Cryptanalysis of Symmetric Encryption Algorithm in a Polynomial Residue Number System

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Technology (B.Tech)

by

Rahul Patel (21CSB0F13)

Venkata Sai Sidhartha Ratnakaram (21CSB0F01)

Anshu Kumar (21CSB0A05)

Supervisor:

Dr. R Padmavathy

Professor

Department of Computer Science and Engineering

NIT Warangal

2021-25

Acknowledgement

We would like to express our heartily gratitude towards Prof. R Padmavathy, CSE Department, for her valuable guidance, supervision, suggestions, encouragement and the help throughout the year and also for the completion of our project work. She kept us going when we were down and gave us the courage to keep moving forward.

We would like to take this opportunity once again to thank Dr. R. Padmavathy, Head of the Department, Computer Science and Engineering, NIT Warangal for giving us this opportunity and resources to work on this project and supporting through out. We also want to thank evaluation committee for their valuable suggestions on my proposals and research and for conducting smooth presentation of the project.

This year-long project was a great learning experience for us and we can say for certain that we have grown leaps and bounds in our scientific thinking skills, critical analysis skills, and research calibre.

Rahul Patel 21CSB0F13

Venkata Sai Sidhartha Ratnakaram 21CSB0F01

Anshu Kumar 21CSB0A05

Date:

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2024-25

APPROVAL SHEET

The Project Work entitled of Symmetric Encryption Algorithm in a Polynomial Residue Number System by **Rahul Patel (21CSB0F13), Venkata Sai Sidhartha Ratnakaram (21CSB0F01), Anshu Kumar (21CSB0A05),** is approved for the degree of Bachelor of Technology (B.Tech) in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor

_____

Dr. R Padmavathy
Professor

Head of the Department

_____

Dr. R. Padmavathy
Professor

Date: _____

Place: NIT, Warangal

Declaration

We declare that this written submission represents our ideas, my supervisor's ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Rahul Patel	Venkata Sai Sidhartha Ratnakaram	Anshu Kumar
21CSB0F13	21CSB0F01	21CSB0A05

Date:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2023-24

Certificate

This is to certify that the Dissertation work entitled **Cryptanalysis of Symmetric Encryption Algorithm in a Polynomial Residue Number System** is a bonafide record of work carried out by **Rahul Patel (21CSB0F13), Venkata Sai Sidhartha Ratnakaram (21CSB0F01), Anshu Kumar (21CSB0A05)**, submitted to the Prof. R Padmavathy of Department of Computer Science and Engineering, in partial fulfilment of the requirements for the award of the degree of B.Tech at National Institute of Technology, Warangal during the 2024-2025.

(Signature)
Dr. R. Padmavathy
Professor
Head of the Department
Department of Computer Science and Engineering
NIT Warangal

(Signature)
Dr. R. Padmavathy
Professor
Department of Computer Science and Engineering NIT Warangal

Abstract

This paper presents a cryptanalysis of a recently proposed symmetric encryption algorithm built on the Polynomial Residue Number System (PRNS). The original scheme claims enhanced security through the use of arbitrarily selected polynomials and pairwise coprime residues as cryptographic keys, leveraging the method of undetermined coefficients for polynomial reconstruction. Our analysis critically evaluates the theoretical assumptions of the algorithm, examining its resistance to known cryptographic attacks, including brute-force, algebraic, and structural cryptanalysis. We identify potential vulnerabilities arising from key selection randomness, residue base structure, and polynomial factorization. By modeling the key-recovery problem, we assess whether the claimed NP-completeness holds under practical conditions. Additionally, we explore scenarios where side-channel information or partial knowledge of the polynomial residues could significantly reduce the effective key space. Experimental results and theoretical arguments are provided to support our findings, offering insights into the algorithm's actual cryptographic strength and suggesting improvements for future iterations.

Keywords —- ciphertext; cryptanalysis; cryptoalgorithm; cryptographic strength; residue number system; symmetric cryptosystem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

With the growing need for secure and efficient data transfer in contemporary digital systems, cryptographic primitives remain an indispensable part of protecting information. Symmetric encryption algorithms are still the foundation of high-throughput communication security because of their speed in computation and simple structure. Nonetheless, progress in algebraic frameworks, especially involving polynomial rings and modular arithmetic, has led to novel cryptographic constructions. One such development is the Polynomial Residue Number System (PRNS), which employs polynomial modular operations for encryption and decryption.

In 2024, Yakymenko et al.[1] proposed a symmetric encryption scheme that uses PRNS, with the goal of providing enhanced performance and security by means of the application of irreducible polynomials and reconstruction using Chinese Remainder Theorem (CRT)-based. Their method posits computational hardness based on NP-complete issues due to the difficulty in factorization of polynomials.

1.2 Motivation

While the PRNS-based scheme of Yakymenko et al.[1] holds the promise of computational efficiency and theoretical security, it is important to carefully test its claims of security. Since new cryptographic primitives have a tendency to conceal unexpected weaknesses behind apparently difficult mathematical problems, careful cryptanalysis is required to establish the actual resilience of the scheme. Also, since the system's security relies on assumptions about irreducible polynomial choice and modular reconstruction, it becomes critical to experimentally verify that these assumptions hold under different attack models. Understanding its limitations and strengths better will guide its practical applicability and future enhancements or uses.

1.3 Problem Statement

The Yakymenko et al.[1] encryption scheme argues for good security based on modular polynomial arithmetic employed in a PRNS setting. Yet, the hardness assumptions for cryptanalysis, especially recovering irreducible polynomials and parameters from modular data, have not been rigorously substantiated. Lacking systematic cryptanalysis, there is no certainty that the scheme can resist brute-force, structural, or algebraic attacks. Hence, the issue is how to evaluate the cryptographic resistance of this PRNS-based symmetric cryptography scheme based on the analysis of its key structure, encryption/decryption process, and vulnerability to standard attack models.

1.4 Objectives

1. To provide a detailed review of the symmetric encryption scheme proposed by Yakymenko et al.[1], focusing on its mathematical structure and operational principles.

2. To analyze the security assumptions made by the authors, particularly regarding the use of irreducible polynomials and CRT-based reconstruction.

3. To develop cryptanalysis cases targeting different components of the scheme, such as key recovery, plaintext leakage, and polynomial reconstruction.

4. To evaluate the feasibility of potential attack strategies, including brute-force, algebraic, and structural attacks, under realistic conditions.
5. To propose improvements or highlight vulnerabilities based on the cryptanalytic findings, contributing to the ongoing development of secure PRNS-based encryption systems.

1.5 Thesis Organization

This thesis is organized into the following chapters:

- **Chapter 1: Introduction -**Provides an overview of the project, including the background, motivation, problem statement, and objectives.
- **Chapter 2: Related Work -** Discusses existing works and research related to polynomial factorization, algebraic techniques and lattice-based algorithms.
- **Chapter 3: Proposed methodology -**Describes the target cryptosystem and the methodology adopted to develop cryptanalytic cases against it.
- **Chapter 4: Results and Discussion -** This section presents the findings and accuracy analysis of the proposed cryptanalysis cases, and outlines enhancements to improve the robustness of the original cryptosystem.
- **Chapter 5: Conclusion and Future Work -** This section concludes the cryptanalysis findings and outlines future work, including formalizing results and drafting a research paper.

Chapter 2

Literature Survey

The survey of literature reviews earlier work in cryptanalysis of symmetric encryption schemes based on polynomials, with an emphasis on those constructed using the Residue Number System (RNS) and polynomial operations. It analyzes several attack models for such schemes, such as brute-force search against sets of irreducible polynomials, algebraic attacks that take advantage of weaknesses in key reconstruction processes, and complexity analysis under the Chinese Remainder Theorem (CRT). Particular emphasis is placed on problems of breaking schemes based on combinatorial keyspaces and on modular polynomial manipulations. The survey also touches on theoretical approaches to measuring cryptography strength by using NP-completeness and opposition to inversion-based attacks. Below is a compilation of the respective cryptanalytic work.

Yakymenko et al. [2] presents theoretical backgrounds of the symmetric encryption based on the residue number system. The peculiarities of this approach are that in the case of restoring a decimal number based on its residuals using the Chinese remainder theorem, multiplication occurs by arbitrarily chosen coefficients (keys). It is established that cryptostability of the developed methods is determined by the number of modules and their bit size. In addition, the described methods are found to allow to almost indefinitely increase the block of plain text for encryption, which eliminates the necessity to use dif-

ferent encryption modes.

Qinnan Luo et al. [3] shows some research and applications based on the theorem. For example, the theorem in polynomial form, the theorem in the form of group theory, the theorem on unitary rings, the theorem on polynomial ring modules, etc. It is not difficult to know that integers and polynomials are special rings, so this the two forms of the theorem are the theorems that can be covered on the unitary ring. In fact, the theorem in the form of group theory is also covered. [3] elaborates the first three forms of the theorem and give their specific applications.

Hoffstein et al. [4] features reasonably short, easily created keys, high speed, and low memory requirements. NTRU encryption and decryption use a mixing system suggested by polynomial algebra combined with a clustering principle based on elementary probability theory. The security of the [4] comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers p and q.

N. K. Tiwari et al. [5] provide a survey of fundamental cryptanalytic techniques applicable across both classical and modern cryptosystems . The paper outlines core methods such as brute-force, differential, linear, and algebraic attacks, with a particular focus on how these methods scale with the complexity of key structure and system design.This work offers foundational insight into how algebraic methods like lattice-based reduction and symbolic equation solving are generalized and adapted in practical cryptanalysis

Jakhar et al. [6] investigate the factorization properties of polynomials over the integers, with a particular focus on irreducibility in polynomial rings. The authors explore conditions under which the irreducible factors of a given polynomial remain irreducible in extensions and analyze their behavior in various algebraic structures. A significant portion of the work revolves around understanding the implications for reducibility when moving from $\mathbb{Z}[x]$ to more general domains.

Berlekamp et al. [7] introduced one of the first efficient algorithms for factoring polynomials over finite fields. This foundational work laid the groundwork for subsequent advancements in computational algebra and remains a cornerstone in symbolic computation and cryptography. The algorithm transforms the problem of polynomial factorization into a linear algebra problem by constructing what is now known as the Berlekamp Q-matrix, derived from the Frobenius map $f(x) \mapsto f(x)^q$. The null space of this matrix reveals non-trivial

factors of the polynomial. Berlekamp's approach provided a polynomial-time algorithm for finite fields of small characteristic, significantly improving upon prior brute-force techniques.

Cantor et al [8] introduced a new probabilistic algorithm for factoring polynomials over finite fields, which significantly improves upon previous deterministic methods in terms of efficiency and practical performance. The paper lays the foundation for what is now commonly referred to as the Cantor–Zassenhaus algorithm, which relies on randomization to simplify the factorization process. The method is particularly notable for its applicability to large polynomials in cryptographic and coding theory contexts. By using properties of finite fields and leveraging randomness to split polynomials into their irreducible components, this algorithm remains a key tool in computational algebra systems.

Miller et al [9] explore a novel enhancement to Coppersmith's lattice-based method for attacking small-exponent RSA. Their work introduces a technique that systematically reduces the lattice and matrix size in such attacks by drawing from observed patterns in smaller parameter regimes—analogous to "focus group" testing.This approach improves both the efficiency and success rate of recovering the RSA secret key in vulnerable instances. Furthermore, the paper highlights how certain lattice basis reduction algorithms,show exceptional promise when integrated with Coppersmith's method. These refinements significantly extend the practical reach of small-exponent RSA attacks. These attacks could be modified for the cryptanalysis of our current cryptosystem

Herrmann et al [10] present a novel lattice-based cryptanalytic technique that solves linear modular equations by exploiting partial knowledge of inputs—such as bits of a factor or plaintext. Their method generalizes earlier Coppersmith-type attacks and provides practical improvements for factoring and RSA key recovery under certain leakage conditions.

Bruno Buchberger et al [11] presents an algorithmic way to convert an arbitrary generating set of a polynomial ideal into a structured Gröbner basis. This transformation simplifies many operations in computational algebra, such as solving systems of polynomial equations, ideal membership testing, and elimination theory.In symmetric cryptosystems where the encryption and decryption processes are modeled using polynomial maps, Buchberger's algorithm can be used to recover unknown plaintext coefficients from encrypted polynomial equations. By expressing the encryption as a multivariate polynomial system with partial knowns (e.g., some coefficients or intermediate results), Gröbner basis computation allows us to reduce the system to a solvable form, effectively per-

forming a structured algebraic cryptanalysis. This is particularly useful in cases where:

- The plaintext is represented as a polynomial with known and unknown coefficients.
- The secret key is embedded in multivariate polynomials.

The paper presented by Jakobsen et al [12] targets block ciphers with low-degree polynomial structure. It uses known plaintext-ciphertext pairs to interpolate the encryption polynomial.Recover the cipher's function as a polynomial over a finite field. Impact on symmetric polynomial cryptosystems: If encryption functions are of low algebraic degree, they are vulnerable.

Md. Kamal Hossain et al [13] Lattice-Based Cryptanalysis of RSA-Type Cryptosystems: A Bibliometric Analysis This paper provides a comprehensive bibliometric analysis of the research landscape focused on lattice-based attacks on RSA-type cryptosystems. It traces the development of these attacks starting from which introduced groundbreaking techniques for solving polynomial equations modulo integers, specifically applied to cryptanalysis of RSA. The paper tracks the growth of lattice-based cryptanalysis in the context of RSA, showcasing the significant role of lattice methods in breaking RSA cryptosystems under certain conditions. It highlights how research on this topic has grown over the years, with a sharp increase in publications post-2010. The authors also examine the academic impact and collaboration networks between leading researchers, institutions, and countries, shedding light on the collaborative nature of this field. The study emphasizes that lattice-based cryptanalysis has evolved to target new variants of RSA, including those with small exponents or partial key information, which can be attacked using lattice reduction techniques.

2.1 Summary of Existing Works

Authors and Paper Name	Work Done
Yakymenko et al. [2] – *Symmetric Encryption Based on the Residue Number System*	Describes RNS-based symmetric encryption using CRT. Highlights how modular structure and key coefficients affect cryptostability. Emphasizes scalability of plaintext blocks eliminating the need for encryption modes.
Qinnan Luo et al. [3] – *Applications of the Chinese Remainder Theorem*	Analyzes CRT in forms such as group theory, unitary rings, and polynomial modules. Shows generalization of CRT to special rings like integers and polynomials.
Hoffstein et al. [4] – *NTRU Encryption Algorithm*	Introduces NTRU, a lattice-based scheme using polynomial algebra and modular reduction. Offers short keys, fast encryption, and low memory usage.
N. K. Tiwari et al. [5] – *Survey of Cryptanalysis Techniques*	Surveys classical and modern cryptanalytic methods. Focuses on brute-force, differential, linear, algebraic, and lattice-based attacks with symbolic solving.
Jakhar et al. [6] – *Irreducibility in Polynomial Rings*	Investigates polynomial irreducibility over integers and extensions. Studies implications for algebraic structures in cryptographic settings.
Berlekamp et al. [7] – *Factoring Polynomials Over Finite Fields*	Introduces Berlekamp's algorithm using Frobenius map and linear algebra. Provides efficient factorization in small characteristic fields; foundational in cryptography.
Cantor et al. [8] – *Cantor–Zassenhaus Algorithm*	Presents a randomized polynomial factorization method over finite fields. Effective for large polynomials in coding and cryptography.
Miller et al. [9] – *Enhancing Coppersmith's Method*	Improves lattice-based attacks on small-exponent RSA. Reduces matrix size for better performance; applicable to cryptosystems with similar weaknesses.
Herrmann et al. [10] – *Solving Linear Modular Equations with Partial Knowledge*	Generalizes Coppersmith-type attacks using lattice reduction. Leverages partial data (e.g., bits of plaintext) for practical RSA cryptanalysis.
Bruno Buchberger et al. [11] – *Gröbner Basis Algorithm*	Develops a method to compute Gröbner bases for solving multivariate polynomial systems. Applies to algebraic cryptanalysis with partially known data.
Jakobsen et al. [12] – *Cryptanalysis of Low-Degree Polynomial Ciphers*	Uses interpolation of plaintext-ciphertext pairs to reconstruct encryption polynomials. Demonstrates vulnerability of low-degree symmetric ciphers.
Md. Kamal Hossain et al. [13] – *Lattice-Based Cryptanalysis of RSA: A Bibliometric Analysis*	Reviews the evolution and impact of lattice attacks on RSA. Tracks growth post-2010 and highlights academic collaboration in this field.

Table 2.1: Summary of Literature Review

Chapter 3

The proposed approach

In this chapter, we provide the foundational context for the target cryptosystem and explore various scenarios relevant to its cryptanalysis.

3.1 Overview of the Target Cryptosystem

This section describes the cryptographic system proposed by Yakymenko et al. [1], which serves as the basis for our analysis. We conduct a cryptanalysis of the scheme to evaluate its security and identify potential weaknesses. Furthermore, the development and refinement of the existing cryptosystem are presented in detail, highlighting improvements in both efficiency and robustness. The following discussion outlines the structure, encryption, and decryption processes involved in the scheme.

3.1.1 Theoretical Foundation of Polynomial RNS

In a Polynomial Residue Number System (PRNS), a polynomial $N(x)$ is represented using modular arithmetic with respect to a set of pairwise coprime

polynomials $p_1(x), p_2(x), \ldots, p_s(x)$. This allows efficient computations by operating on smaller degree polynomials (residues) independently. [1]

The representation of $N(x)$ in PRNS is as follows:

$$b_i(x) = N(x) \mod p_i(x), \quad \text{for } i = 1, 2, \ldots, s \tag{3.1}$$

Here, $b_i(x)$ is the residue of $N(x)$ modulo $p_i(x)$.

To reconstruct the original polynomial $N(x)$ from the set of residues $\{b_i(x)\}$, the **Chinese Remainder Theorem (CRT)** in the ring of polynomials $\mathbb{Z}[x]$ is used: [1]

$$N(x) = \left(\sum_{i=1}^{s} b_i(x) \cdot M_i(x) \cdot m_i(x) \right) \mod P(x) \tag{3.2}$$

where:

$$P(x) = \prod_{i=1}^{s} p_i(x) \tag{3.3}$$

$$M_i(x) = \frac{P(x)}{p_i(x)} \tag{3.4}$$

$$m_i(x) := M_i^{-1}(x) \mod p_i(x) \tag{3.5}$$

Each $m_i(x)$ is the modular inverse of $M_i(x)$ modulo $p_i(x)$, satisfying:

$$M_i(x) \cdot m_i(x) := 1 \mod p_i(x)$$

For correct reconstruction, the degree of $N(x)$ must be strictly less than the degree of the product polynomial $P(x)$:

$$\deg(N(x)) < \deg(P(x)) \tag{3.6}$$

3.1.2 Development of Polynomial Symmetric Encryption Methods in the RNS

In the enhanced version of the Polynomial Residue Number System (PRNS), encryption does not rely on the standard modular inverses $m_i(x)$, but

instead utilizes arbitrary polynomials $k_i(x)$ for added security. These polynomials are chosen such that they remain coprime to their respective modulus polynomials $p_i(x)$, and are kept secret between the sender and the receiver. [1]

The conditions for selecting such parameters are:

$$\deg(N(x)) < \deg(P(x)), \quad \gcd(k_i(x), p_i(x)) = 1 \tag{3.7}$$

Each party involved computes the Chinese Remainder Theorem (CRT) parameters as follows:[1]

$$M_i(x) = \frac{P(x)}{p_i(x)}, \quad m_i(x) := M_i^{-1}(x) \mod p_i(x) \tag{3.8}$$

Using the known residues $b_i(x) = N(x) \mod p_i(x)$, the encryption process is carried out by computing:

$$N'(x) = \left(\sum_{i=1}^{s} b_i(x) \cdot M_i(x) \cdot k_i(x) \right) \mod P(x) \tag{3.9}$$

Here, $N'(x)$ represents the ciphertext polynomial, which is transmitted securely to the receiver.

For decryption, the receiver first computes the auxiliary value $q_i(x)$, which incorporates both the modular inverse of $M_i(x)$ and the inverse of $k_i(x)$: [1]

$$q_i(x) = \left(m_i(x) \cdot k_i^{-1}(x) \mod p_i(x) \right) \tag{3.10}$$

Then, the encrypted residue modulo $p_i(x)$ is determined as:

$$b_i'(x) = N'(x) \mod p_i(x) \tag{3.11}$$

To recover the original residue $b_i(x)$, the following relation is used:

$$b_i(x) = \left(b_i'(x) \cdot q_i(x) \right) \mod p_i(x) \tag{3.12}$$

This can also be expressed by substituting $q_i(x)$:

$$b_i(x) = \left(b_i'(x) \cdot m_i(x) \cdot k_i^{-1}(x) \right) \mod p_i(x)$$

Once all original residues $b_i(x)$ are reconstructed, the plaintext polynomial $N(x)$ is obtained using the standard CRT reconstruction: [1]

$$N(x) = \left(\sum_{i=1}^{s} M_i(x) \cdot m_i(x) \cdot b_i(x) \right) \mod P(x) \tag{3.13}$$

Alternatively, combining earlier expressions gives a decryption formula directly from encrypted components:

$$N(x) = \left(\sum_{i=1}^{s} M_i(x) \cdot m_i(x) \cdot b'_i(x) \cdot q_i(x) \right) \mod P(x) \tag{3.14}$$

This method ensures secure reconstruction of the original message $N(x)$, leveraging both secret parameters and CRT properties.

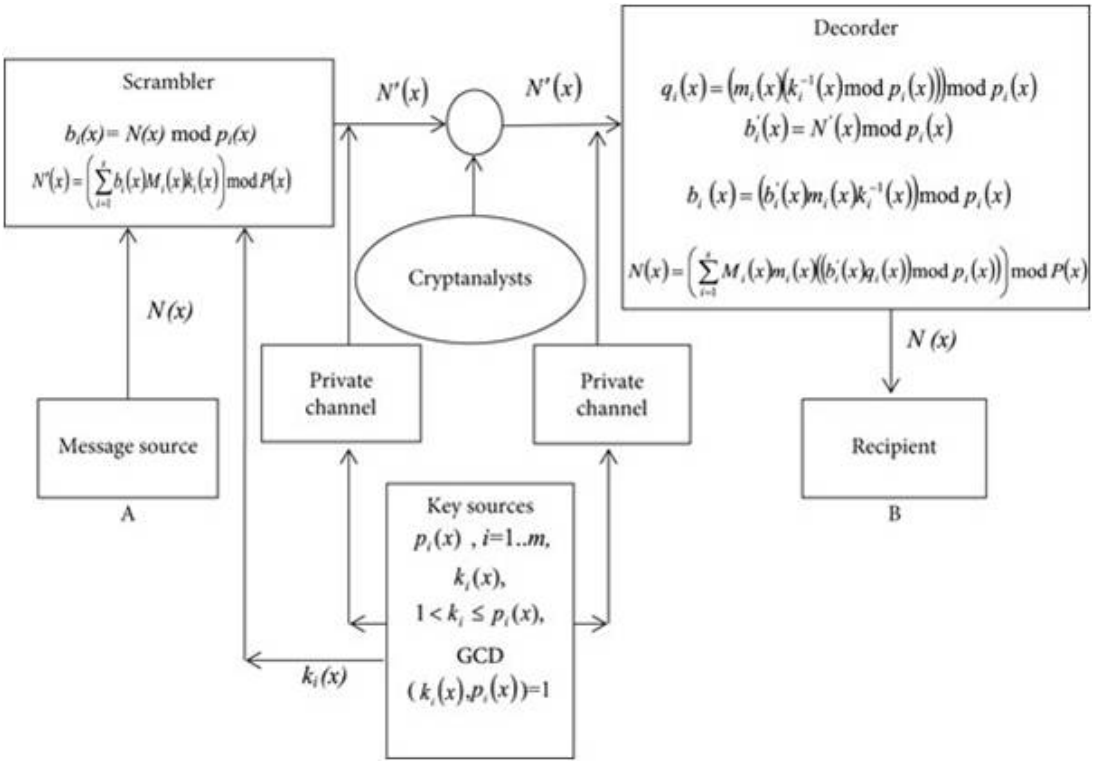

Figure 3.1: Scheme of the target polynomial symmetric encryption in RNS [1].

3.2 Formulation of Cryptanalysis Cases

To assess the security of the given cryptosystem, we have developed several distinct cases for cryptanalysis. Each case targets specific structural aspects of

the system, such as the properties of the modulus polynomials, the nature of the plaintext, the lattice structure and the availability of partial information to an attacker. These cases are designed to explore potential vulnerabilities under different assumptions and attack scenarios, providing a comprehensive evaluation of the cryptosystem's resilience.

3.2.1 Polynomial Modulo Attack

This section presents a cryptanalytic approach aimed at recovering the secret irreducible polynomial $p(x)$, which is central to the security of the proposed cryptosystem. The cryptosystem operates over the finite field $\mathbb{GF}(p)$, and encryption involves computing (3.1)

$$b(x) = N(x) \mod p(x)$$

where $N(x)$ is the plaintext polynomial and $b(x)$ is the encrypted output or residual. The adversary's objective is to reconstruct the secret polynomial $p(x)$ by leveraging the known plaintext and its corresponding residual.

Adversary model: The attack model assumes that the adversary has access to both the plaintext polynomial $N(x)$ and the corresponding residual $b(x)$. Using these, the adversary computes the difference polynomial

$$d(x) = N(x) - b(x) \tag{3.15}$$

Since (3.1) $b(x) = N(x) \mod p(x)$, the subtraction yields $d(x)$, which is divisible by the secret polynomial $p(x)$. In other words, $p(x) \mid d(x)$. The critical insight here is that factorizing $d(x)$ can reveal $p(x)$ as one of the factors, provided that the correct degree and irreducibility conditions are met.

Security Assumptions: The security of the system relies on the assumption that $p(x)$ is irreducible over $\mathbb{GF}(p)$, and that the field size p and degree n are sufficiently large to prevent brute-force or enumeration-based attacks. However, if either p or n is small, the number of possible irreducible polynomials is reduced, making the system vulnerable to polynomial-time attacks using efficient factorization algorithms.

Theoretical cryptanalysis: From a theoretical standpoint, the attack proceeds in three primary steps. First, the adversary computes the difference poly-

nomial according to (3.15) , which is computationally trivial and can be done in linear time with respect to the polynomial degree. Second, $d(x)$ is factorized using the Cantor-Zassenhaus algorithm [8], a probabilistic algorithm with expected time complexity $O(n^3)$. Third, the adversary filters out the factors that are irreducible and of degree n, as these are the most likely candidates for $p(x)$.

The success of the attack depends on the number of irreducible degree-n factors in $d(x)$. If there are k such factors, the probability of randomly selecting the correct one is $\frac{1}{k}$. This probability increases as k decreases, and is particularly dangerous when $d(x)$ has very few irreducible factors of degree n.

The total number of irreducible polynomials of degree n over $\mathbb{GF}(p)$ is given by the formula:

$$I(p,n) = \frac{1}{n} \sum_{d \mid n} \mu(d) \cdot p^{n/d} \tag{3.16}$$

where $\mu(d)$ is the Möbius function. In general, this value is approximately $\frac{p^n}{n}$, which represents the size of the search space. This bound helps quantify the security margin provided by the choice of p and n. Smaller values of p or n substantially reduce the space, hence increasing the risk of successful cryptanalysis.

Algorithm: Cryptanalysis of the Proposed Cryptosystem (Case-1)

Input: Plaintext polynomial $N(x)$, residual $b(x) = N(x) \bmod p(x)$
Output: Candidate irreducible polynomial(s) $p(x)$
Step 1: Compute the difference polynomial:

$$d(x) \leftarrow N(x) - b(x)$$

Step 2: Factor $d(x)$ over $\mathbb{F}_p$ using the Cantor-Zassenhaus algorithm:

$$\texttt{Factors} \leftarrow \texttt{CantorZassenhaus}(d(x))$$

Step 3: Filter the factors to find irreducible polynomials of degree n:

$$\texttt{Candidates} \leftarrow \{f(x) \in \texttt{Factors} \mid \deg(f(x)) = n \text{ and } f(x) \text{ is irreducible}\}$$

Step 4: For each candidate $f(x) \in \texttt{Candidates}$, check if it satisfies:

$$N(x) \bmod f(x) = b(x)$$

If true, return $f(x)$ as the valid $p(x)$.

Cantor-Zassenhaus factorization algorithm: The Cantor-Zassenhaus algorithm is a **probabilistic algorithm** for factoring polynomials over a finite field $\mathbb{F}_p$, where p is a prime number. Given a square-free polynomial $f(x) \in \mathbb{F}_p[x]$, the goal is to find its irreducible factors.

The algorithm relies on repeatedly applying random polynomial selection, greatest common divisor (GCD) computation, polynomial division, and recursive factorization. It is efficient and particularly useful for factoring polynomials of moderate degree over small finite fields. [8]

Algorithm: Cantor-Zassenhaus Algorithm

A square-free polynomial $f(x) \in \mathbb{F}_p[x]$ A list of irreducible factors of $f(x)$
Step 1: Check if $\deg(f(x)) = 1$. If true, return $[f(x)]$.
Step 2: Initialize an empty list `Factors` to store irreducible factors.
Step 3: While $f(x)$ is reducible, repeat the following steps:
Step 3.1: Choose a random polynomial $g(x) \in \mathbb{F}_p[x]$, with $\deg(g) < \deg(f(x))$.
Step 3.2: Compute the greatest common divisor:

$$d(x) = \gcd(f(x), g(x))$$

Step 3.3: If $1 < \deg(d(x)) < \deg(f(x))$, then: Append $d(x)$ to `Factors`. Update $f(x) \leftarrow \frac{f(x)}{d(x)}$.
Step 3.4: If $d(x) = 1$ or $d(x) = f(x)$, go back to Step 3.1 and choose a new $g(x)$.
Step 4: Append the remaining $f(x)$ to `Factors`.
Step 5: Return the list `Factors`.

3.2.2 Polynomial Factorization and Parameter Recovery in GF(p)

The objective of this cryptanalysis procedure is to recover the hidden parameters $p_i(x)$ and $k_i(x)$ from a composite polynomial $P(x)$, which is assumed to be constructed as the product of several irreducible polynomials over the finite field $\mathbb{F}_p$. The process leverages polynomial factorization, modular arithmetic, and known plaintext-ciphertext pairs to reconstruct the internal structure of the cryptosystem.

Factorization: Initially, the polynomial $P(x)$ is factored into its irreducible components using algorithms as Berlekamp [7], which are efficient for fields of small characteristic. Once the factors $p_i(x)$ are obtained, the corresponding quotient polynomials are computed according to (3.4) and (3.5) respsectively.

$$M_i(x) = \frac{P(x)}{p_i(x)}$$

$$m_i(x) := M_i^{-1}(x) \mod p_i(x)$$

Computation: To extract the key polynomials $k_i(x)$, known plaintext-ciphertext pairs $(N_j(x), N'_j(x))$ are utilized. The values $b_{i,j}(x) = N_j(x) \mod p_i(x)$ and $b'_{i,j}(x) = N'_j(x) \mod p_i(x)$ are computed, representing the encryption residues before and after encryption. The key component is then derived via:

$$k_i(x) := b'_{i,j}(x) \cdot m_i(x) \cdot b_{i,j}^{-1}(x) \mod p_i(x) \tag{3.17}$$

Success condition: A valid solution requires that $b_{i,j}(x)$ be invertible modulo $p_i(x)$, and at least one known plaintext-ciphertext pair must be available. The overall computational complexity is $O(n^2)$ for each modular inverse and $O(s \cdot n^2)$ to recover all $k_i(x)$, where s is the number of irreducible factors.

Given known pairs $(N_j(x), N'_j(x))$, calculate according to (3.1) and (3.11) respectively.

$$b_{i,j}(x) = N_j(x) \bmod p_i(x), \quad b'_{i,j}(x) = N'_j(x) \bmod p_i(x)$$

Then solve according to (3.17)

$$k_i(x) := b'_{i,j}(x) \cdot m_i(x) \cdot b_{i,j}^{-1}(x) \mod p_i(x)$$

A unique solution is possible when $b_{i,j}(x)$ is invertible modulo $p_i(x)$, and at least one known plaintext-ciphertext pair is required. The overall complexity is $O(n^2)$ per factor for inversion, and $O(s \cdot n^2)$ to recover all $k_i(x)$.

Algorithm: Recovery of Secret Parameters

Input: Composite polynomial $P(x)$, known plaintext-ciphertext pairs $\{(N_j(x), N'_j(x))\}$
Output: Recovered irreducible factors $\{p_i(x)\}$ and key polynomials $\{k_i(x)\}$
Step 1: Factor $P(x)$ over $\mathbb{F}_p$:

$$P(x) = \prod_{i=1}^{s} p_i(x)$$

Step 2: For each $p_i(x)$, compute:

$$M_i(x) = \frac{P(x)}{p_i(x)}$$

Step 3: Compute modular inverse:

$$m_i(x) := M_i^{-1}(x) \mod p_i(x)$$

Step 4: For each known pair $(N_j(x), N'_j(x))$, compute:

$$b_{i,j}(x) = N_j(x) \mod p_i(x), \quad b'_{i,j}(x) = N'_j(x) \mod p_i(x)$$

Step 5: Solve for key polynomial:

$$k_i(x) := b'_{i,j}(x) \cdot m_i(x) \cdot b_{i,j}^{-1}(x) \mod p_i(x)$$

Step 6: Return all $p_i(x)$ and $k_i(x)$

Berlekamp's algorithm is an efficient method for factoring square-free polynomials over a finite field $\mathbb{F}_p$. The key idea is to reduce the factorization problem to a problem of finding the null space of a linear transformation defined over the finite field.

The algorithm begins by ensuring the input polynomial is square-free by computing the greatest common divisor of the polynomial and its derivative. Then, it constructs the so-called Berlekamp matrix, whose null space corresponds to the space of polynomials that satisfy a certain congruence. Each non-trivial solution in this null space can be used to extract a non-trivial factor of the original polynomial through GCD computations. The process is recursively applied to each factor until all irreducible components are found.[7]

Berlekamp's algorithm is particularly efficient for polynomials over small

fields and has a complexity of approximately $O(n^3)$, where n is the degree of the polynomial. [7]

Algorithm: Berlekamp Polynomial Factorization

Input: A square-free polynomial $f(x) \in \mathbb{F}_p[x]$
Output: Irreducible factors of $f(x)$
Step 1: Ensure the polynomial is square-free:

$$\text{If } \gcd(f(x), f'(x)) \neq 1, \text{ divide out the common factor}$$

Step 2: Construct the Berlekamp matrix $Q \in \mathbb{F}_p^{n \times n}$, where $n = \deg f(x)$
Each entry $Q_{i,j}$ is the coefficient of x^j in $x^{ip} \mod f(x)$, for $0 \leq i, j < n$
Step 3: Compute the null space of $Q - I$, where I is the identity matrix:

$$(Q - I)v = 0$$

Each non-trivial solution v defines a polynomial $g(x) \in \mathbb{F}_p[x]$
Step 4: Use each $g(x)$ to factor $f(x)$:

$$\text{For each } c \in \mathbb{F}_p, \text{ compute } \gcd(f(x), g(x) - c)$$

If the result is a non-trivial divisor, it is a factor of $f(x)$
Step 5: Recursively apply the algorithm to each non-trivial factor until all factors are irreducible
Step 6: Return all irreducible factors of $f(x)$

3.2.3 Coppersmith-Based Cryptanalysis for Partial Plaintext Recovery

Coppersmith's method is a mathematical technique used to find small solutions to modular polynomial equations. It was introduced by Don Coppersmith in the 1990s and has become a fundamental tool in cryptanalysis, especially in breaking RSA variants and attacking systems where a small portion of the secret (like a small polynomial root) is leaked or constrained.[9]

In this attack scenario, some information about the plaintext polynomial $N(x)$ is leaked. The attacker's goal is to recover the full plaintext efficiently. The attacker is assumed to know part of the plaintext, some modular reductions (residues), and the polynomials used in the modular reductions. The approach relies on the assumption that the unknown part of the plaintext is a small poly-

nomial.

Attacker's Knowledge: The attacker knows two moduli $p_1(x)$, $p_2(x)$, part of the plaintext polynomial $g(x)$, and the modular reductions $b_1(x), b_2(x)$. The full plaintext is assumed to be:

$$N(x) = g(x) + r(x)$$

where $r(x)$ is a small, unknown error polynomial.

The cryptosystem provides modular reductions:

$$N(x) := b_1(x) \mod p_1(x)$$

$$N(x) := b_2(x) \mod p_2(x)$$

Substituting $N(x) = g(x) + r(x)$ gives:

$$g(x) + r(x) := b_1(x) \mod p_1(x)$$

$$g(x) + r(x) := b_2(x) \mod p_2(x)$$

Rewriting for $r(x)$, we get:

$$r(x) := b_1(x) - g(x) \mod p_1(x)$$

$$r(x) := b_2(x) - g(x) \mod p_2(x)$$

Using the Chinese Remainder Theorem (CRT), combine these into:

$$r(x) := d(x) \mod P(x)$$

where $P(x) = p_1(x) \cdot p_2(x)$ and $d(x)$ is computed accordingly.

Since $r(x)$ is small, it is possible to find it as a small root of the polynomial:

$$f(x) = r(x) - d(x)$$

modulo $P(x)$, using Coppersmith's method.

Coppersmith's Method Overview: Coppersmith's method is designed to find small roots of modular polynomial equations. The idea is to construct a lattice from shifted polynomials of $f(x)$, reduce the lattice using the LLL algorithm, and extract small roots from the resulting short vectors. This is effective when the root is small in comparison to the modulus.[9]

Algorithm: Partial Plaintext Recovery via Coppersmith's Method

Input:
Known part of plaintext polynomial $g(x)$
Known moduli $p_1(x), p_2(x)$
Modular reductions $b_1(x), b_2(x)$
Output: Recovered plaintext polynomial $N(x)$
Step 1: Assume the structure $N(x) = g(x) + r(x)$
Step 2: Compute differences:

$$d_1(x) = b_1(x) - g(x) \mod p_1(x)$$

$$d_2(x) = b_2(x) - g(x) \mod p_2(x)$$

Step 3: Use CRT to compute:

$$r(x) := d(x) \mod P(x), \quad \text{where } P(x) = p_1(x) \cdot p_2(x)$$

Step 4: Construct polynomial:

$$f(x) = r(x) - d(x) := 0 \mod P(x)$$

Step 5: Apply Coppersmith's method:

- Construct a lattice using shifted multiples of $f(x)$
- Use LLL to find a short vector
- Recover the small root $r(x)$

Step 6: Recover full plaintext:

$$N(x) = g(x) + r(x)$$

3.2.4 Recovering Missing Plaintext Coefficients

Attacker's Goal: The primary objective of the attacker is to recover the missing coefficients of the plaintext polynomial $N(x)$, which have been partially hidden through encryption.

Attacker's Knowledge: The attacker has access to the encrypted polynomial $N'(x)$, given by

$$N'(x) = -a_0x^6 - a_1x^5 - a_2x^4 - a_3x^3 - a_4x^2 - a_5x + a_6,$$

where the coefficients a_0, a_1, a_2, a_3 of the original plaintext polynomial $N(x)$ are already known. In addition to this, the attacker is also provided with a set of modulus polynomials used in the encryption process:

$$p_1(x) = x^2 + x + 1, \quad p_2(x) = x^3 + x + 1, \quad p_3(x) = x^2 + 1.$$

Attack Strategy: To recover the unknown coefficients, the attacker follows a structured approach. First, polynomial division is applied to compute the modular residues of $N'(x)$ with respect to each modulus polynomial. These residuals, denoted by

$$b'_{i,j}(x) = N'_j(x) \mod p_i(x),$$

provide a simplified view of the encrypted data in a reduced polynomial space. Next, using the information from the known coefficients and the modulus polynomials, the attacker constructs a lattice representation that captures the algebraic structure of the problem. Finally, the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm is applied to this lattice in order to identify a short vector, which corresponds to a close approximation of the original plaintext polynomial. This vector reveals the values of the previously unknown coefficients.

How the Attack Works: The attack works by leveraging the mathematical structure embedded in the encrypted data. The modular reductions serve to isolate critical fragments of the ciphertext, making it easier to analyze. The lattice constructed from this data incorporates both the known plaintext coefficients and the structure of the encryption. When reduced using the LLL algorithm, the lattice yields a nearly orthogonal basis that includes a short vector closely aligned with the original polynomial $N(x)$. This vector reveals the missing coefficients, allowing the attacker to reconstruct the plaintext polynomial with high accuracy.

Complexity: The computational complexity of this attack is manageable. Polynomial division operations required in the initial phase of the attack have a complexity of $O(n^2)$, while the LLL lattice reduction algorithm, which is the

most intensive step, has a worst-case complexity of $O(n^3)$. Despite the cubic time complexity, LLL is efficient enough for polynomials of moderate degree, making this attack feasible in practice.

Overview: This attack focuses on recovering unknown coefficients of a plaintext polynomial $N(x)$ using its encrypted version $N'(x)$ and partial knowledge of its original form. By utilizing modular arithmetic and known polynomial relations, a lattice is constructed to model the algebraic constraints of the problem. The LLL algorithm is then applied to this lattice to extract a short vector, which reveals the missing coefficients. This approach highlights how partial information leakage and structured encryption schemes can be exploited through lattice-based cryptanalysis.

Algorithm: Lenstra–Lenstra–Lovász (LLL)

- Given a basis $b_0, b_1, \ldots, b_n$.
- Compute the Gram-Schmidt basis $b_0^*, b_1^*, \ldots, b_n^*$.
- Set index $k = 1$.
- For j from $k-1$ down to 0:

$$\mu_{k,j} = \frac{b_k \cdot b_j^*}{b_j^* \cdot b_j^*}$$

- If $|\mu_{k,j}| > 0.5$, update b_k:

$$b_k = b_k - \lfloor \mu_{k,j} \rceil b_j$$

- Check if the Lovász condition holds:

$$\delta \cdot \|b_{k-1}^*\|^2 \leq \|b_k^* + \mu_{k,k-1} b_{k-1}^*\|^2$$

- If true, increment k. Otherwise, swap b_k and b_{k-1}, update Gram-Schmidt coefficients, set $k = \max(k-1, 1)$ and repeat.
- The algorithm stops when $k > n$. The output is a reduced lattice basis.

3.2.5 Recovering Plaintext Polynomial Coefficients Using Gröbner Bases

Let the plaintext be a polynomial $N(x)$, and let $p_i(x)$ be a publicly known modulus polynomial chosen such that each $p_i(x)$ is pairwise coprime. Let $k_i(x)$ be a secret key polynomial satisfying:

$$\gcd(k_i(x), p_i(x)) = 1$$

This implies that an inverse $k_i^{-1}(x) \mod p_i(x)$ exists.

To mask $N(x)$, we construct a ciphertext polynomial $N_i'(x)$ such that:

$$N_i'(x) := N(x) \cdot k_i(x) \pmod{p_i(x)} \tag{3.18}$$

To lift this congruence into the ring of polynomials $\mathbb{F}[x]$, we introduce an auxiliary polynomial $c_i(x)$. The full ciphertext becomes:

$$N_i'(x) = N(x) \cdot k_i(x) + c_i(x) \cdot p_i(x) \tag{3.19}$$

This ensures that:

$$N_i'(x) \mod p_i(x) = N(x) \cdot k_i(x)$$

Since $\gcd(k_i(x), p_i(x)) = 1$, the inverse $k_i^{-1}(x) \mod p_i(x)$ exists and allows recovery of $N(x)$ as:

$$N(x) = \left(N_i'(x) \cdot k_i^{-1}(x)\right) \mod p_i(x)$$

The final form of the ciphertext polynomial is:

$$N_i'(x) = N(x) \cdot k_i(x) + c_i(x) \cdot p_i(x)$$

Goal: Recover the unknown coefficients a_6, a_5, a_4 of the plaintext polynomial $N(x)$.

Known Information: Partial plaintext coefficients a_0, a_1, a_2, a_3; Ciphertext polynomial $N'(x)$; Modulus polynomials $p_1(x), p_2(x), p_3(x)$

Ciphertext Structure: The ciphertext is of the form:

$$N'(x) = N(x) \cdot k_i(x) + c_i(x) \cdot p_i(x)$$

where $k_i(x)$ are the secret key polynomials, $p_i(x)$ are the modulus polynomials, and $c_i(x)$ is the quotient polynomial.

Recovery Strategy Using Gröbner Bases:

Step 1: Define a system of polynomial equations based on:

$$N'(x) = N(x) \cdot k_i(x) + c_i(x) \cdot p_i(x)$$

Step 2: Substitute known values into the equations to reduce unknowns.

Step 3: Use a Gröbner basis to solve for the remaining unknown coefficients.

The Gröbner basis simplifies the algebraic system using Buchberger's Algorithm. From the reduced system, one can extract the missing coefficients of $N(x)$.

Buchberger's Algorithm Overview:

Computes a Gröbner basis for a given ideal in a polynomial ring. It iteratively refines a set of polynomials until a basis is formed and simplifies solving polynomial systems.

Key Concepts:

Ideal: A set of polynomials closed under addition and multiplication by any polynomial in the ring.

Gröbner Basis: A structured basis of the ideal where leading terms follow a standard pattern.

Algorithm Steps in Buchberger's Algorithm

Input: Initialize with $S = \{f_1, f_2, \ldots, f_m\}$
Output: The stabilized set S is the Gröbner basis.
Step 1: Compute S-polynomials:

$$S(f_i, f_j) = \frac{\text{lcm}(lt(f_i), lt(f_j))}{lt(f_i)} f_i - \frac{\text{lcm}(lt(f_i), lt(f_j))}{lt(f_j)} f_j$$

where $lt(f)$ is the leading term.
Step 2: Reduce $S(f_i, f_j)$ by polynomial division using elements in S.
Step 3: If the remainder is non-zero, add it to S.

Step 4: Repeat until no new polynomials are added.

Final Output: The stabilized set S is the Gröbner basis.

Properties of Gröbner Bases:

Uniqueness: Depends on monomial order.

Solvability: Enables efficient solving of polynomial systems.

Normalization: Every ideal has a Gröbner basis.

3.2.6 Recovering plaintext polynomial using Lagrange Interpolation

Goal: Recover the original plaintext polynomial $N(x)$ from an encrypted polynomial.

Encryption Mechanism: The encryption masks the coefficients of the plaintext polynomial using a secret key polynomial. The ciphertext polynomial embeds this masking and is further altered using modular arithmetic.

Attack Strategy: The attack leverages **Lagrange Interpolation** to reconstruct the original plaintext polynomial from partial information available to the attacker.

Known to the Attacker: The ciphertext polynomial representing the encrypted message. The correction polynomials $q_i(x)$ used during encryption. The modulus polynomials $p_i(x)$ defining the modular space.

Step 1: Compute the Encrypted Residues

The attacker first reduces the ciphertext polynomial modulo each $p_i(x)$ to extract the encrypted residues:

$$b_i'(x) = \text{Ciphertext Polynomial mod } p_i(x)$$

Step 2: Undo the Masking

The true residues of the plaintext polynomial are recovered by multiplying each encrypted residue with its corresponding correction polynomial:

$$b_i(x) = b'_i(x) \cdot q_i(x) \bmod p_i(x)$$

Step 3: Construct the Lagrange Basis Polynomials

To isolate each recovered residue during interpolation, the Lagrange basis polynomials are constructed as:

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{3} \frac{x - p_j(x)}{p_i(x) - p_j(x)}$$

Each $L_i(x)$ evaluates to 1 at $p_i(x)$ and 0 at all other $p_j(x)$, effectively filtering out unwanted terms.

Step 4: Reconstruct the Plaintext Polynomial

The plaintext polynomial is reconstructed by combining the residues and their corresponding basis polynomials:

$$N(x) = \sum_{i=1}^{3} b_i(x) \cdot L_i(x)$$

Lagrange Interpolation

Given a set of n distinct points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, the goal is to construct a polynomial $P(x)$ of degree at most $n - 1$ that interpolates these points.

Step 1: Define the Lagrange Basis Polynomials

For each $i = 1, 2, \ldots, n$, compute the basis polynomial:

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Each $L_i(x)$ is 1 at x_i and 0 at all x_j where $j \neq i$, ensuring proper isolation of terms.

Step 2: Construct the Interpolating Polynomial

Use the basis polynomials to build the final interpolated polynomial:

$$P(x) = \sum_{i=1}^{n} y_i \cdot L_i(x)$$

This polynomial satisfies $P(x_i) = y_i$ for each known point, perfectly interpolating the data.
Use Cases: Useful in polynomial reconstruction, error correction, cryptanalysis, and numerical methods.

Chapter 4

Performance analysis

4.1 Results and Discussions

In this section we will give you a result of implementations of the proposed cryptosystem as well as the practical results of the proposed attacks on the target cryptosystem.

4.1.1 Implementation Environment and Tools

Language used: The implementation of the algorithm was done in Python 3.12 because Python is efficient for mathematical calculations

Libraries used: SymPy is a Python library designed for symbolic mathematics. It allows manipulation of algebraic expressions in their exact symbolic form, which is crucial in cryptographic attacks that involve polynomial identities, transformations, and simplifications
Attacks often take place over finite fields or rings defined by modular constraints. SymPy handles modular arithmetic symbolically, which allows for clean and precise expressions when computing residues or performing reductions modulo a polynomial Unlike floating-point libraries (e.g., NumPy), SymPy guarantees

exact arithmetic with no rounding errors, which is critical in cryptographic settings where precision and correctness are non-negotiable.The symbolic nature of SymPy expressions makes the output human-readable, which is extremely helpful for debugging intermediate steps in attacks and understanding the underlying mathematical transformation

Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

Installed RAM: 8.00 GB

4.1.2 Implementation of the cryptosystem

Plaintext: This is the original message encoded as a polynomial with integer coefficients. In this case, the plaintext is $15x^6 + 18x^5 + 12x^4 + 5x^3 + 18x^2 + 17x + 3$, defined over the domain $\mathbb{Z}$.

P(x): This is the modulus polynomial used in the encryption scheme. All operations are performed modulo $P(x)$.

Mi(x): These are polynomials used for applying the Chinese Remainder Theorem (CRT). They are pairwise coprime and form the modulus base.

mi(x): These are the inverses used in the CRT computations. Each $mi(x)$ is the multiplicative inverse of the product of all other $Mi(x)$ modulo $Mi(x)$.

Ciphertext: This represents the encrypted version of the plaintext polynomial after being processed through the CRT-based encryption algorithm.

q_i(x): These are the quotient polynomials computed during the CRT process. They are involved in reconstructing the original polynomial.

residues: These are the remainders obtained by reducing the plaintext modulo each of the $Mi(x)$. They represent the polynomial in its CRT-decomposed form.

b_i(x): Each $b_i(x)$ is the product of the corresponding residue and the respective inverse polynomial $mi(x)$. These are summed for reconstruction.

N_i(x): These are intermediate terms derived as the product of $mi(x)$ and $q_i(x)$, used in the CRT-based reconstruction.

Recovered Plaintext: This is the final result after applying the CRT to combine the components. Ideally, this should exactly match the original plaintext polynomial.

Plaintext:
```
Poly(15*x^6 + 18*x^5 + 12*x^4 + 5*x^3 + 18*x^2 + 17*x + 3, x, domain='ZZ')
```

Modulus Product $P(x)$:
```
Poly(500*x^7 + 10395*x^6 + 26823*x^5 + 24340*x^4 + 21279*x^3 + 2336*x^2 +
86*x + 1, x, domain='ZZ')
```

Mi(x):
```
[Poly(20*x^4 + 403*x^3 + 795*x^2 + 61*x + 1), Poly(25*x^5 + 466*x^4 +
338*x^3 + 467*x^2 + 43*x + 1), Poly(500*x^5 + 1395*x^4 + 1213*x^3 + 1111*x^2
+ 68*x + 1)]
```

Modular Inverses $m_i(x)$:
```
[Poly(-619123/113421*x^2 - 9267443/2835525*x - 15098719/2835525),
   Poly(1425339/728035*x + 60621317/14560700),
   Poly(14762983/122559492*x + 264910981/122559492)]
```

Ciphertext $N'(x)$:
```
Poly(344082385676659842450073/40000000000*x^6 + ...  +
191750616472002568507/200000000000)
```

q$_i$(x):
```
[Poly(-2317438371/11027205497*x^2 - 104170752283/827040412275*x -
169444759579/827040412275),
   Poly(22612277/408427635*x + 2884924193/24505658100),
   Poly(9697542703/1842191724252*x + 58005114375/614063908084)]
```

Residues $b_i(x)$:
```
[Poly(1966343/78125*x^2 + 9984/625*x + 230123/78125),
   Poly(-257835177/640000*x - 4391959/640000),
   Poly(-26190564*x - 1459548)]
```

b'i(x):
```
[Poly(588175320473002988/3814697265625*x^2 + ...), ...]
```

N$_i$(x):
```
[Poly(-43793/3437*x^2 - 7242097/945175*x - 11742376/945175), ...]
```

Secret keys $k_i(x)$:
```
[Poly(34*x^2 + 26*x + 27), Poly(27*x + 36), Poly(38*x + 25)]
```

Recovered Plaintext:
```
Poly(15*x^6 + 18*x^5 + 12*x^4 + 5*x^3 + 18*x^2 + 17*x + 3, x, domain='QQ')
```

Figure 4.1: Cryptosystem implementation results

4.1.3 Polynomial Modulo Attack

Plaintext: This is the original message encoded as a polynomial with integer coefficients. In this case, the plaintext is $15x^6 + 18x^5 + 12x^4 + 5x^3 + 18x^2 + 17x + 3$, defined over the domain $\mathbb{Z}$.

$P(x)$**:** This is the modulus polynomial used in the encryption scheme. All operations are performed modulo $P(x)$.

Factors: These are the factors of the **Plaintext - Residues**, obtained using the Cantor–Zassenhaus Algorithm.

$M_i(x)$**:** These are the polynomials used in applying the Chinese Remainder Theorem (CRT). They are pairwise coprime and form the modulus base.

$m_i(x)$**:** These are the inverses used in the CRT computations. Each $m_i(x)$ is the multiplicative inverse of the product of all other $M_j(x)$ (for $j \neq i$) modulo $M_i(x)$.

$b'_i(x)$**:** These are the masked residues obtained from the ciphertext polynomial modulo each modulus polynomial.

$b'_i(x) \cdot m_i(x)$**:** This term is part of the recovery process for the secret key, computed over the domain $\mathbb{Q}$.

$b_i^{-1}(x)$**:** This is the inverse of the original residue, also defined over the domain $\mathbb{Q}$.

$k(x)$**:** This is the recovered secret key polynomial.

Factor Polynomials $\pi_i(x)$:
```
[Poly(x^2 + x + 1), Poly(x^3 + x + 1), Poly(x^2 + 1)]
```

Modular Inverses $m_i(x)$:
```
[Poly(1/3*x + 2/3), Poly(2/3*x^2 - 1/3*x + 1/3), Poly(-x)]
```

Encrypted Residues $b'_i(x)$:
```
[Poly(-21*x - 39), Poly(54*x^2 + 124*x + 59), Poly(-108*x + 24)]
```

Product $b'_i(x) \cdot m_i(x)$:
```
[Poly(-20*x - 19), Poly(-20*x^2 - 79*x - 45), Poly(-24*x - 108)]
```

Modular Inverses of $b_i(x)$:
```
[Poly(7/127*x - 6/127), Poly(-2220/1877*x^2 + 1497/1877*x - 3232/1877), Poly(-5/204*x
- 1/68)]
```

Recovered Secret Keys $K_i(x)$:
```
[Poly(x^2 + 2*x + 3), Poly(x^3 + x^2 + 1), Poly(x^2 + 3*x + 2)]
```

Figure 4.2: Polynomial Modulo Attack

4.1.4 Polynomial Factorization and Parameter Recovery in GF(p)

Factors: These are the factors of the $P(x)$**:**, obtained using the Berlekamp Algorithm.

$M_i(x)$**:** These are the polynomials used in applying the Chinese Remainder Theorem (CRT). They are pairwise coprime and form the modulus base.

$m_i(x)$**:** These are the inverses used in the CRT computations. Each $m_i(x)$ is the multiplicative inverse of the product of all other $M_j(x)$ (for $j \neq i$) modulo $M_i(x)$.

$b'_i(x)$**:** These are the masked residues obtained from the ciphertext polynomial modulo each modulus polynomial.

$b'_i(x) \cdot m_i(x)$**:** This term is part of the recovery process for the secret key, computed over the domain $\mathbb{Q}$.

$b_i^{-1}(x)$**:** This is the inverse of the original residue, also defined over the domain $\mathbb{Q}$.

$k(x)$**:** This is the recovered secret key polynomial.

Factors:
```
[x^2 + x + 1, x^3 + x + 1, x^2 + 1]
```

Modular Inverses $m_i(x)$:
```
[Poly(1/3*x + 2/3), Poly(2/3*x^2 - 1/3*x + 1/3), Poly(-x)]
```

Residues $b_i(x)$:
```
[-7*x - 13, 3*x^2 + 48*x + 31, 30*x - 18]
```

Encrypted Residues $b'_i(x)$:
```
[-21*x - 39, 54*x^2 + 124*x + 59, 24 - 108*x]
```

Product $b'_i(x) \cdot m_i(x)$:
```
[Poly(-20*x - 19), Poly(-20*x^2 - 79*x - 45), Poly(-24*x - 108)]
```

Modular Inverses of $b_i(x)$**:**
```
[Poly(7/127*x - 6/127), Poly(-2220/1877*x^2 + 1497/1877*x - 3232/1877), Poly(-5/204*x -
1/68)]
```

Recovered Secret Keys $k_i(x)$:
```
[Poly(x + 2), Poly(x^2 - x), Poly(3*x + 1)]
```

Reconstructed Key Polynomials $K_i(x)$:
```
[Poly(x^2 + 2*x + 3), Poly(x^3 + x^2 + 1), Poly(x^2 + 3*x + 2)]
```

Figure 4.3: Polynomial factorization and parameter recovery

4.1.5 Partial Plaintext Recovery via Coppersmith's Method

We are given a polynomial $N(x) = 15x^6+18x^5+12x^4+5x^3+18x^2+17x+3$. It consists of a known part $g(x) = 15x^6 + 18x^5 + 12x^4$ and an unknown secret part $r(x) = 5x^3 + 18x^2 + 17x + 3$ that we aim to recover.

We are also given two moduli: $p_1(x) = x^4 + 1$ and $p_2(x) = x^4 + 2$.

We compute the residues of $N(x)$ modulo each polynomial: $b_1(x) = N(x) \mod p_1(x)$, $b_2(x) = N(x) \mod p_2(x)$.

Next, we subtract the known part $g(x)$ from these residues to isolate the congruence of the unknown part $r(x)$: $r(x) := b_1(x) - g(x) \mod p_1(x)$, and $r(x) := b_2(x) - g(x) \mod p_2(x)$.

To solve this system of congruences, we apply the **Coppersmith method**, which leverages the **LLL (Lenstra–Lenstra–Lovász)** lattice basis reduction algorithm. we construct a lattice from the given modular equations. The goal is to find a **small root** $r(x)$ of the polynomial modulo the product of the moduli. By embedding the modular constraints into a lattice and applying LLL reduction, we recover a short vector that corresponds to the secret part $r(x)$ of the original polynomial $N(x)$.

We then compute the solution: $r(x) := b_1(x) \cdot v(x) \cdot p_2(x) + b_2(x) \cdot u(x) \cdot p_1(x) \mod \text{lcm}(p_1(x), p_2(x))$.

Finally, we reduce the result modulo the least common multiple to obtain the minimal polynomial representation.

The result is: **recovered** $r(x) = 5x^3 + 18x^2 + 17x + 3$.

Therefore, we conclude: **Successfully recovered** $r(x)$ **using polynomial CRT.**

Known part of plaintext:
`Poly(15*x^6 + 18*x^5 + 12*x^4, x, domain='ZZ')`

Residual $r(x) := b_1(x) - g(x) \mod p_1(x)$:
`Poly(5*x^3 + 18*x^2 + 17*x + 3, x, domain='ZZ')`

Residual $r(x) := b_2(x) - g(x) \mod p_2(x)$:
`Poly(5*x^3 + 18*x^2 + 17*x + 3, x, domain='ZZ')`

Recovered unknown plaintext $r(x)$:
`Poly(5*x^3 + 18*x^2 + 17*x + 3, x, domain='ZZ')`

Figure 4.4: Plaintext Recovery using Coppersmith's method

4.1.6 Recovering Missing Plaintext Coefficients

We demonstrate a **lattice-based method** to recover the partial plaintext polynomial from a given ciphertext. This approach is inspired by **Coppersmith's method**, which allows recovering small roots of polynomial equations modulo a known modulus.

Given: The ciphertext polynomial is

$$N'(x) = -64x^6 - 250x^5 - 360x^4 - 545x^3 - 492x^2 - 403x - 172$$

Modulus polynomials:

$$p_1(x) = x^2 + x + 1, \quad p_2(x) = x^3 + x + 1, \quad p_3(x) = x^2 + 1$$

Step 1: Compute modular residues. We reduce $N'(x)$ modulo each polynomial to obtain:

$$b_1(x) = N'(x) \bmod p_1(x), \quad b_2(x) = N'(x) \bmod p_2(x), \quad b_3(x) = N'(x) \bmod p_3(x)$$

Step 2: Convert to coefficient vectors. Each modulus polynomial is expressed as a vector of its coefficients. To ensure consistency in dimensions, we pad these vectors to equal length.

Step 3: Construct the lattice. A lattice matrix is built using the padded coefficient vectors as basis vectors. The modular constraints are embedded into the structure of the lattice.

Step 4: Apply LLL reduction. Using the **LLL algorithm**, we reduce the lattice basis to obtain short vectors. The shortest vector is expected to correspond to the coefficient vector of the unknown plaintext polynomial, assuming it has sufficiently small coefficients.

Step 5: Recover the plaintext. The first row of the reduced basis is interpreted as the recovered plaintext polynomial. This vector satisfies all given modular constraints and is likely the original message.

FyPLL Dependency: FyPLL is a Python library for working with polynomial lattices and solving lattice-based cryptographic problems.

GMP Dependency: High-precision arithmetic is handled by the `libgmp` library, which is loaded using:

```
import ctypes
try:
    gmp = ctypes.CDLL("libgmp.dll")
    print("GMP loaded successfully")
except OSError as e:
    print("Failed to load GMP:", e)
```

Original Polynomial:
Poly(-64x^6-250x^5-360x^4-545x^3-492x^2-403x-172, domain='ZZ')
Modulus Polynomials:
p1: $x^2 + x + 1$
p2: $x^3 + x + 1$
p3: $x^2 + 1$
Encrypted Residues:
b1'(x): $-21x - 39$
b2'(x): $54x^2 + 124x + 59$
b3'(x): $24 - 108x$
Decryption Factors:
q1'(x): $0.\overline{3}$
q2'(x): $0.\overline{1}x^2 - 0.\overline{2}x + 0.\overline{5}$
q3'(x): $-0.1x - 0.3$
Unmasked Residues:
b1(x): $-7.0x - 13.0$
b2(x): $3.0x^2 + 48.0x + 31.0$
b3(x): $30.0x - 18.0$
Recovered Plaintext:
N(x) = 15.0x^6 + 18.0x^5 + 12.0x^4 + 5.0x^3 + 18.0x^2 + 17.0x + 3.0

Figure 4.5: Recovering Missing Plaintext Coefficients

4.1.7 Recovering Plaintext Polynomial Coefficients Using Gröbner Bases

In this section, we demonstrate how to symbolically recover the unknown coefficients of a plaintext polynomial using the **Gröbner basis technique**. This approach is especially useful when partial information about the plaintext is known, and modular relations are provided via the encryption process.

Step 1: Define unknown coefficients. We assume the plaintext polyno-

mial $N(x)$ is of degree 6, and we denote the unknown coefficients as:

$$N(x) = a_6x^6 + a_5x^5 + a_4x^4 + 5x^3 + 18x^2 + 17x + 3$$

Here, a_6, a_5, a_4 are the **unknown coefficients**, while a_3, a_2, a_1, a_0 are **known**.

Step 2: Define known and unknown key polynomials. The encryption key consists of polynomials $k_1(x), k_2(x), k_3(x)$. We assume:

$$k_1(x) = x^2 + 2x + 3, \quad k_2(x) = x^3 + x^2 + 1$$

Note: $k_3(x)$ is unknown and considered in full generality.

Step 3: Known modulus polynomials:

$$p_1(x) = x^2 + x + 1, \quad p_2(x) = x^3 + x + 1, \quad p_3(x) = x^2 + 1$$

Step 4: Ciphertext equation system. The ciphertext polynomial is:

$$N'(x) = -64x^6 - 250x^5 - 360x^4 - 545x^3 - 492x^2 - 403x - 172$$

Based on the encryption model, we have:

$$N'(x) = N(x)k_i(x) + c_i(x)p_i(x), \quad \text{for } i = 1, 2, 3$$

where $c_1(x), c_2(x), c_3(x)$ are unknown multiplier polynomials.

Step 5: Gröbner basis construction. We express the three equations symbolically and substitute the known plaintext coefficients. This results in a system of polynomial equations in the unknowns $a_4, a_5, a_6, c_1, c_2, c_3$. Using **Buchberger's algorithm**, we compute a **Gröbner basis** for the ideal generated by the system:

$$\text{basis} = \texttt{groebner}(f_1 - f_1^{\text{rhs}}, f_2 - f_2^{\text{rhs}}, f_3 - f_3^{\text{rhs}})$$

Step 6: Solving the system. We solve the reduced Gröbner basis for the unknowns. The output is:

$$a_6 = 15, \quad a_5 = 18, \quad a_4 = 12$$

Using symbolic computation and Gröbner basis techniques, we have successfully recovered the unknown portion of the plaintext polynomial:

$$N(x) = 15x^6 + 18x^5 + 12x^4 + 5x^3 + 18x^2 + 17x + 3$$

This confirms that the polynomial model of the ciphertext allows for partial recovery of plaintext when algebraic structure and known values are leveraged.

Partially Known Plaintext Polynomial:
a0 = 3, a1 = 17, a2 = 18, a3 = 5

Partially Known Secret Key:
k1 = x² + 2x + 3
k2 = x³ + x² + 1
p1 = x² + x + 1
p2 = x³ + x + 1
p3 = x² + 1

Groebner Basis System:
192*a4*c1*x + 912*a4*c2*x + ... (truncated for brevity)
⋮
3*a4*x + 3*a5*x + 3*a6*x + ...

Recovered Coefficients:
a4 = 15, a5 = 18, a6 = 12

Figure 4.6: Polynomial cryptanalysis via Gröbner basis

4.1.8 Recovering plaintext polynomial using Lagrange Interpolation

Given the ciphertext and the modulo polynomials and the quotient polynomials Here $P(x)$ is computed using the modulo polynomials

Here true residues are unmasked

Now, computing $M(x)$ and $m(x)$, Lagrange Interpolation is used to derive the initial plaintext

4.2 Modification of the cryptosystem

In this section, we present the proposed modifications to the existing cryptosystem aimed at addressing the discrepancies identified during the cryptanalysis.

4.2.1 Secure Polynomial Encryption via PRF-Based Random Masking

Encoding with Random Polynomial Masking: To enhance the security of polynomial-based cryptosystems, we propose a masking mechanism using a random polynomial $R(x)$. This method involves constructing a masked version

Ciphertext Polynomial:
```
Poly(-64*x**6 - 250*x**5 - 360*x**4 - 545*x**3 - 492*x**2 - 403*x - 172, x, domain='ZZ')
```

Modulus Polynomials:
```
p1 = x² + x + 1
p2 = x³ + x + 1
p3 = x² + 1
```

Encrypted Residues:
```
b1'(x) = -21x - 39
b2'(x) = 54x² + 124x + 59
b3'(x) = 24 - 108x
```

Decryption Factors:
```
q1'(x) = 0.333...
q2'(x) = 0.111...x² - 0.222...x + 0.555...
q3'(x) = -0.1x - 0.3
```

True Residues (Unmasked):
```
b1(x) = -7.0x - 13.0
b2(x) = 3.0x² + 48.0x + 31.0
b3(x) = 30.0x - 18.0
```

Recovered Plaintext N(x):
```
15.0x + 18.0x + 12.0x + 5.0x³ + 18.0x² + 17.0x + 3.0
```

Figure 4.7: Recovery using Lagrange Interpolation

of the plaintext polynomial $N(x)$ by adding a randomized, noise-like component that is a multiple of the public modulus polynomial $P(x)$. Specifically, the encoded message $N_{\text{enc}}(x)$ is computed as:

$$N_{\text{enc}}(x) = N(x) + R(x) \cdot P(x) \tag{4.1}$$

Here, $R(x)$ is a random polynomial generated in such a way that it sufficiently masks the original plaintext $N(x)$. The polynomial $P(x)$ represents the product of all the modulus polynomials used in the encryption scheme.

The polynomial $R(x)$ is not arbitrarily chosen. It must satisfy certain conditions to provide effective masking. Its degree must be at least the difference between the degrees of $N(x)$ and $P(x)$, i.e., $\deg(R) \geq \deg(N) - \deg(P)$. This ensures that the multiplication $R(x) \cdot P(x)$ yields a polynomial of sufficiently high degree and entropy to make the encoded message indistinguishable from random noise.

To securely generate $R(x)$, both communicating parties pre-share a secret key K. A pseudo-random function (PRF) is used to deterministically derive $R(x)$ from K and a nonce (a random or sequential value used once per encryption). The derivation is expressed as:

$$R(x) = \text{PRF}(K \parallel \text{nonce}) \tag{4.2}$$

The nonce guarantees that the random polynomial changes with every encryption, providing semantic security by preventing ciphertext reuse.

Decoding and Recovery of the Plaintext: On the receiver's end, decryption is straightforward provided the secret key K and nonce are known. The receiver reconstructs $R(x)$ in the same way and simply subtracts $R(x) \cdot P(x)$ from the received $N_{\text{enc}}(x)$ to recover the original message:

$$N(x) = N_{\text{enc}}(x) - R(x) \cdot P(x) \tag{4.3}$$

This process is efficient and does not require solving any complex equations, thanks to the deterministic generation of $R(x)$.

Security Properties and Attack Resistance

The security of this approach lies in the properties of $R(x)$. Since it is generated using a PRF, it appears pseudorandom to any adversary lacking the key. The product $R(x) \cdot P(x)$ introduces high-degree, high-entropy noise into the ciphertext, making it computationally infeasible to distinguish from random values. Even if $R(x)$ itself has small coefficients, its multiplication with $P(x)$ significantly amplifies the masking effect.

This masking mechanism also provides resistance against attacks like Coppersmith's method, which typically exploit small root properties. Because the masked polynomial appears as a large, noisy object, small root attacks become impractical, and no structural weakness is exposed without knowledge of $R(x)$.

4.2.2 Encryption and Decryption Using Polynomial CRT

This encryption scheme utilizes the Chinese Remainder Theorem (CRT) for polynomials to securely split, encrypt, and reconstruct messages. The process begins with the plaintext, for instance, a numerical representation such as "PSMFSRND = 1518120518171303", which is divided into smaller blocks like 1518, 1205, 1817, and 1303. Each block is then converted into a polynomial $N_i(x)$, where the degree of $N_i(x)$ is less than the degree of a corresponding modulus polynomial $p_i(x)$.

Encryption: Each $N_i(x)$ is reduced modulo its corresponding $p_i(x)$, resulting in residues $b_i(x)$ such that

$$b_i(x) := N_i(x) \mod p_i(x).$$

The final ciphertext polynomial $N'(x)$ is constructed using the formula:

$$N'(x) = \sum_{i=1}^{s} b_i(x) \cdot M_i(x) \cdot k_i(x) \mod P(x)$$

where $M_i(x) = \frac{P(x)}{p_i(x)}$, and $k_i(x)$ is a modular inverse used for CRT reconstruction.

For decryption, the receiver computes

$$b'_i(x) := N'(x) \mod p_i(x)$$

for each modulus polynomial. To remove the masking, each $b'_i(x)$ is multiplied by a precomputed inverse $q_i(x)$, which is derived from $k_i(x)$, resulting in

$$b_i(x) := b'_i(x) \cdot q_i(x) \mod p_i(x).$$

Finally, the original message is reconstructed by concatenating the coefficients of all recovered $b_i(x)$.

4.3 Feasibility study

This feasibility study analyzes six cryptanalytic techniques—Cantor-Zassenhaus, Berlekamp, Coppersmith, LLL-based recovery, Gröbner basis, and Lagrange interpolation—across various Galois fields. Execution times show that factorization methods perform best on small fields,Cantor-Zassenhaus is seen to perform better than Berlekamp while algebraic attacks like Coppersmith

and Gröbner basis improve significantly when more plaintext coefficients are known. These trends highlight the practical viability of algebraic recovery when partial information is available.

Table 4.1: Performance comparison of crypanalytic cases (time in microseconds)

Method	Galois Field Size (p)				
	5	47	101	1009	2003
Berlekamp (P)	117,185.35	119,898.09	144,148.12	264,570.00	458,820.00
Cantor-Zassenhaus	80,101.00	83,231.00	91,910.00	103,020.00	116,700.00
Coppersmith	132,021.00	161,047.00	186,090.00	420,327.43	741,361.00
LLL (Lattice)	145,301.00	170,6003.32	200,000.00	468,200.50	805,006.70
Gröbner Basis	162,037.23	181,010.00	210,067.00	490,209.00	830,050.62
Lagrange Interpolation	75,210.00	78,109.00	85,257.00	96,031.00	110,240.11

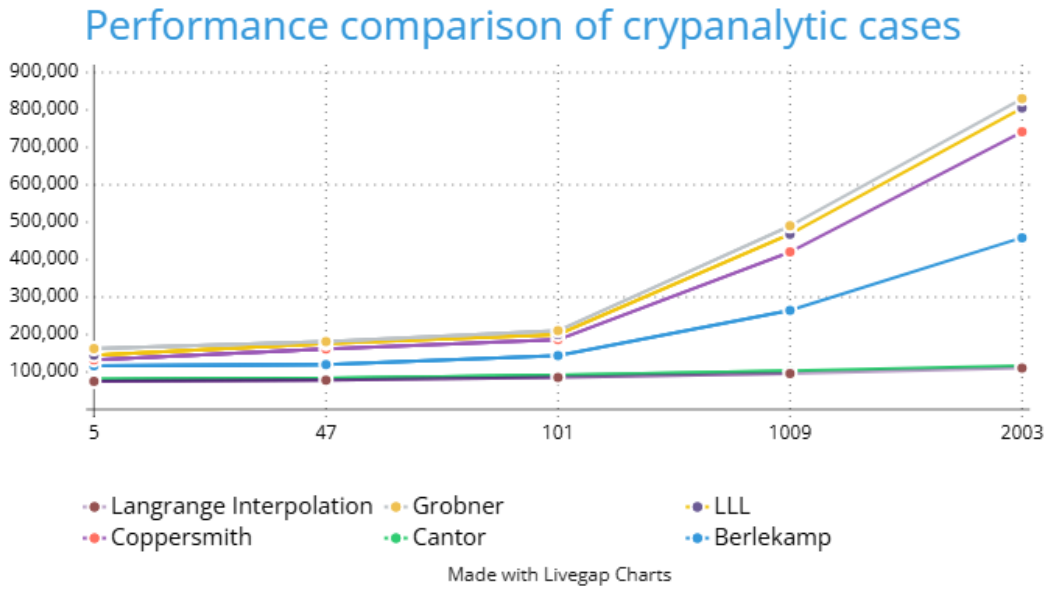

Figure 4.8: Comparision of cryptanalysis cases

The table 4.1 presents execution times (in microseconds) for six cryptanalytic methods across various Galois fields $\mathbb{F}_p$ with increasing prime sizes: $p = 5, 47, 101, 1009, 2003$. In all scenarios, the plaintext polynomial was fixed at degree 6, and encryption was performed using three known modulus polynomials, each of degree at most 3.

Cantor–Zassenhaus consistently outperforms most other methods, especially in lower fields, due to its efficient randomized factorization mechanism. **Lagrange Interpolation** also demonstrates strong performance across all fields, as it only depends on solving for coefficients given a set of values, making it computationally light.

Berlekamp's method shows a gradual increase in runtime, but remains

competitive due to its deterministic nature, particularly effective in small to mid-sized fields. Algebraic recovery methods—**Coppersmith**, **LLL**, and **Gröbner Basis**—while slower overall, are influenced by the number of known plaintext coefficients. In this benchmark, they were evaluated assuming only 2 coefficients of the plaintext were known. Their performance would improve significantly if 3 or more coefficients were known, reducing the solution space and enabling faster recovery.

Thus, while interpolation and factorization methods dominate in speed, algebraic techniques offer deeper recovery potential, especially when more side information (like known coefficients) is available.

Since algebraic techniques rely heavily on the available coefficient information, we present a comparison of Grobner basis, Coppersmith's method, and LLL lattice reduction, evaluating their performance across different Galois fields and varying levels of known coefficients.

Table 4.2: Execution times (in μs) vs. number of known coefficients for $\mathbb{F}_{101}$ (1/3)

Method	2 Coeffs.	3 Coeffs.	4 Coeffs.
Coppersmith	151,101	92,500	62,037
LLL	171,038	101,200	66,503
Gröbner Basis	188,529	119,328	73,206

Table 4.3: Execution times (in μs) vs. number of known coefficients for $\mathbb{F}_{1009}$ (2/3)

Method	2 Coeffs.	3 Coeffs.	4 Coeffs.
Coppersmith	265,034	162,109	103,249
LLL	285,063	174,012	108,023
Gröbner Basis	309,045	188,539	120,010

Table 4.4: Execution times (in μs) vs. number of known coefficients for $\mathbb{F}_{2003}$ (3/3)

Method	2 Coeffs.	3 Coeffs.	4 Coeffs.
Coppersmith	150,035	92,519	60,205
LLL	171,038	101,204	66,510
Gröbner Basis	188,526	119,338	73,280

Chapter 5

Conclusion and Future Work

5.1 Conclusions

In this report, we analyzed the structure and security of a polynomial-based cryptosystem, with a strong focus on its vulnerabilities through factorization, algebraic and lattice-based cryptanalysis techniques. We explored modifications to enhance the robustness of the cryptosystem, including adjustments in key structure, encoding mechanisms, and the use of Polynomial Residue Number System (PRNS) for improved efficiency and security. The cryptanalysis revealed specific discrepancies that were effectively addressed through the proposed enhancements.

5.2 Future Work

As a continuation of this research, future work can focus on developing an academic article that establishes the proposed cryptosystem modifications, presents cryptanalysis results, and includes performance analyses. Such an article would contribute valuable theoretical insights, practical applications, and potential real-world uses of the new scheme. Additionally, formal publication would facilitate peer review and open avenues for collaboration

Bibliography

[1] I. Yakymenko, M. Karpinski, R. Shevchuk, and M. Kasianchuk, "Symmetric encryption algorithms in a polynomial residue number system," *Journal of Applied Mathematics*, vol. 2024, pp. 1–12, 05 2024. (document), 1.1, 1.2, 1.3, 1, 3.1, 3.1.1, 3.1.1, 3.1.2, 3.1.2, 3.1.2, 3.1.2, 3.1

[2] M. M. Kasianchuk, I. Z. Yakymenko, and Y. M. Nykolaychuk, "Symmetric cryptoalgorithms in the residue number system," *Cybernetics and Systems Analysis*, vol. 57, pp. 329–336, Mar 2021. 2, **??**

[3] Q. Luo, "Research and application of chinese remainder theorem," *Theoretical and Natural Science*, vol. 9, pp. 45–53, 11 2023. 2, **??**

[4] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *Algorithmic Number Theory* (J. P. Buhler, ed.), (Berlin, Heidelberg), pp. 267–288, Springer Berlin Heidelberg, 1998. 2, **??**

[5] N. K. Tiwari and B. Chaurasia, "Various approaches towards cryptanalysis," *International Journal of Computer Applications*, vol. 127, no. 14, pp. 1–5, 2015. 2, **??**

[6] A. Jakhar and S. Kotyada, "On the irreducible factors of a polynomial ii," *Journal of Algebra*, vol. 556, pp. 1–20, 2020. 2, **??**

[7] E. R. Berlekamp, "Factoring polynomials over finite fields," *The Bell System Technical Journal*, vol. 46, no. 8, pp. 1853–1859, 1967. 2, **??**, 3.2.2, 3.2.2

[8] D. G. Cantor and H. Zassenhaus, "A new algorithm for factoring polynomials over finite fields," *Mathematics of Computation*, vol. 36, no. 154, pp. 587–592, 1981. 2, **??**, 3.2.1, 3.2.1

[9] S. D. Miller, B. Narayanan, and R. Venkatesan, "Coppersmith's lattices and "focus groups": An attack on small-exponent rsa," *Journal of Number Theory*, vol. 222, pp. 376–392, 2021. 2, **??**, 3.2.3

[10] M. Herrmann and A. May, "Solving linear equations modulo divisors: On factoring given any bits," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 406–424, Springer, 2009. 2, **??**

[11] B. Buchberger, "An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal," *Journal of Symbolic Computation*, vol. 41, no. 3-4, pp. 475–511, 2006. Originally PhD Thesis, University of Innsbruck, 1965. Translated by M.P. Abramson. 2, **??**

[12] T. Jakobsen and L. R. Knudsen, "The interpolation attack on block ciphers," in *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 28–40, Springer, 1997. 2, **??**

[13] M. K. Hossain, G. Sorwar, A. Ahmad, and W. Saman, "Lattice-based cryptanalysis of rsa-type cryptosystems: A bibliometric analysis," *Cybersecurity*, vol. 7, no. 1, pp. 1–26, 2024. 2, **??**