reduced to a degree-two diagonalized basis. From this basis, a targeted cyclic selection of S -polynomial pairs inherently guarantees a degree-fall, yielding a full-rank linear subsystem. By relying exclusively on deterministic matrix row operations over $\mathbb{F}_q$, the algorithm recovers the exact LWE secret vector without resorting to heuristic estimates, such as the degree of regularity.

Furthermore, we provided a rigorous mathematical complexity analysis, establishing that the total worst-case computational cost is strictly bounded by $\mathcal{O}(n^6)$ field operations. We also extended this theoretical framework to the scenario of binary secrets, proving that the introduction of Boolean field equations compresses the system to a square-free basis. This explicitly reduces the data complexity to exactly $m = \binom{n}{2}$ samples, while maintaining the robust $\mathcal{O}(n^6)$ polynomial-time recovery bound.

5.2 Further Work

A natural continuation of this research is the generalization of both the proposed algorithm and its underlying algebraic methodology. The primary direction is to extend this exact approach to the setting of LWE with bounded errors. The Arora-Ge framework, which translates the LWE problem into a PoSSo instance, is particularly well-suited to scenarios where the error distribution is supported on a finite, bounded set, as the size of this set directly dictates the degree of the resulting polynomial system.

Moreover, LWE with bounded errors constitutes a highly relevant variant for practical applications, making the deployment of LWE schemes more viable in constrained real-world settings. Because the cryptographic literature often implicitly assumes bounded-error regimes when analyzing LWE-based constructions, a focused algebraic study of this setting appears both natural and highly promising.

Can the proposed approach be effectively generalized to solve the LWE problem with Bounded Errors?

Another possible research direction concerns the required number of samples. In the current binary error setting, with n variables and degree 2, the algorithm requires a number of samples m equal to the number of exact degree-two monomials in n variables. It is therefore natural to ask what happens when fewer samples are available.

Can the proposed approach be adapted to operate effectively in an undersampled regime?

Finally, it is worth investigating whether the idea underlying the algorithm can be extended beyond the LWE setting to more general PoSSo problems, and how such an extension could contribute to a better understanding of the complexity of Gröbner basis computations.

Can this approach be successfully applied to other PoSSo scenarios and help characterize the complexity of Gröbner basis computations?

These questions remain open and appear to be highly promising directions for future cryptographic and algebraic research.

A | Probability of Linear Independence for LWE Samples

Throughout this thesis, the algorithm introduced is designed to operate under the condition that the initial coefficient matrix constructed from the LWE samples achieves full rank. Specifically, the rows representing the expanded polynomials must be linearly independent to successfully diagonalize the quadratic block and establish the required basis for the degree-fall mechanism.

In algebraic cryptanalysis, to evaluate the probability that a set of LWE samples satisfies this condition, it is standard practice to model the expanded polynomials as random vectors within the target vector space over $\mathbb{F}_q$. This modeling allows for an explicit computation of the probability that the sampled vectors are linearly independent, providing a theoretical guarantee on the success rate of the method.

Let D denote the total dimension of the vector space, and let $m \leq D$ denote the number of uniformly random vectors drawn from $\mathbb{F}_q^D$. The probability $\Pr[m, D, q]$ that these m vectors are linearly independent is given by the ratio of the number of ways to choose m linearly independent vectors to the total number of possible m -tuples of vectors:

$$\begin{aligned}\Pr[m, D, q] &= \prod_{i=0}^{m-1} \frac{q^D - q^i}{q^D} \\ &= \prod_{i=0}^{m-1} \left(1 - \frac{1}{q^{D-i}}\right).\end{aligned}$$

We now evaluate this exact probability for the two specific algorithmic scenarios presented in this work.

A.1 Case 1: Binary Errors

In the general `BinaryError`-LWE scenario, the algorithm operates over the complete set of degree-two monomials. The total dimension of the vector space (the width

of the coefficient matrix) is:

$$D_1 = \binom{n+2}{2}.$$

To successfully isolate the quadratic block, the algorithm requires exactly

$$m_1 = \binom{n+1}{2}$$

samples. We observe that the difference between the ambient dimension and the number of vectors is strictly a function of n :

$$\begin{aligned} D_1 - m_1 &= \frac{(n+2)(n+1)}{2} - \frac{(n+1)n}{2} \\ &= \frac{n+1}{2}(n+2-n) = n+1. \end{aligned}$$

Substituting $D_1 = m_1 + n + 1$ into our probability formula, we obtain:

$$\begin{aligned} \Pr[m_1, D_1, q] &= \prod_{i=0}^{m_1-1} \left(1 - \frac{1}{q^{m_1+n+1-i}}\right) \\ &= \left(1 - \frac{1}{q^{n+2}}\right) \left(1 - \frac{1}{q^{n+3}}\right) \cdots \left(1 - \frac{1}{q^{m_1+n+1}}\right). \end{aligned}$$

Because $q \geq 2$ and n is typically large in cryptographic settings, the term $\frac{1}{q^{n+2}}$ is exceptionally small. Therefore, this product converges overwhelmingly to 1, demonstrating that the $\binom{n+1}{2}$ samples are linearly independent with high probability.

A.2 Case 2: Binary Errors and Binary Secrets

Under the assumption of a binary secret, the Boolean field equations $x_i^2 - x_i = 0$ restrict the polynomial ring to a square-free basis. The new ambient dimension of the vector space is:

$$D_2 = \binom{n}{2} + n + 1.$$

Because the squared terms are eliminated, the algorithm requires fewer samples, specifically

$$m_2 = \binom{n}{2}.$$

Substituting $D_2 = m_2 + n + 1$ into the probability formula yields the identical structural bound:

$$\Pr[m_2, D_2, q] = \prod_{i=0}^{m_2-1} \left(1 - \frac{1}{q^{m_2+n+1-i}}\right)$$

$$= \left(1 - \frac{1}{q^{n+2}}\right) \left(1 - \frac{1}{q^{n+3}}\right) \cdots \left(1 - \frac{1}{q^{m_2+n+1}}\right).$$

Conclusion. In both the standard binary-error case and the square-free binary-secret case, the difference between the matrix dimension and the required number of samples is exactly $n+1$. Consequently, both variations of the algorithm succeed in finding a full-rank initial basis with a nearly identical, overwhelmingly high probability of approximately $1 - \mathcal{O}(q^{-(n+2)})$.

A.3 Numerical Example: Practical Parameter Sizes

To contextualize these theoretical bounds within real-world cryptographic settings, we evaluate the probability product using established lattice-based parameters. In modern lattice cryptography, a dimension of $n = 256$ is a standard choice, serving as the base dimension for numerous schemes and cryptanalytic benchmarks.

Conversely, there is no single standard choice for the prime modulus q . The modulus is typically selected based on the specific noise rate and security targets of the scheme, often governed by the general requirement that $q > n$ to prevent trivial decryption failures. For example, the standardized key encapsulation mechanism CRYSTALS-Kyber utilizes $q = 3329$ [8], while cryptanalytic literature specifically targeting Binary-LWE, such as the lattice decoding attacks proposed by Bai and Galbraith, frequently evaluates parameters like $q = 4093$ [10].

As demonstrated in our probability formulas, the likelihood of generating a full-rank matrix is approximately $1 - \mathcal{O}(q^{-(n+2)})$. Crucially, this success probability strictly increases as the modulus q grows. Therefore, to establish a conservative lower bound on the algorithm's success rate, we evaluate the system using a small, valid prime modulus, such as the Fermat prime $q = 257$. If the matrix achieves full rank with overwhelming probability for this small modulus, it is mathematically guaranteed to perform even better for larger, standard choices.

Evaluating the general binary-error scenario (Case 1) with $n = 256$ and $q = 257$, the matrix dimensions evaluate to:

- vector space dimension: $D_1 = \binom{258}{2} = 33153$,
- required samples: $m_1 = \binom{257}{2} = 32896$.

For the binary-secret scenario (Case 2), the structural elimination of the squared terms reduces the matrix dimensions to:

- vector space dimension: $D_2 = \binom{256}{2} + 256 + 1 = 32897$,

- required samples: $m_2 = \binom{256}{2} = 32640$.

In both cases, the algorithmic failure probability (the chance that the generated coefficient matrix fails to achieve full row rank) is strictly dominated by the dominant subtractive term in the probability product, which evaluates to:

$$\text{Failure Probability} \approx q^{-(n+2)} = 257^{-258}.$$

Because $q = 257 \approx 2^8$, we can highly accurately approximate this failure probability in base-2:

$$257^{-258} \approx (2^8)^{-258} = 2^{-2064}.$$

A failure probability of 2^{-2064} is negligible. In modern computational cryptography, probabilities below 2^{-128} are considered practically impossible to observe. Because any larger standard modulus (such as 3329 or 4093) will only further diminish this failure probability, we conclude that for any practical cryptographic parameter choice, the initial algebraic system generated by the LWE samples will achieve full rank with virtual certainty.

B | MAGMA implementation

This appendix provides the core implementation of the algebraic attack discussed in the thesis. The code is written in MAGMA and shows the fundamental steps used to solve the **BinaryError-LWE** instance. We also included basic time-tracking commands within the main function to help measure the performance of each individual step. These timings can be printed to the terminal by setting the `debug` flag to `true`.

B.1 Sample Generation

The first step is to generate a random LWE instance and convert its samples into a quadratic polynomial system.

Algorithm 4 LWE Sample and Polynomial System Generation

Require: Dimension n , prime modulus q , number of samples m

Ensure: Polynomial system $\mathcal{F}$, secret vector $\mathbf{s}$ and polynomial ring P

```
1: function GenerateLWEInstance(n, q, m)
2:   Fq := GF(q);                                ▷ Define the finite field
3:   P := PolynomialRing(Fq, n, "grevlex");      ▷ Define the polynomial ring
4:   AssignNames(~P, ["x" * IntegerToString(i):i in [1..n]]);

5:   s := Vector([Random(Fq):i in [1..n]]);      ▷ Secret vector
6:   A := Matrix(Fq, m, n, [Random(Fq):i in [1..m*n]]);  ▷ Matrix
7:   e := Vector([Fq!Random([0,1]):i in [1..m]]);  ▷ Error vector
8:   b := Vector([InnerProduct(A[i],s)+e[i]:i in [1..m]]);

9:   F := [];                                     ▷ Conversion to polynomials
10:  for i in [1..m] do
11:    ax := Fq!0;
12:    for j in [1..n] do
13:      ax += A[i,j] * P.j;
14:    end for
15:    err_poly := b[i] - ax;
```

```

16:   Append(~F, err_poly * (err_poly - 1));
17: end for
18: return F, s, P;
19: end function;

```

B.2 Polynomial System Reduction

Here we define the forward elimination (**DiagA**) and backward substitution (**DiagB**) functions. These are used to perform Gaussian elimination directly on the polynomials.

Algorithm 5 Polynomial System Reduction (**DiagA** and **DiagB**)

```

1: function DiagA(F)
2:   for i in [1..#F] do
3:     /* Pivot Search */
4:     p := i;
5:     for k in [i+1..#F] do
6:       if F[k] ne 0 and (F[p] eq 0 or
          LeadingMonomial(F[k]) gt LeadingMonomial(F[p])) then
7:         p := k;
8:       end if
9:     end for
10:    /* Swap rows */
11:    temp := F[i];
12:    F[i] := F[p];
13:    F[p] := temp;
14:    /* Proceed with elimination */
15:    if F[i] ne 0 then
16:      lc := LeadingCoefficient(F[i]);
17:      F[i] := F[i] / lc;                                ▷ Normalize pivot row
18:      lm_i := LeadingMonomial(F[i]);
19:      for j in [i+1..#F] do
20:        if F[j] ne 0 and LeadingMonomial(F[j]) eq lm_i then
21:          c := LeadingCoefficient(F[j]);
22:          F[j] := F[j] - c * F[i];                        ▷ Eliminate downward
23:        end if
24:      end for
25:    end if
26:  end for
27:  return [f:f in F | f ne 0];

```

```

28: end function;

29: function DiagB(F)
30: for i in [#F..1 by -1] do
31:   lm_i := LeadingMonomial(F[i]);
32:   for j in [i-1..1 by -1] do
33:     c := MonomialCoefficient(F[j], lm_i);
34:     if c ne 0 then
35:       F[j] := F[j] - c * F[i];           ▷ Eliminate upward
36:     end if
37:   end for
38: end for
39: return F;
40: end function;

```

B.3 Proposed Algorithm

The main function orchestrates the entire process. It reduces the initial system, generates the required S -polynomials, and performs the final reduction to obtain a diagonalized linear subsystem. The algorithm returns a system of linear equations of the form $x_i - c_i = 0$, from which the secret components c_i can be trivially recovered.

Algorithm 6 Direct Implementation of the Proposed Algorithm

Require: Polynomial system $\mathcal{F}$, polynomial ring P , dimension n , boolean flag `debug`

Ensure: Diagonalized linear subsystem $\mathcal{L}_{\text{RREF}}$

```

1: function DirectAlgorithm(F, P, n, debug)
2: t_total_start := Cputime();

3: /* Phase 1: Initial System Diagonalization */
4: t0 := Cputime();
5: F_REF := DiagA(F);
6: F_RREF := DiagB(F_REF);
7: t_phase1 := Cputime(t0);

8: /* Phase 2: S-Polynomial Generation and Reduction */
9: t0 := Cputime();
10: L_LIN := [];
11: for i in [1..n] do

```

```

12:   idx1 := i;
13:   idx2 := (i mod n) + 1;
14:   lm_f := P.idx1 * P.idx1;
15:   lm_g := P.idx1 * P.idx2;
16:   f_poly := 0; g_poly := 0;
17:   for poly in F_RREF do
18:     if LeadingMonomial(poly) eq lm_f then
19:       f_poly := poly;
20:     end if
21:     if LeadingMonomial(poly) eq lm_g then
22:       g_poly := poly;
23:     end if
24:   end for
25:   if f_poly ne 0 and g_poly ne 0 then
26:     S := P.idx2 * f_poly - P.idx1 * g_poly;
27:     for poly in F_RREF do
28:       lm_i := LeadingMonomial(poly);
29:       c := MonomialCoefficient(S, lm_i);
30:       if c ne 0 then
31:         S := S - c * poly;
32:       end if
33:     end for
34:     if S ne 0 then
35:       Append(~L_LIN, S);
36:     end if
37:   end if
38: end for

39: /* Phase 3: Second Reduction */
40: t0 := Cputime();
41: L_LIN := DiagA(L_LIN);
42: L_RREF := DiagB(L_LIN);
43: t_phase3 := Cputime(t0);

44: t_total := Cputime(t_total_start);

45: /* Optional Timings */
46: if debug then
47:   printf "\n-- Execution Times --\n";
48:   printf "Phase 1:  %o s\n", t_phase1;
49:   printf "Phase 2:  %o s\n", t_phase2;
50:   printf "Phase 3:  %o s\n", t_phase3;
51:   printf "Total:   %o s\n\n", t_total;

```

```
52: end if  
53: return L_RREF;  
54: end function;
```

B.4 Implementation and Code Availability

We have presented the core logic of the algebraic attack operating directly on polynomials. Both this implementation and the optimized matrix-based version, which were used to obtain the experimental results discussed in Chapter 4, are publicly available at:

https://github.com/fumianimax/SOLVING_LWE_WITH_BINARY_ERRORS

The repository also includes the complete suite of benchmarking scripts used to generate the performance tables.

Bibliography

- [1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. “Efficient Lattice (H)IBE in the Standard Model”. In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 553–572. DOI: 10.1007/978-3-642-13190-5_28.
- [2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. “Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE”. In: *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Springer, 2010, pp. 98–115. DOI: 10.1007/978-3-642-14623-7_6.
- [3] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. “Simultaneous Hardcore Bits and Cryptography against Memory Attacks”. In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*. Ed. by Omer Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer, 2009, pp. 474–495. DOI: 10.1007/978-3-642-00457-5_28.
- [4] Martin R. Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors”. In: *IACR Cryptol. ePrint Arch.* (2015), p. 46. URL: <http://eprint.iacr.org/2015/046>.
- [5] Martin R. Albrecht et al. “Algebraic Algorithms for LWE Problems”. In: *IACR Cryptol. ePrint Arch.* (2014), p. 1018. URL: <http://eprint.iacr.org/2014/1018>.
- [6] Benny Applebaum et al. “Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems”. In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 595–618. DOI: 10.1007/978-3-642-03356-8_35.

- [7] Sanjeev Arora and Rong Ge. “New Algorithms for Learning in Presence of Errors”. In: *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*. Ed. by Luca Aceto, Monika Henzinger, and Jirí Sgall. Vol. 6755. Lecture Notes in Computer Science. Springer, 2011, pp. 403–415. DOI: 10.1007/978-3-642-22006-7_34.
- [8] Roberto Avanzi et al. *CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation (Version 3.02)*. Submission to the NIST Post-Quantum Cryptography Standardization Project (Round 3). Aug. 4, 2021. URL: <https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf>.
- [9] László Babai. “On Lovász’ lattice reduction and the nearest lattice point problem”. In: *Comb.* 6.1 (1986), pp. 1–13. DOI: 10.1007/BF02579403.
- [10] Shi Bai and Steven D. Galbraith. “Lattice Decoding Attacks on Binary LWE”. In: *Information Security and Privacy - 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings*. Ed. by Willy Susilo and Yi Mu. Vol. 8544. Lecture Notes in Computer Science. Springer, 2014, pp. 322–337. DOI: 10.1007/978-3-319-08344-5_21.
- [11] Abhishek Banerjee, Chris Peikert, and Alon Rosen. “Pseudorandom Functions and Lattices”. In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 719–737. DOI: 10.1007/978-3-642-29011-4_42.
- [12] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. “On the complexity of the F5 Gröbner basis algorithm”. In: *J. Symb. Comput.* 70 (2015), pp. 49–70. DOI: 10.1016/J.JSC.2014.09.025. URL: <https://doi.org/10.1016/j.jsc.2014.09.025>.
- [13] Avrim Blum, Adam Kalai, and Hal Wasserman. “Noise-tolerant learning, the parity problem, and the statistical query model”. In: *J. ACM* 50.4 (2003), pp. 506–519. DOI: 10.1145/792538.792543.
- [14] Joppe W. Bos et al. “Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 1006–1018. DOI: 10.1145/2976749.2978425.

-
- [15] Zvika Brakerski. “Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP”. In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 868–886. DOI: 10.1007/978-3-642-32009-5_50.
 - [16] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*. Ed. by Shafi Goldwasser. ACM, 2012, pp. 309–325. DOI: 10.1145/2090236.2090262.
 - [17] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “Fully Homomorphic Encryption without Bootstrapping”. In: *IACR Cryptol. ePrint Arch.* (2011), p. 277. URL: <http://eprint.iacr.org/2011/277>.
 - [18] Zvika Brakerski and Vinod Vaikuntanathan. “Efficient Fully Homomorphic Encryption from (Standard) LWE”. In: *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. Ed. by Rafail Ostrovsky. IEEE Computer Society, 2011, pp. 97–106. DOI: 10.1109/FOCS.2011.12.
 - [19] Zvika Brakerski et al. “Classical hardness of learning with errors”. In: *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 575–584. DOI: 10.1145/2488608.2488680.
 - [20] B. Buchberger. “Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)”. English translation in *J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions*. Vol. 41, Number 3-4, Pages 475–511, 2006. PhD thesis. Mathematical Institute, University of Innsbruck, Austria, 1965.
 - [21] Johannes Buchmann et al. “High-Performance and Lightweight Lattice-Based Public-Key Encryption”. In: *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, IoTPTS@AsiaCCS, Xi’an, China, May 30, 2016*. Ed. by Richard Chow and Gökay Saldamli. ACM, 2016, pp. 2–9. DOI: 10.1145/2899007.2899011.
 - [22] Luca Campa. “Algebraic Cryptanalysis and Its Application to Arithmetization Oriented Primitives”. Master’s Thesis. MA thesis. University of Udine and Alpen-Adria-Universität Klagenfurt, 2024. URL: <https://lucacampa.it/assets/pdf/thesis.pdf>.

- [23] David Cash et al. “Bonsai Trees, or How to Delegate a Lattice Basis”. In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 523–552. DOI: 10.1007/978-3-642-13190-5_27.
- [24] Nicolas Courtois et al. “Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations”. In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 392–407. ISBN: 978-3-540-45539-4.
- [25] David A. Cox, John Little, and Donal O’Shea. *Ideals, Varieties, and Algorithms*. Fourth. Undergraduate Texts in Mathematics. Springer, 2015. ISBN: 978-3-319-16720-6. DOI: 10.1007/978-3-319-16721-3.
- [26] David A. Cox, John B. Little, and Donal O’Shea. *Using Algebraic Geometry / by David A. Cox, John Little, DONAL OSHEA*. eng. 1st ed. 1998. Graduate Texts in Mathematics, 185. New York, NY: Springer New York, 1998. ISBN: 1-4757-6911-3.
- [27] Miguel Cueto Noval et al. “On the Soundness of Algebraic Attacks Against Code-Based Assumptions”. In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part VI*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15606. Lecture Notes in Computer Science. Springer, 2025, pp. 385–415. DOI: 10.1007/978-3-031-91095-1_14.
- [28] Jan-Pieter D’Anvers et al. “Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM”. In: *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*. Ed. by Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 10831. Lecture Notes in Computer Science. Springer, 2018, pp. 282–305. DOI: 10.1007/978-3-319-89339-6_16.
- [29] Jean-Charles Faugère. “A new efficient algorithm for computing Gröbner bases (F4)”. In: *Journal of Pure and Applied Algebra* 139.1-3 (June 1999), pp. 61–88. DOI: 10.1016/S0022-4049(99)00005-5.
- [30] Jean-Charles Faugère. “A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)”. In: *International Symposium on Symbolic and Algebraic Computation Symposium - ISSAC 2002*. Colloque avec actes et comité de lecture. internationale. Villeneuve d’Ascq, France: ACM, July 2002, pp. 75–83. DOI: 10.1145/780506.780516.

-
- [31] Jean-Charles Faugère et al. “Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering”. In: *J. Symb. Comput.* 16.4 (1993), pp. 329–344. DOI: 10.1006/JSC0.1993.1051.
 - [32] Ralf Fröberg and Joachim Hollman. “Hilbert Series for Ideals Generated by Generic Forms”. In: *J. Symb. Comput.* 17.2 (1994), pp. 149–157. DOI: 10.1006/JSC0.1994.1008.
 - [33] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. “Trapdoors for hard lattices and new cryptographic constructions”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.
 - [34] Lewis Glabush et al. *FrodoKEM: A CCA-Secure Learning With Errors Key Encapsulation Mechanism*. Cryptology ePrint Archive, Paper 2025/1861. 2025. DOI: 10.62056/ayivom2hd.
 - [35] Shafi Goldwasser et al. “Robustness of the Learning with Errors Assumption”. In: *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*. Ed. by Andrew Chi-Chih Yao. Tsinghua University Press, 2010, pp. 230–240. URL: <http://conference.iis.tsinghua.edu.cn/ICS2010/content/papers/19.html>.
 - [36] Timothy J. Hodges, Sergio D. Molina, and Jacob Schlather. “On the existence of homogeneous semi-regular sequences in $F_2[X_1, \dots, X_n]/(X_1^2, \dots, X_n^2)$ ”. In: *Journal of Algebra* 476 (2017), pp. 519–547. ISSN: 0021-8693. DOI: <https://doi.org/10.1016/j.jalgebra.2016.11.025>.
 - [37] Ravi Kannan. “Minkowski’s Convex Body Theorem and Integer Programming”. In: *Math. Oper. Res.* 12.3 (1987), pp. 415–440. DOI: 10.1287/MOOR.12.3.415.
 - [38] Arjen Lenstra, Hendrik Lenstra, and Lovász László. “Factoring Polynomials with Rational Coefficients”. In: *Mathematische Annalen* 261 (Dec. 1982). DOI: 10.1007/BF01457454.
 - [39] Richard Lindner and Chris Peikert. “Better Key Sizes (and Attacks) for LWE-Based Encryption”. In: *IACR Cryptol. ePrint Arch.* (2010), p. 613. URL: <http://eprint.iacr.org/2010/613>.
 - [40] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On Ideal Lattices and Learning With Errors over Rings”. In: *Advances in Cryptology – EUROCRYPT 2010*. Vol. 6110. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1.

- [41] Daniele Micciancio and Chris Peikert. “Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller”. In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41.
- [42] Chris Peikert. “Public-key cryptosystems from the worst-case shortest vector problem: extended abstract”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*. Ed. by Michael Mitzenmacher. ACM, 2009, pp. 333–342. DOI: 10.1145/1536414.1536461.
- [43] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. “A Framework for Efficient and Composable Oblivious Transfer”. In: *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*. Ed. by David A. Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 554–571. DOI: 10.1007/978-3-540-85174-5_31.
- [44] Chris Peikert and Brent Waters. “Lossy trapdoor functions and their applications”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, 2008, pp. 187–196. DOI: 10.1145/1374376.1374406.
- [45] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.
- [46] Oded Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *J. ACM* 56.6 (2009), 34:1–34:40. DOI: 10.1145/1568318.1568324.
- [47] Claus Schnorr and M. Euchner. “Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems”. In: *Mathematical Programming* 66 (Aug. 1994), pp. 181–199. DOI: 10.1007/BF01581144.
- [48] Matthias Johann Steiner. “The Complexity of Algebraic Algorithms for LWE”. In: *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III*. Ed. by Marc Joye and Gregor Leander. Vol. 14653. Lecture Notes in Computer Science. Springer, 2024, pp. 375–403. DOI: 10.1007/978-3-031-58734-4_13.

- [49] Chao Sun, Mehdi Tibouchi, and Masayuki Abe. “Revisiting the Hardness of Binary Error LWE”. In: *Information Security and Privacy - 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30 - December 2, 2020, Proceedings*. Ed. by Joseph K. Liu and Hui Cui. Vol. 12248. Lecture Notes in Computer Science. Springer, 2020, pp. 425–444. DOI: 10.1007/978-3-030-55304-3_22.

List of Figures

2.1	The standard PoSSo pipeline for algebraic cryptanalysis.	22
3.1	Graphical representation of the straightforward linear-algebraic approach for recovering the LWE secret vector. As a general visual convention for this and following figures: coloured blocks denote dense matrices, white blocks with a filled diagonal denote diagonal matrices, and upper-triangular blocks indicate matrices in Row Echelon Form (REF).	28
3.2	A visual overview of the proposed algorithm’s core mathematical phases. Block colour conventions follow those established in Figure 3.1.	30
3.3	Comprehensive visual summary of the presented algorithm. The diagram illustrates four main steps of the workflow: (1) The initial coefficient matrix representing the polynomial system is transformed into Reduced Row-Echelon Form (RREF). (2) A targeted set of n S -polynomials is generated based on the diagonalized structure. (3) These polynomials are deterministically reduced against the system basis to yield linear relations. (4) A final RREF step is performed on the augmented system to explicitly recover the secret vector $\mathbf{s}$. Block colour conventions follow those established in Figure 3.1.	31
3.4	Overview of the Forward Elimination step, reducing the initial system to Row-Echelon Form. Block colour conventions follow those established in Figure 3.1.	32
3.5	Overview of the Backward Substitution step, yielding the fully diagonalized leading submatrix. Block colour conventions follow those established in Figure 3.1.	33
3.6	Overview of targeted S -polynomial generation.	35
3.7	Overview of the division-free S -polynomial reduction against the diagonal system. Block colour conventions follow those established in Figure 3.1.	35

3.8	Overview of the final diagonalization. The linear subsystem undergoes final REF and RREF steps to explicitly recover the secret vector $\mathbf{s}$. Block colour conventions follow those established in Figure 3.1.	36
4.1	Average recovery time (ms) as a function of the number of variables n over $\mathbb{F}_{257}$. The plot compares our direct approach () with MAGMA's F4 ()	58
4.2	Asymptotic scaling of the average recovery time (ms) over $\mathbb{F}_{257}$. The plot compares the matrix-optimized implementation () against MAGMA's F4 ()	59

List of Tables

2.1	An overview of the complexities of algebraic attacks on LWE with binary errors, a secret $\mathbf{s}$ of length n and m samples. $H_2(x) = -x \log x - (1-x) \log(1-x)$ is the binary entropy function, $\ln$ is the natural logarithm, ω is the linear algebra constant ($2 \leq \omega \leq 3$), and $\beta = c - 0.5 - \sqrt{c(c-1)}$. The last two columns indicate whether the bounds rely on unproven theoretical assumptions (such as semi-regularity or probabilistic estimates) and/or heuristic estimates of the degree of regularity d_{reg} were utilized in formulating the complexity bounds.	24
3.1	Summary of worst-case computational complexities per phase. . .	51
3.2	Comparative overview of algebraic attacks on LWE with binary errors. Refer to Table 2.1 for detailed definitions of the notation and parameters used.	52
4.1	Average execution time (ms) per instance of our direct implementation versus MAGMA's native F4 over $\mathbb{F}_{257}$	58
4.2	Performance comparison between the matrix-optimized algorithm and MAGMA's F4. As the dimension scales, the $\mathcal{O}(n^6)$ complexity of our algorithm overcomes the initial constant-factor efficiency of F4, showing better performance for $n \geq 80$	60
4.3	Empirical success rates for a fixed dimension $n = 10$ across varying prime moduli. The algorithm recovers the secret vector consistently, demonstrating that the full-rank condition is easily satisfied even for minimal values of q	62

