UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO VICTOR PACHECO SOBRAL

ON THE SECURITY OF MULTIVARIATE ENCRYPTION SCHEMES

RIO DE JANEIRO
2022

JOÃO VICTOR PACHECO SOBRAL

ON THE SECURITY OF MULTIVARIATE ENCRYPTION SCHEMES

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Severino Collier Coutinho

RIO DE JANEIRO
2022

CIP - Catalogação na Publicação

```
        Sobral, João Victor Pacheco
S677o      On the security of multivariate encryption
        schemes / João Victor Pacheco Sobral. -- Rio de
        Janeiro, 2022.
           42 f.

           Orientador: Severino Collier Coutinho.
           Trabalho de conclusão de curso (graduação) -
        Universidade Federal do Rio de Janeiro, Instituto
        de Computação, Bacharel em Ciência da Computação,
        2022.

           1. Criptografia Pós-Quântica (PQC). 2.
        Criptografia de Chave Pública Multivariável (MPKC).
        3. Criptografia de Chave Pública. I. Coutinho,
        Severino Collier, orient. II. Título.
```

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

JOÃO VICTOR PACHECO SOBRAL

ON THE SECURITY OF MULTIVARIATE ENCRYPTION SCHEMES

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 29 de agosto de 2022

BANCA EXAMINADORA:

S. C. Coutinho

Severino Collier Coutinho
Ph.D. (Leeds)

[illegible]

João Antônio Recio da Paixão
D.Sc. (PUC-RJ)

[illegible]

Hugo de Holanda Cunha Nobrega
Ph.D. (Universiteit van Amsterdam)

“There is no branch of mathematics, however abstract, which may not someday be applied to phenomena of the real world."

Nikolai Lobachevsky

RESUMO

A criptografia moderna está em perigo por causa dos computadores quânticos, mesmo que, limitados por hardware, já existem algoritmos que podem quebrar os esquemas de chave pública mais utilizados para tráfego de informação. Criptografia multivariável é um bom candidato para criação de esquemas criptográficos seguros até para computadores quânticos, pois são baseadas em um problema NP-Difícil. Nesse trabalho, nós explicamos ataques comuns a criptossistemas multivariáveis além de estudarmos a teoria e implementação deles.

Palavras-chave: Criptografia Pós-Quântica (PQC); Criptografia de Chave Pública Multivariável (MPKC); Criptografia de Chave Pública

ABSTRACT

Modern cryptography is endangered by quantum computers because, even though we do not have good hardware, we already have algorithms to break most public key schemes that are widely used. Multivariate cryptography is a good candidate to generate quantum safe cryptography schemes since it is based on a NP-Hard problem even for quantum computers. We explain common attacks to multivariate cryptosystems and go through the theory and implementation of them.

Keywords: Post Quantum Cryptography (PQC); Public Key Cryptography; Multivariate Public Key Cryptography (MPKC)

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

γ	Greek letter Gama
Δ	Greek letter Delta
ϕ	Greek letter Phi
$\in$	Belong
$\notin$	Does not belong
$\circ$	Map composition
$\oplus$	Direct sum
$\setminus$	Set subtraction
Ω	Greek letter Omega
$\prec$	Well order precedes
$\cup$	Union
$\cap$	Intersection

CONTENTS

1 INTRODUCTION

Quantum computers are endangering most public key cryptography schemes, there are already quantum algorithms capable of factoring a big number or solving the discrete logarithm problem (SHOR, 1999), but due to hardware limitation it is still not feasible. On the other hand, symmetric encryption schemes are known to be quantum resistant already, Grover's algorithm (GROVER, 1996) only reduces the key search by half of the bits, this means that AES-256 is still practical even for quantum computers.

In the last three decades, multivariate public key cryptography (MPKC) have been researched because it is considered one main candidate for post quantum cryptography. Since the first scheme proposed at (MATSUMOTO; IMAI, 1988), most schemes follow a common construction, the public key P is the composition of three maps $P = T \circ F \circ S$ where S, T are linear maps, F is a $\mathbb{F}^n \to \mathbb{F}^m$ map and $\mathbb{F}$ is a finite field. This map F is called central map and it has n variables and m equations. The central map must have the property that the system $F(\boldsymbol{x}) = \boldsymbol{y}$ is easily solvable for the ciphertext $\boldsymbol{y}$.

The main problem discussed in this work is how to build a secure multivariate encryption scheme, ever since the first was launched we currently have no scheme that is both secure and efficient (IKEMATSU; NAKAMURA, 2022). Even though we don't have encryption schemes with those properties, we currently have digital signature schemes that are based on the same problem.

This work approaches multiple attacks on multivariate schemes that explore the structure of the central map. We define rank attacks that are based on the rank of the map when it is translated to matrices and we also see an example and implementation of an algebraic attack. Not only we describe the attacks but we also show possible variations and combinations of attacks that could lead to a smaller time complexity.

We aim to describe attacks and variations in order to achieve a better time complexity for direct and algebraic attacks on multivariate encryption schemes. A big portion of multivariate attacks uses the Groebner basis algorithm in order to solve a system of polynomial equations and we will explain all the theory and applications of these methods.

This work has the following structure, Chapter 2 touches the preliminaries, all the basic theory that we need to know before studying Groebner basis and multivariate schemes. Chapter 3 will go through the Groebner basis theory, applications and implementation of algorithms. Chapter 4 talks about the general construction of multivariate schemes and Chapter 5 describes the attacks on known encryption schemes.

2 PRELIMINARIES

These are the fundamentals to understand how some cryptography schemes work.

2.1 RINGS AND FIELDS

A ring is a set that contains two operations, addition (+) and multiplication ($*$). A ring, with an associated set $\mathcal{R}$, must satisfy those conditions and axioms (LANG, 2002):

- Multiplication and addition are associative. Meaning that for all elements a, b and c in $\mathcal{R}$ it satisfy $a * (b * c) = (a * b) * c$ and $a + (b + c) = (a + b) + c$.
- Addition is commutative. That is, for all a and b in $\mathcal{R}$, $a + b = b + a$ holds.
- There are multiplicative and additive identity element. Meaning that if 1 is the multiplicative identity, 0 the additive identity and a an element in $\mathcal{R}$ then $1 * a = a * 1 = a$ and $0 + a = a + 0 = a$.
- For each a in $\mathcal{R}$ exists an additive inverse. Meaning that, if $-a$ is the inverse of a, 0 is the additive identity then $a - a = 0$.
- Multiplication distributes over addition. For all a, b, c in R it is true that $a*(b+c) = (a * b) + (a * c)$ and $(b + c) * a = (b * a) + (c * a)$.

As an example, the set of natural numbers $\mathbb{N}$, with the operations addition and multiplication, does not form a ring since there are no additive inverses, so there is no x such that $x + 3 = 0$. All rings covered in this study will be using 0 as the additive identity element and 1 as the multiplicative identity.

A field is a special kind of commutative ring. In other words it has all the axioms and conditions of a ring plus the multiplication is also commutative, and all nonzero elements have an inverse. That is for every x in a field, there is a y such that $x * y = 1$.

As an example, the set of integers $\mathbb{Z}$, with the default operations, is a ring but not a field, since the multiplicative inverses are not in the base set. There can be infinite and finite fields, this study will be focused on finite fields.

A polynomial ring, $\mathcal{R}[X]$, is an algebraic structure that allows the representation of polynomials in a base ring, this means that an element p of this ring is of the form:

$$p = c_0 + c_1X + c_2X^2 + ... + c_nX^n$$

where $c_0, c_1, c_2, ..., c_n$ are the coefficients of p, all of them are elements of $\mathcal{R}$. $X, X^2, ..., X^n$ are the powers of X, and X is called variable.

2.2 IDEALS AND QUOTIENT RING

Let $\mathbb{K}$ be any field, an ideal is a subset $I \in \mathbb{K}[x_1, ..., x_n]$ satisfying these conditions:

- $0 \in I$
- If $p, g \in I$, then $p + g \in I$
- If $p \in I$ and $h \in \mathbb{K}[x_1, ..., x_n]$, then $hp \in I$

Definition. Let $p_1, ..., p_n \in \mathbb{K}[x_1, ..., x_n]$. Then

$$\langle p_1, ..., p_n \rangle = \left\{ \sum_{i=1}^{n} h_i p_i | h_1, ..., h_n \in \mathbb{K}[x_1, ..., x_n] \right\}$$

Lemma. We call $I = \langle p_1, ..., p_n \rangle$ the ideal generated by $p_1, ..., p_n$.

Proof. Since $0 = \sum_{i=1}^{n} 0 * p_i$, $0 \in I$. Suppose that $p = \sum_{i=1}^{n} f_i p_i$ and $q = \sum_{i=1}^{n} g_i p_i$, let h be a polynomial in $\mathbb{K}[x_1, ..., x_n]$. Then

$$p + q = \sum_{i=1}^{n} (f_i + g_i) p_i$$

$$hp = \sum_{i=1}^{n} (hf_i) p_i$$

completes the proof. (COX; LITTLE; O'SHEA, 2007, p. 30).

Definition. An ideal I is finitely generated if and only if there is a generator of I which is finite.

Definition. Let f, g be two polynomials of $\mathbb{K}[x_1, ..., x_n]$ and I an ideal of this same ring. We say f and g are congruent module I if $f - g \in I$. We denote it as $f \equiv g \mod I$.

Proposition. Let I be an ideal of $\mathbb{K}[x_1, ..., x_n]$. The congruence modulo I is an equivalence relation on $\mathbb{K}[x_1, ..., x_n]$.

Proof.

- **Reflexive:** $f - f = 0 \in I$ for every $f \in \mathbb{K}[x_1, ..., x_n]$, by the definition of ideal.
- **Transivity:** Considering $f \equiv g \equiv h \mod I$. Since I is closed under addition, we have $f - h = f - g + g - h \in I$, therefore $f \equiv h \mod I$.
- **Symmetry:** Suppose that $f \equiv g \mod I$. Then $f - g \in I$, implying $g - f = (-1)(f - g) \in I$. Therefore $g \equiv f \mod I$.

Definition. The quotient of $\mathbb{K}[x_1, ..., x_n]$ modulo I, written as $\mathbb{K}[x_1, ..., x_n]/I$, is the set of equivalence classes:

$$\mathbb{K}[x_1, ..., x_n]/I = \{[f] | f \in \mathbb{K}[x_1, ..., x_n]\}.$$

This quotient is a ring (COX; LITTLE; O'SHEA, 2007, Proposition 5,p.241).

2.3 FINITE FIELDS AND EXTENSIONS

A finite field is a field with a finite number of elements. Generally denoted as $\mathbb{F}_n$ with n being the order, that is, the number of elements in the field. We will be considering only prime finite fields, so in this case n will always be a prime number.

Let's consider any field $\mathbb{F}$, we say $\mathbb{K}$ is a field extension of $\mathbb{F}$ if $\mathbb{K} \supseteq \mathbb{F}$. $\mathbb{K}$ is by definition an $\mathbb{F}$-vector space with scalar multiplication inherited by the field multiplication, the dimension of this vector space is called the degree of the extension. Supposing n is the degree of $\mathbb{K}$, then we can build an isomorphism:

$$\phi : \mathbb{K} \to \mathbb{F}^n$$

It is easy to see that if $\mathbb{F}$ has size p then the size of $\mathbb{K}$ is p^n. The smallest element c in $\mathbb{F}$ such that $c * 1 = 0$ is known as characteristic of $\mathbb{F}$ and, for a prime field the characteristic is a prime number q. This means that the field $\mathbb{F}_q$ is contained in $\mathbb{F}$, so $\mathbb{F}$ is an extension of $\mathbb{F}_q$. Thus, the number of elements in every finite field is a prime power.

A polynomial $p(x) \in \mathbb{F}[x]$ is irreducible if it cannot be factored into the product of two nonconstant polynomials. The quotient ring $\mathbb{F}/\langle p(x)\rangle$ will be an extension field $\mathbb{K}$ of $\mathbb{F}$ of degree n, in fact all extension fields will have this form. Since $p(x) = x^n + a_{n-1}x^{n-1} + ... + a_1x + a_0 = 0 \in \mathbb{K}$, we can rewrite elements of this field as polynomials in terms of x having degree at most $n-1$ because $x^n = -a_{n-1}x^{n-1} - ... - a_1x - a_0$ then the following isomorphism is valid:

$$\phi : a_0 + a_1x^1 + \cdots + a_{n-1}x^{n-1} \in \mathbb{K} \longmapsto (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}^n$$

2.4 FROBENIUS POWERS

Following the same definitions as the last section, let's recall that every finite group with g elements satisfies $x^g = d$ with d being the identity element. We know that $\mathbb{F}$ is a field then all nonzero elements have a multiplicative inverse, thus $\mathbb{F}^*$ is a multiplicative group and the identity element is 1. For all $k \in \mathbb{F}^*$, we can conclude that $k^{q^n} = k$ and $k^{q^n-1} = 1$ with q being the characteristic of $\mathbb{F}$ knowing that q^n is the number of elements in $\mathbb{F}$.

Definition 2.3.1. The Frobenius Transformation is known as the function $\mathbb{F} \to \mathbb{F}$ defined by $k \mapsto k^q$. This function is an $\mathbb{F}_q$-linear transformation, this means that $(ak + z)^q = ak^q + z^q$ for all $a \in \mathbb{F}_q$ and $x, z \in \mathbb{F}$.

Since that Frobenius Transformation $X \mapsto X^q$ for $X \in \mathbb{K}$ is a $\mathbb{F}$-linear transformation, all polynomials that can be written as below are too.

$$\mathcal{F}(X) = \sum_{i=0}^{n-1} a_i X^{q^i} \tag{2.1}$$

In fact, this mean that the composition $\phi \circ \mathcal{F} \circ \phi^{-1} : \mathbb{F}^n \to \mathbb{F}^n$ is $\mathbb{F}$-linear as well. So we have n degree 1 homogeneous polynomials. Let $a = b_0 + b_1 x + ... + b_{n-1}x^{n-1} \in \mathbb{K}$ if we raise it to the power q^i, we get $a^{q^i} = b_0 + b_1 x^{q^i} + ... + b_{n-1}(x^{n-1})^{q^i}$, then if we put everything in matrices we have, for all i from 0 to $n-1$:

$$\Delta = \begin{bmatrix} x^{(0)} & x^{(1)} & \cdots & x^{(n-2)} & x^{(n-1)} \\ (x^{(0)})^{q^1} & (x^{(1)})^{q^1} & \cdots & (x^{(n-2)})^{q^1} & (x^{(n-1)})^{q^1} \\ (x^{(0)})^{q^2} & (x^{(1)})^{q^2} & \cdots & (x^{(n-2)})^{q^2} & (x^{(n-1)})^{q^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (x^{(0)})^{q^{n-1}} & (x^{(1)})^{q^{n-1}} & \cdots & (x^{(n-2)})^{q^{n-1}} & (x^{(n-1)})^{q^{n-1}} \end{bmatrix}$$

$$\begin{bmatrix} a \\ a^q \\ a^{q^2} \\ \vdots \\ a^{q^{n-1}} \end{bmatrix} = \Delta \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}$$

Recalling that $\phi(a) = (b_0, ..., b_{n-1})^T$ then $\alpha = \Delta \cdot \phi(a)$. Since Δ is invertible (LIDL; NIEDERREITER, 1997), we can do $\phi(a) = \Delta^{-1}\alpha$. Calling M a matrix representation of a linear transformation F then $F \circ \phi(a) = M \cdot \Delta^{-1} \cdot \alpha$, this mean that we can rewrite $\phi(a)^{-1} \circ F \circ \phi(a)$ as a dot product of $(x^0, x^1, ..., x^{n-1})^T$ and $M \cdot \Delta^{-1} \cdot \alpha$. This dot product can be rewritten as the same equation (2.1).

3 GROEBNER BASIS

To better understand the attacks on multivariate cryptography, first it is essential to know what a Groebner basis is. It is a finite generating set of an ideal that gives us information about it with some operations. The most important information that we can get from a Groebner basis is a set of solutions to the polynomial equations formed by an ideal, this information is pretty much how we break multivariate schemes, depending on the scheme design. In this chapter we prove the existence, show the usefulness and go through an algorithm to compute the basis for polynomial rings over a field.

3.1 DEFINITIONS

Let's define $\mathcal{R}$ as a multivariate polynomial ring $\mathbb{F}[x_1, ..., x_n]$ where $\mathbb{F}$ is a field. A monomial is any x^a, with $a \in \mathbb{N}^n$, that can be rewritten as $x_1^{a_1} \cdot x_2^{a_2} \cdots x_{n-1}^{a_{n-1}} \cdot x_n^{a_n}$ and $a_1, ..., a_n \in \mathbb{N}$. Calling $\mathcal{M}$ the set of all monomials in $\mathcal{R}$ then we can also write any polynomial in $\mathcal{R}$ as:

$$p = \sum_{x^a \in \mathcal{M}} b_a x^a$$

Note that each polynomial written in this form is unique and almost all b_a in $\mathbb{F}$ are zeros.

To compute the Groebner basis it is necessary to define the order of monomials, a monomial ordering is any relation $<$ on $\mathcal{M}$ that satisfies these three conditions:

- $<$ is a well-order, this means that every non-empty subset of $\mathcal{M}$ has a least element in this ordering.
- Multiplication respects the ordering, if $x^a, x^b, x^c \in \mathcal{M}$ then $x^a < x^b \implies x^a \cdot x^c < x^b \cdot x^c$
- Any two distinct elements of $\mathcal{M}$ are comparable, this means that $<$ is a total order relation.

The Groebner basis depends on the ordering chosen, as does the complexity of the algorithm used to compute it. In this work, we will be using only two orderings, which we will now define.

Definition: Given $a \in \mathbb{N}^n$, we define $|a| = \sum_{i=1}^{n} a_i$.

- Lexicographic order(lex): $a <_{lex} b$ if and only if the leftmost entry that is not zero of $b - a$ is positive. This is the alphabetical order, just like the dictionary.

- Graded reverse lexicographic order(grevlex): $a <_{grevlex} b$ if and only if $|a| < |b|$ or $|a| = |b|$ and $a <_{rlex} b$. This order is useful because it bounds the total degree when running the division algorithm.

The ordering is necessary to know how to determine the remainder in the polynomial division algorithm. This algorithm is as simple as the euclidean division algorithm, actually it is a generalization of it. The following definitions complete the basics to understand the algorithm, let $p = \sum_{x^a \in \mathcal{M}} b_a x^a$ be a nonzero polynomial in $\mathcal{R}$:

- The exponent of p is $exp(p) = max_{<}(a : b_a \neq 0)$. We fix $exp(0) = -\infty$.
- The support of p is $supp(p) = \{x^a : b_a \neq 0\}$, that is the variables with a coefficient different from 0.
- The degree of p is $deg(p) = \sum_{i=1}^{n} b_i$ where $(b_1, ..., b_n) = exp(p)$.
- The leading monomial of p is $LM(p) = x^{exp(f)}$.
- The leading coefficient of p is $LC(p) = b_a$ with $a = exp(p)$.
- The leading term of p is $LT(p) = LC(p)LM(p)$.

Example: A polynomial with 3 variables and $\mathbb{F} = \mathbb{R}$, $<$ being the lex order. Let's call p the polynomial:

$$p = 3 \cdot x_1^2 x_2^3 + 6 \cdot x_1 x_3^2 + x_2 x_3^4$$
$$supp(p) = \{x_1^2 x_2^3, x_1 x_3^2, x_2 x_3^4\} \quad LM(p) = x_1^2 x_2^3 \quad exp(p) = (2,3,0)$$
$$LC(p) = 3 \quad LT(p) = 3 \cdot x_1^2 x_2^3$$

3.1.1 Euclidean division for multivariate polynomials

Let's describe a division algorithm for multivariate polynomials over a field.

Theorem 3.1. Let $P = (p_1, ..., p_m)$ be an ordered $m-$tuple of nonzero polynomials in $\mathcal{R}$, and for each i let $a_i = \exp(p_i)$. Then, for every $p \in R$, there exist polynomials $q_1, ..., q_m, r \in R$ such that

$$p = q_1 p_1 + ... + q_m p_m + r$$

where

$$q_i \in \bigoplus_{a \in \Omega_i - a_i} \mathbb{F}x^a \text{ with } \Omega_i = (a_i + \mathbb{N}^n) \setminus \bigcup_{j=1}^{i-1} (a_j + \mathbb{N}^n)$$

$$r \in \bigoplus_{a \in \Omega} \mathbb{F}x^a \text{ with } \Omega = \mathbb{N}^n \setminus \bigcup_{i=1}^{m} (a_i + \mathbb{N}^n)$$

In addition, $\exp(p) \geq (\exp(q_i p_i), \exp(h))$.

Proof. Supposing p has a term that is divisible by some $LM(p_i)$, let's take the greatest of all these terms. With $cx^a LT(p_i)$, $c \neq 0 \in \mathbb{F}$ and $a \in \mathbb{N}$, assuming that i is the smallest index satisfying these conditions, then $a + a_i \in \Omega_i$ and $a \in \Omega_i - a_i$. If we apply the same to $p' = p - cx^a p_i$, we will get a polynomial whose leading monomial precedes and is smaller than $cx^a LT(p_i)$ and since this term is a maximum, then a term $c' x^{a'} LT(p'_i)$ from p' is strictly smaller than $cx^a LT(p_i)$.

Since we fixed a well order, this process will reach to a 0 polynomial and will stop. In the end we will group all terms to obtain $(q_1, ..., q_m)$ and $r = p - q_1 p_1 - ... - q_m p_m$. This also proves that we have an algorithm for the division and the algorithm finishes with the correct answer.

Algorithm 1: Euclidean division for polynomials

```
**Data:** $p \in \mathcal{R}$, $(p_1, ..., p_m) \in (\mathcal{R} \setminus \{0\})^m$
**Result:** $(q_1, ..., q_m) \in \mathcal{R}^m$ and $r \in \mathcal{R}$
 1: $q_1 \leftarrow 0, ..., q_m \leftarrow 0$
 2: $r \leftarrow 0$
 3: $f \leftarrow p$
 4: while $f \neq 0$ do
 5:    $i \leftarrow 1$
 6:    divided $\leftarrow$ false
 7:    while $i \leq m$ and divided = false do
 8:       if $LT(p_i)$ divides $LT(f)$ then
 9:          $q_i \leftarrow q_i + LT(f)/LT(p_i)$
10:          $f \leftarrow f - (LT(f)/LT(p_i))p_i$
11:          divided $\leftarrow$ true
12:       else
13:          $i \leftarrow i + 1$
14:       if divided = false then
15:          $r \leftarrow r + LT(f)$
16:          $f \leftarrow f - LT(f)$
```

The result of this algorithm depends on the order of p, this means that the monomial order can produce different results. In conclusion the division algorithm does not tell us that a polynomial lies in the ideal generated by the divisors, the Groebner basis have the set of divisors that can tell us this.

3.2 THEORY

In this section we go through the theory behind the Groebner basis and prove its existence. To prove it, we use Dickson's Lemma.

3.2.1 Dickson's Lemma

A monomial ideal is an ideal of $\mathcal{R}$ that is generated by monomials. In other words I is a monomial ideal if there is $A \subseteq \mathcal{M}$ such that $I = \langle A \rangle$. By convention, the ideal generated by the empty set is $\mathcal{R}$, therefore is a monomial ideal.

With this definition in mind, now we describe an important property of monomial ideals, which is the base of the proof of Dickson's lemma.

Proposition 3.2.0. Let I be a monomial ideal of $\mathcal{R}$. Then a polynomial p lies in I if and only if $supp(p) \subseteq I$.

Proof. In fact the monomials of a monomial ideal are the $\mathcal{R}$-multiples of its generators. In general this does not work for any ideal, just monomial ideals. For example there can be $p \in I$ with terms outside I, that $\mathbb{F} = \mathbb{R}, n = 2$ and $I = \langle x_1 + x_2 \rangle$, then $x_1 + x_2 \in I$ but $x_1 \notin I$. This is important because it implies that a monomial ideal is characterized by the monomials in it.

Our last definition for the lemma is a partial order $\prec$ over $\mathbb{N}^n$ as $A \prec B \iff \forall i : a_i \leq b_i$ where $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n) \in \mathbb{N}^n$. This definition is consistent with the division definition we made before, so $A \prec B$ implies that x^A divides x^B.

Proposition 3.2.1. Given a subset $\mathcal{M} \neq \emptyset$ of $\mathbb{N}^n$ there exists $A \in \mathcal{M}$ such that $A \prec B$ for all $B \in \mathcal{M}$.

Proof. Any nonempty subset $\mathcal{M} \subset \mathbb{N}^n$ that satisfies $\mathcal{M} + \mathbb{N}^n = \mathcal{M}$ is determined by its $\prec$-minimal elements:

$$\mathcal{M} = \bigcup_{A \prec -\min} (A + \mathbb{N}^n)$$

Lemma 3.2.2. Combinatorial Dickson's lemma. Let $\mathcal{M}$ be a nonempty subset of $\mathbb{N}^n$ then $\mathcal{M}$ has a finite number of $\prec$-minimal elements.

Proof. This is a proof by induction on n. For $n = 1$, the result follows the Well-ordering Principle, now let's assume that the result is still true for $n-1$, we need to show that it holds for n. Let $A = (a_1, ..., a_n) \in \mathcal{M}$ be a $\prec$-minimal element of $\mathcal{M}$. Then any other $\prec$-minimal element must lie in:

$$\mathcal{M} \setminus (A + \mathbb{N}^n) = \bigcup_{i=1}^{n} \mathcal{A}_a^i$$

$$\mathcal{A}_a^i = \bigcup_{a=0}^{a_i} \{(c_1, \ldots, c_n) \in \mathcal{M} : c_i = a\}$$

Now let's get another $\prec$-minimal element $B = (b_1, ..., b_n) \in \mathcal{M}$, let i be such $b_i < a_i$ then B is a $\prec$-minimal element of $\mathcal{A}_{b_i}^i$. Let π_i be the projection from $\mathbb{N}^n$ onto $\mathbb{N}^{n-1}$ that drops the entry with index i, we need to prove that $\pi_i(B)$ is a $\prec$-minimal element of $\pi_i(\mathcal{A}_{b_i}^i)$.

If there is a $C = (c_1, ..., c_n) \in \mathcal{A}_{b_i}^i$, with $\pi_i(C) \prec \pi_i(B)$ but $\pi_i(C) \neq \pi_i(B)$, given $c_i < b_i$, we conclude that $C \prec B$ but $C \neq B$ and this is a contradiction because B is a $\prec$-minimal element of $\mathcal{A}^i_{b_i}$. By the induction hypothesis, there is a finite number of $\prec$-minimal elements of $\mathcal{A}^i_{b_i}$ for each i, therefore there is a finite number of $\prec$-minimal elements of $\mathcal{M}$. This concludes the proof.

Lemma 3.2.3. Dickson's lemma. Let I be a monomial ideal, $I = \langle \mathcal{A} \rangle$ with $\mathcal{A} \subseteq \mathcal{M}$, then there is a finite subset $\mathcal{B} \subseteq \mathcal{A}$ such that $I = \langle \mathcal{B} \rangle$.

Proof. Let's assume $\mathcal{A} \neq \emptyset$ otherwise it would be trivial. Let $\mathcal{M} = exp\{A \in \mathbb{N}^n : x^A \in \mathcal{A}\} \neq \emptyset$. By lemma 3.2.2, $\mathcal{M}$ has a finite number of $\prec$-minimal elements, let's call them $m_1, ..., m_k$. We want to prove that $I = \langle \mathcal{B} \rangle$ where $\mathcal{B} = \{x^{m_1}, ..., x^{m_k}\} \subseteq \mathcal{A}$. By the proposition 3.2.0 we've shown that p lies in I if and only if $supp(p) \subseteq I$, every $x^A \in \mathcal{A}$ lie in $\langle \mathcal{B} \rangle$. Since $A \in \mathcal{M}$ then $m_i \prec A$ for some i and x^{m_i} divides x^A, in conclusion $x^A \in \mathcal{B}$ finishing the proof.

3.2.2 Hilbert's Basis Theorem

This theorem is a classical result in commutative algebra, it states that all ideals of $\mathcal{R}$ are finitely generated. We will state the theorem and prove it using Dickson's Lemma, which is a particular case of Hilbert's basis theorem.

Theorem 3.2. Let I be an ideal of $\mathcal{R}$, then I is finitely generated.

Proof. For the case where $I = \{0\}$ it is trivial. So, first let's assume $I \neq \{0\}$, defining $LM(I)$ as the set of all leading monomials of the polynomials that are in I, then $\langle LM(I) \rangle$ is a monomial ideal. By Dickson's lemma, there exists $g_1, ..., g_m \in I$ that $\langle LM(I) \rangle = \langle LM(g_1), ..., LM(g_m) \rangle$. If we claim $I = \langle g_1, ..., g_m \rangle$ the theorem would be proved.

We know that $I \supseteq \langle g_1, \dots, g_m \rangle$. Let $p \in I$ and let's apply the division algorithm to divide p by $(g_1, ..., g_m)$, obtaining $b_1, ..., b_m, r \in \mathcal{R}$ such that:

$$p = b_1 g_1 + ... + b_m g_m + r$$

$$r = p - b_1 g_1 - ... - b_m g_m \in I$$

If $r \neq 0$ then $LM(r) \in \langle LM(g_1), ..., LM(g_m) \rangle$ and $LM(r)$ will be divisible by some $LM(g_i)$, this is contradictory to Theorem 3.1, concluding that $r = 0$. Consequently:

$$p = b_1 g_1 + ... + b_m g_m + 0 \in \langle g_1, ..., g_m \rangle$$

finishing the proof.

3.2.3 Existence of Groebner basis

In this subsection we will go through the existence and some properties of the Groebner basis, some types of Groebner basis are unique and we will describe them as well. Firstly we need some definitions and propositions.

Definition. $G = \{g_1, ..., g_k\}$ is a Groebner basis of an ideal I of $\mathcal{R}$ if

$$\langle LM(g_1), ..., LM(g_k)\rangle = \langle LM(I)\rangle$$

where $LM(I)$ is a set containing all leading monomials of the polynomials in I. Now we can say that these bases generate the ideal I.

Proposition 3.3. Consider I a nonzero ideal of $\mathcal{R}$, then I has a Groebner basis and this basis generates I.

Proof. Let's look at the proof of Hilbert's Basis theorem, $\langle LM(I)\rangle$ is finitely generated by some $LM(g_i)$, and $g_i \in I$, therefore the Groebner basis exists. If we use the division algorithm this condition implies that $I = \langle g_1, ..., g_k\rangle$, thus the Groebner basis generates I.

Proposition 3.4. Let $\{g_1, ..., g_k\} \subset I$, where I of $\mathcal{R}$. Then G is a Groebner basis if and only if for all $p \in I$, the reminder of the division of p by $(g_1, ..., g_k)$ is zero.

Proof. Let's prove first $\implies$. Assume $p = g + r$ where $p \in I$, $g \in I$ $r \neq 0 \in I$ and no monomial r is divisible by any $LM(g_i)$. Since $r = f - g \in I$, $LM(r) \in \langle LM(I)\rangle = \langle LM(g_1, ..., LM(g_k)\rangle$ therefore $LM(r)$ is divisible by some of $LM(g_i)$ giving a contradiction. Since the division algorithm has this same form, the remainder of the division of p by $(g_1, ..., g_m)$ is 0.

Now we need to prove $\Longleftarrow$. Let $p \in I$, now apply the division algorithm, dividing p by G. Let's assume that the reminder is zero, then $q_1, ..., q_k \in \mathcal{R}$ satisfy the solution of the algorithm. This implies that $exp(p) = \max\{exp(q_1p_1), ..., exp(q_kp_k)\}$ and $LM(p) = LM(q_i)LM(p_i)$ for some i, so $LM(p) \in \langle LM(g_1), ..., LM(g_k)\rangle$. This concludes the proof.

Corollary 3.5. No matter how they are computed, the remainders of the division algorithm with Groebner bases as divisors are unique.

Proof. Assume that G is a Groebner basis and let $p \in \mathcal{R}$. Let's suppose that $r, r' \in \mathcal{R}$ are the remainders of the division of p by G, then $p = g + r, p = g' + r'$ where $g, g' \in I$ and no monomial of the remainders is divisible by any element of $LM(G)$.

This implies that $(r - r') + (g' - g) = 0$ and $r - r'$ satisfies the same property. We know that 0 and $g - g'$ are in I the previous proposition implies $r - r' = 0$ thus $r = r'$, finishing the proof.

This corollary implies that the remainder of every polynomial is well-defined modulo I, so the quotient ring $\mathcal{R}/I$ is unique. Even though this is unique, the Groebner basis itself is not, for example, if G is a Groebner basis of an ideal I, it is clear that $G \cup \{p\}$ is a Groebner basis of I for all $p \in I$.

Given an ideal I of $\mathcal{R}$ and G the Groebner basis of this ideal, we say that G is a minimal Groebner basis if every $g \in G$ is monic and for all $k \in G, LM(k) \notin \langle LM(G - \{k\})\rangle$. Let $G = \{g_1, ..., g_l\}$ be a minimal Groebner basis of I, and $a_i = exp(g_i)$ for any i. $\{a_1, ..., a_l\}$ is the set of $\prec$-minimal elements of $exp(I)$. Actually any two minimal Groebner basis have the same cardinality. Even though the exponents are unique, the basis itself is not.

Given an ideal I of $\mathcal{R}$ and G a Groebner basis of I, we say that G is a reduced Groebner basis if every $g \in G$ is monic and for all $k \in G$, no monomial of k lies in $\langle LM(G - \{k\}) \rangle$

Proposition 3.6. Assume $I \neq 0$ is a polynomial ideal, then I has a reduced Groebner basis that is unique.

Proof. Let $G = \{g_1, ..., g_k\}$ be a minimal Groebner basis of I, $a_i = exp(g_i)$ for each i. We can assume that $a_1 \prec ... \prec a_r$. Now let's replace each g_i by its remainder when it is divided by $(g_1, ..., g_{i-1})$ then, by induction, $g_i \in x^{a_i} + \bigoplus_{a \in \mathbb{N}^n \setminus exp(I); a \prec a_i} \mathbb{F}x^a$. This new G is a reduced Groebner basis. Suppose that $\{p_1, ..., p_k\}$ is another reduced Groebner basis of I, then for all i we have $p_i - g_i \in I \cap \left(\bigoplus_{a \in \mathbb{N}^n \setminus exp(I)} \mathbb{F}x^a \right) = \{0\}$, therefore $p_i = g_i$, showing both the uniqueness and the existence. Algorithm 2 shows how this computation can be done efficiently.

3.3 APPLICATIONS

The Groebner basis have plenty of applications on computational algebra and algebraic geometry. We will discuss two applications one of which will be the base of most attacks to multivariate cryptosystems.

3.3.1 Ideal Equality Problem

This problem is to check if two polynomial ideals I and H of $\mathcal{R}$ are equal. One simple way to solve this is to compute the Groebner basis for each ideal and see if each polynomial of each basis lies in the other ideal.

Algorithm 2: Reduced Groebner Basis

```
Require: $A$ as a Groebner basis of $\langle A \rangle$
Ensure: $A$ as a reduced Groebner basis
1: $F \leftarrow A$, $f \leftarrow p$
2: for $g \in F$ do
3:     $g' \leftarrow$ remainder on division of $g$ by $A$
4:     $A \leftarrow A \setminus \{g\}$
5:     $A \leftarrow A \cup \{g'\}$
```

3.3.2 Solving Systems of Polynomial Equations

This is the most important problem (for cryptography) that we can solve with Groebner basis. There may be some cases where the polynomial equations do not have a solution or have many. Just as a naming convention, overdetermined polynomial systems have no solution, undetermined polynomial systems have infinite solutions, those that have the same number of variables and equations have a finite number of solutions.

When we have a linear system $Ax = b$, to solve it we need to apply rules that do not change the solution and change the system to an easier to solve. This means that the solutions are the same for both bases of the vector space spanned by the expressions, so changing the basis to a more structured one will help us find a solution. However, in polynomial equations the vector space spanned by the polynomials is not the only thing to consider, the ideal will take care of this part. So we are building a correspondence of Gaussian elimination to Groebner basis algorithms and structured bases will be the basis itself. In other words, the solutions to $(p_1 = 0, ..., p_n = 0)$ are the same to $(f_1 = 0, ..., f_k = 0)$ when $\langle p_1, ..., p_n\rangle = \langle f_1, ..., f_k\rangle$, so it is sufficient to compute the solutions using any basis of $\langle p_1, ..., p_n\rangle$.

A polynomial system $(p_1 = 0, ..., p_n = 0)$ is zero-dimensional if it has a finite number of solutions. Computing solutions of a zero-dimensional system that is also a reduced Groebner basis is efficient, and this is because the Groebner basis has a triangular form. Therefore, we can compute the reduced Groebner basis first and then solve the system generated by the basis polynomials.

Proposition 3.7. Let $G = \{g_1, ..., g_k\}$ be the reduced lex Groebner basis of $\langle G\rangle$. If the system $(g_1 = 0, ..., g_k = 0)$ is zero-dimensional, then the following algorithm computes all solutions of it.

```
Algorithm 3: Solutions of polynomial system given by $(g_1 = 0, \ldots, g_k = 0)$
  Require: $G = \{g_1, \ldots, g_k\}$ a reduced Groebner basis of $\langle G\rangle$
  Ensure: $S_1$, the set that describes the solutions to the polynomial system of $G = 0$
   1: Let $g$ be the only polynomial in $G \cap \mathbb{F}[x_n]$
   2: $S_n \leftarrow \{a \in \mathbb{F} : g(a) = 0\}$
   3: for $j \leftarrow n-1, n-2, \ldots, 1$ do
   4:     for $(a_{j+1}, \ldots, a_n) \in S_{j+1}$ do
   5:         $p \leftarrow \gcd(\{h(x_j, a_{j+1}, \ldots, a_n) : h \in (G \cap \mathcal{R}_{j-1}) \setminus \mathcal{R}_j\})$
   6:         $S_j \leftarrow S_j \cup \{(a, a_{j+1}, \ldots, a_n) : p(a) = 0\}$
```

Note that if $1 \in G$ then it is a trivial case and the solution is $\emptyset$.

3.4 BUCHBERGER'S ALGORITHM

In this last section we will describe an algorithm to produce Groebner basis. If we consider the proposition 3.4, G will be a Groebner basis of I if the remainder when dividing every polynomial in I by G is zero. I can have an infinite number of polynomials, so this can be impossible to compute if we run a naive algorithm.

Proposition 3.8. For all i from 2 to k and for each $\prec$-minimal element γ of

$$B = \left[\bigcup_{j=1}^{i-1} (a_j - a_i + \mathbb{N}^n)\right] \bigcap \mathbb{N}^n$$

Let $G = \{g_1, ..., g_k\} \subset \mathcal{R}$ be such that the remainder of the division of $x^\gamma g_i$ by G is zero. Then G is a Groebner basis of $I = \langle g_1, ..., g_k \rangle$.

Proof. Assume that there is a nonzero polynomial $p \in I$ with a nonzero remainder when divided by G. This polynomial can be written as a linear combination of polynomials of the form $x^b g_i$ because it is in I. We know that the quotient and the remainder are a linear combination of the dividend, so there must exist $x^b g_i$ with a nonzero remainder.

Let's remember that $a_j = deg(g_j)$ for each j. If $b \in \Delta_i - a_i$, then the decomposition $x^b g_i = 0g_1 + ... + x^b g_i + ... + 0g_k + 0$ satisfies the condition of this theorem and it would be the output of the division algorithm, hence the polynomial would have no remainder. Since this is absurd, it follows that $b \in B$. From proposition 3.2.1, there is a $\prec$-minimal element $\gamma \in B$ such that $\gamma \prec b$, so we can write

$$x^\gamma g_i = \sum_{i=1}^{k} q_i g_i$$

where q_i is the quotient, implying that $\gamma + a_i = exp(x^\gamma g_i) = max\{exp(q_j) + a_j\} \implies exp(q_j) + a_j \leq \gamma + a_i$ when $j \geq i$. Then, $exp(q_j) + a_j \in (a_i + \mathbb{N}^n)$ and this contradicts the main theorem of the division algorithm.

$x^b g_i$ has a nonzero remainder and we know that:

$$x^b g_i = \sum_{j=1}^{k} x^{b-\gamma} q_j g_j$$

there must exist $i_1 \in \{1, ..., k\}$ such that $x^{b-\gamma} q_{i_1} g_{i_1}$ has a nonzero remainder, in the expansion of $x^{b-\gamma} q_{i_1}$ there is a monomial x^{b_1} with $b_1 \leq b - \gamma + exp(q_{i_1})$ such that $x^{b_1} g_{i_1}$ has nonzero remainder. It follows that, whenever $i_1 \geq i$, the following is a strict inequality:

$$b_1 + a_{i_1} < b - \gamma + exp(p_{i_1}) + a_{i_1} \leq b + a_i$$

If we keep iterating like this, we obtain a sequence $(b_j, a_j) \in \mathbb{N}^n \times \{1, ..., k\}$ with $b_{j+1} + a_{j+1} \prec b_j + a_{i_j}$ as a strict inequality if $i_j \leq i_{j+1}$. Since $i_j \in \{1, ..., k\}$, there is a constant subsequence of $\{i_j\}$, and we can assume that this subsequence is the sequence itself. The a's will cancel out in the inequality, hence $b_{j+1} < b_j$ for all $j \in \mathbb{N}$ and this concludes the proof because this is absurd, since $<$ is a well order (defined in 3.1). This result is important because it tells us that we can always find the Groebner basis with a finite number of polynomials.

A $\prec$-minimal element of B is a $\prec$-minimal element of the set $\{a_{ji} : j = 1, ..., i-1\}$ where $a_{ji} = a_j - a_i$, we have:

$$x^{a_{ij}} = \frac{lcm(x^{a_i}, x^{a_j})}{x^{a_i}} = \frac{x^{a_j}}{gcd(x^{a_i}, x^{a_j})}$$

Given two polynomials $p, g \in \mathcal{R}$, the S-polynomial of p and g is:

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

where $\gamma = max\{exp(p), exp(g)\}$, such that $x^\gamma = lcm(LM(p), LM(g))$.

In the first step of computing the remainder of $x^{a_{ji}} g_i$, the polynomials $S(g_i, g_j)$ appears, therefore it is sufficient for their remainders to be zero in order to obtain a Groebner basis. Since these S-polynomials are inside I, this is a necessary condition.

Proposition 3.9. G is a Groebner basis of I if and only if for each $1 \leq i, j \leq k$, the remainder on division of $S(g_i, g_j)$ by G is zero.

This proposition defines an algorithm to test if a finite generating set is a Groebner basis. We can use these criteria to produce basis from finite generating sets by using algorithm 4. The algorithm begins with a finite basis for an ideal and then adds every S-polynomial that did not go to zero on division by the basis. If the algorithm finishes then all S-polynomials leaves zero remainders when divided by the basis, therefore this proposition assures that the basis is a Groebner basis.

Algorithm 4: Buchberger's algorithm

```
Require: $P := (p_1, \ldots, p_k) \in \mathcal{R}^k$
Ensure: A Groebner basis $G$ for $\langle p_1, \ldots, p_k \rangle$ with $\{p_1, \ldots, p_k\} \subseteq G$
1: $G \leftarrow P$
2: repeat
3:    $G \leftarrow G'$
4:    for each pair $l, m$, $l \neq m \in G'$ do
5:       $S \leftarrow$remainder on division of $S(l, m)$ by $G'$
6:       if $S \neq 0$ then
7:          $G \leftarrow G \cup \{S\}$
8: until $G = G'$
```

Proposition 3.10. The Buchberger's Algorithm finishes in a finite number of steps. The proof to this proposition can be found at (COX; LITTLE; O'SHEA, 2007, p 91-92).

4 MULTIVARIATE PUBLIC KEY CRYPTOGRAPHY

In this chapter we describe some ideas behind Multivariate Public Key Encryption, we define the general construction idea and give an example of an old construction that is already obsolete.

4.1 CRYPTOGRAPHY

Before we go through the multivariate part, it is necessary to understand what type of cryptography we will be doing. Multivariate schemes are good candidates to be quantum resistant public key cryptography (PKC) algorithms.

4.1.1 Public Key Cryptography

This type of cryptography is defined by a one way trapdoor function γ, in other words it is a few-to-one function that is easy to compute for whoever is sending the message and hard to invert by only knowing the function. Another requirement is that whoever is receiving the message will have a secret key that turns the problem of inverting the function easy.

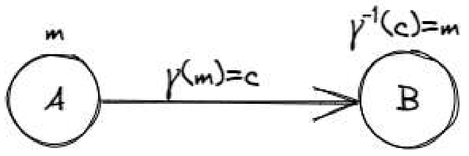

Figure 1 – Protocol that let A send messages securely to B

Suppose that we have three people, Ana, Bernardo and Carlos. Ana wants to send a message m to Bernardo and to solve her problem she needs to evaluate the function $c = \gamma(m)$, then she can send c to Bernardo. Since we defined γ as a trapdoor, nobody else should be able to read it. When Bernardo receives c he can use his secret key to read m.

In the cryptographic terminology, m is called the **plaintext**, c is the **ciphertext**, the secret that Bernardo needs to decrypt the message is called the **Secret Key** and the function γ is the **Public Key**. Also, when we evaluate the function γ with any plaintext we call it encryption and when we invert the function we call decryption.

Example. Let's define the famous RSA as an example. The public key is a modulus that is the product of two prime numbers p and q, and an exponent e. Note that only the modulus is public, the primes p and q are not. The private key is $d = e^{-1} \mod \phi(n)$.

The γ function will be the modular exponentiation $m^e = c \mod n$ and its inverse is the function $c^d = m \mod n$.

Note that any entity without d cannot obtain, by any means, the message m, unless the factorization of n is discovered. Factoring a number is a hard problem, thus γ is a trapdoor function based on the problem of factoring numbers, if we choose a small modulus it will be easy to factor, so we must choose one that is huge, usually 2048 bits long. To better understand how and why RSA really works, please refer to (COUTINHO; APLICADA, 2005).

4.1.2 Post-Quantum Cryptography

Big companies and governments are investing a lot on researching quantum computers, and some researchers estimate that in a couple of decades we can build good quantum computers. Those computers could break most public key cryptography we use today. Peter Shor developed an algorithm (SHOR, 1999) capable of finding a cycle in a group in polynomial time, this means that most PKC like RSA, Diffie-Hellman, Elliptic Curve Cryptography and other systems that use a trapdoor function based in factoring or discrete log are in danger.

Currently, RSA and ECC are widely used in communication, according to Google (GOOGLE, 2022) 98% of the websites browsed in Chrome are using HTTPS. The TLS protocol, which HTTPS and most encrypted transport protocols are based, uses PKC for the key exchange at the beginning of connection.

Since most of those schemes are in danger, it is needed to develop new schemes to guarantee privacy. NIST proposed a 3 round standardization with a group of researchers, but currently there are no multivariate encryption algorithms being considered. Most encryption schemes considered are based on lattices and Goppa codes.

4.2 MULTIVARIATE PUBLIC KEY CRYPTOSYSTEMS

This work is focused on encryption schemes, but this section will describe the general construction of any cryptosystem based on the problem of solving polynomial equations over a finite field.

In Chapter 2 we defined $\mathbb{F}$ as a finite field with q (prime) elements and $\mathbb{K}$ is the extension field of $\mathbb{F}$ of degree n. $\mathcal{R}_{\leq d}$ is the set of polynomials in $\mathbb{F}[x_1, ..., x_n]$ of degree at most d, if we fix $d = 2$ then all polynomials will be quadratic. $F : \mathbb{F}^n \to \mathbb{F}^m$ is called a **regular function** if it is given by m multivariate polynomials (all functions $\mathbb{F}^n \to \mathbb{F}^m$ are regular once we impose $x^q = x$) . Let's now define the main problem on which will be based our trapdoor function.

The MQ Problem. Let $p_1, ..., p_n \in \mathcal{R}$ be quadratic multivariate polynomials chosen uniformly at random. Find $(a_1, ..., a_n) \in \mathbb{F}^n$ that is a root for all p_i from 1 to n. There may be no $(a_1, ..., a_n)$ values that satisfy this condition.

To know if a system of multivariate polynomial equations has a solution is NP-Hard even for quadratic polynomials over $\mathbb{F}_2$ (GAREY; JOHNSON, 1990). Since we still don't have any quantum algorithm to break these systems in polynomial time, and we do not expect NP to be P, we can use this problem to build a trapdoor function. However, there are many NP-Hard problems that, in particular cases, can be solved in polynomial time, for example the SAT problem (we know a polynomial time algorithm to solve 2-SAT), but this is not the case for the MQ Problem.

For the multivariate cryptosystems the trapdoor will be a regular function $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$. Given $F : \mathbb{F}^n \to \mathbb{F}^m$ defined by n quadratic polynomials chosen randomly and given c in the range of F, it is difficult to find $a \in \mathbb{F}^n$ such that $F(a) = c$. To find a we need to solve the system $p_1(x) = c_1, ..., p_n(x) = c_n$ where p_i are the polynomials defining F. Maybe this looks the same problem as the above, but the difference is that we turn the quadratic polynomials to $q_i(x) = p_i(x) - c_i$ therefore q_i is not uniformly random, and the problem is not the same as the MQ. Even though they are not the same, we will make an assumption that the problem is as hard as the MQ problem and will use it to build our trapdoor function with an easy way to invert for those who know the secret key.

In the next subsections, we describe procedures to generate regular functions that are easy to invert with the secret key. There is no guarantee that these functions are easy to invert with only the secret, most procedures to generate trapdoors made until today are either invertible or the functions are not generated randomly.

In general constructions we will not be restricted to quadratic polynomials, in fact we will be showing a general construction that maps $\mathbb{F}^n \to \mathbb{F}^m$ with $m \geq n$ and not only this, the degree of polynomials will be ≥ 2.

4.2.1 Construction

Given a regular function $F : \mathbb{F}^n \to \mathbb{F}^m$ and two linear transformations $S : \mathbb{F}^m \to \mathbb{F}^m$ and $T : \mathbb{F}^n \to \mathbb{F}^n$, we define the bipolar construction as $P : \mathbb{F}^n \to \mathbb{F}^m = T \circ F \circ S$. Note that P is a regular function as well.

Let's assume F has the property that any equation $F(x_1, ..., x_n) = (c_1, ..., c_n)$ where $(c_1, ..., c_n) \in F(\mathbb{F}^n)$ can be solved easily. F cannot be our public key because anyone could solve it, so we need to create two linear transformations T, S to hide the structure of F. The main idea is that the function S mixes the variables and T will mix the equations, Figure 2 illustrates this.

Since $P^{-1} = S^{-1} \circ F^{-1} \circ T^{-1}$, someone can easily invert $P(x_1, ..., x_n) = (c_1, ..., c_n)$ knowing S, F, T. This means that we need to consider them as the secret information and P is the public information. If someone knows only P then it is like a map factorization, this problem is hard in general and is close related to the Jacobian conjecture (OLIVEIRA, 2018).

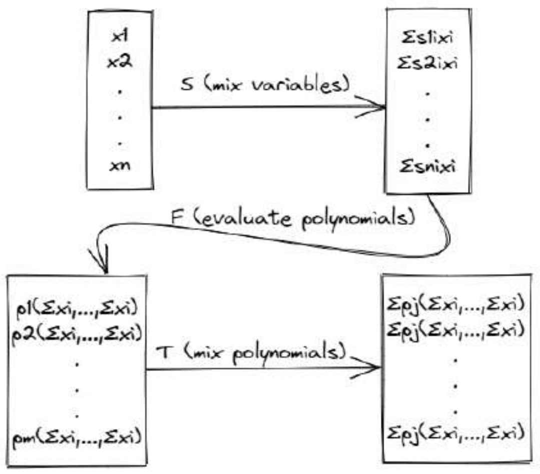

Figure 2 – Construction of multivariate trapdoor illustrated

If we choose F as a linear function, then P will be linear as well, this will make the trapdoor to be easily invertible with Gaussian elimination without having knowledge of S, F or T. Considering F to be linear would kill our trapdoor, so nobody would choose it, therefore that is not the only thing we need to think when building a trapdoor.

There are many ideas concerning how a secure F should be built, but many of them have a high computational power requirement. Another common problem is the public key size. An encryption scheme needs to be safe and efficient for any system, embedded or not. A common way to build F's is the lifting idea, which consists of using the correspondence of polynomials to build a trapdoor with $P = (P_1, ..., P_n)$ and $P_i = T_i \circ Drp(\mathcal{F}) \circ S$ where $\mathcal{F}$ is a homogeneous polynomial in the polynomial ring of an extension field of $\mathbb{F}$ and $Drp(\mathcal{F}) = \phi \circ \mathcal{F} \circ \phi^{-1}(x)$. Let's recall that ϕ is the same map presented in section 2.4.

4.3 HIDDEN FIELD EQUATIONS

As an example of multivariate encryption scheme we will describe and analyze the security of the Hidden Field Equations (HFE) cryptosystem. This scheme was proposed by Patarin (PATARIN, 1996).

Definition. The weight of a polynomial is the number of non-zero coefficients of it. For example the weight of $x^2 + y$ is 2.

Given a weight two low degree polynomial $\mathcal{F}(X)$ and a fixed bound B, in other words:

$$\mathcal{F}(X) = \sum_{q^i+q^j \leq B} a_{ij} X^{q^i+q^j}$$

Depending on B, this function can be inverted easily. Let's choose S and T as two secret linear transformations $\mathbb{F}^n \to \mathbb{F}^n$ and build the trapdoor function by computing the composition $P = T \circ \mathcal{F} \circ S$. This is the trapdoor of the HFE cryptosystem.

Encryption and Decryption. To make it simpler, the public key will be a set of multivariate polynomials $p_1, ..., p_n$ over $\mathbb{F}_q$, the message m will be a vector $(x_1, ..., x_n)$ in $\mathbb{F}_q^n$ and thus the ciphertext will be the evaluation of each polynomial on m, $c = (p_1(x_1, ..., x_n), p_2(x_1, ..., x_n), ..., p_n(x_1, ..., x_n)) \in \mathbb{F}_q^n$.

The decryption is simple since we know $S, \mathcal{F}$ and T (those are the private key). Just apply the trapdoor to the ciphertext and the result will be the message.

Security of HFE. We can write the polynomial $\mathcal{F}$ as:

$$\mathcal{F}(X) = \begin{pmatrix} X^{q^0} & X^{q^1} & \cdots & X^{q^{n-1}} \end{pmatrix} \begin{pmatrix} * & \cdots & * & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ * & \cdots & * & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} X^{q^0} \\ X^{q^1} \\ \vdots \\ X^{q^{n-1}} \end{pmatrix}$$

If we pay attention this looks the representation of a quadratic form for multiple variables but with X^{q^i}. The top left matrix that is $\lfloor \log_q B \rfloor \times \lfloor \log_q B \rfloor$ will be nonzero (represented by asterisks). The rank of this matrix will be as low as $\lfloor \log_q B \rfloor$, since B is a small number by construction, then the rank will be small too.

Proposition 4.3.1. (KIPNIS; SHAMIR, 1999) We can model a matrix P_i of dimensions $n \times n$ in $\mathbb{F}$ that represents the ith quadratic polynomial of the trapdoor function. In other words each $p_i(x) = x^T P_i x$, then there are $\lambda_1, ..., \lambda_n$ such that the matrix $\sum_{i=1}^{n} \lambda_i P_i$ has rank $\lfloor \log_q B \rfloor$.

This problem is called the MinRank problem and we will be discussing it in the next chapter. There are several algorithms, that include computing the Groebner basis, which can solve this problem. Since our rank is low it is an easy problem by computational means. In general this is a very hard problem, but in our case we know that there is a solution with small rank.

The main implication of this security analysis is that the trapdoor functions from HFE will never be considered a regular random function, which implies the attack to always work. Therefore, this scheme is not viable for a small B. There are several other schemes that are based on this one and that try not to be vulnerable to the MinRank attack by changing the rank or the design, so it will not rely on a low rank for the decryption, one such example is the ZHFE (PORRAS; BAENA; DING, 2014).

5 ATTACKS ON MPKC

In this chapter we go through some known attacks on MPKC, we will describe better how the attack on HFE works and in the end we will show results of a simple implementation of the attack proposed by Yasuhiko Ikematsu and Shuhei Nakamura to break a scheme proposed by Jiahui Chen. From now on we will be using the simpler notation of multiple polynomials rather than the notation with one polynomial on the extension field of $\mathbb{F}_q$. Those are the most common attacks on multivariate schemes:

- **Rank attacks:** a set of attacks which explore in particular the rank of the system of polynomial equations.
- **Direct attack:** this is simply be able to find pt in $P(pt) = ct$ where P is the public key and ct is the ciphertext.
- **Algebraic attacks:** any attack that changes the original polynomial system to an easier to solve.

5.1 RANK ATTACKS

5.1.1 The MinRank problem

This is a fundamental problem in Linear Algebra of finding a low-rank linear combination of matrices. The formal definition of this problem is as follows. Let $M_1, ..., M_m$ be matrices in $\mathcal{M}_{k\times k}(\mathbb{F}_q)$. The MinRank problem $MR(m, k, n, r, \mathbb{F}_q; M_1, ..., M_m)$ asks us to find $a = (a_1, ..., a_m) \in \mathbb{F}_q^m$ such that:

$$rank\left(\sum_{i=1}^{m} a_i M_i\right) \leq r$$

As stated in (BUSS; FRANDSEN; SHALLIT, 1999, p. 582-586) this problem is NP-Complete. A good property of this problem is that it can be modeled as a multivariate quadratic system, here we will describe how to do it but Shamir and Kipnis (KIPNIS; SHAMIR, 1999) goes way deeper on this modeling since they first proposed it.

Let's call H the linear combination $\sum_{i=1}^{m} a_i M_i$. If a is the solution, then H will have rank at most r and, by the rank-nullity theorem, the dimension of the kernel of H will be at least $n - r$. This means that there are at least $n - r$ linear independent vectors in the kernel of H. Let's fix those vectors as:

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 & 0 & ... & 0 & x_1^{(1)} & ... & x_r^{(1)} \end{bmatrix}^T$$

$$\vdots$$

$$\boldsymbol{x}^{(n-r)} = \begin{bmatrix} 0 & 0 & ... & 1 & x_1^{(n-r)} & ... & x_r^{(n-r)} \end{bmatrix}^T$$

Now we have the following multivariate quadratic system:

$$\left(\sum_{i=1}^{m} a_i M_i\right)_{k\times n} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ \boldsymbol{x}_1^{(1)} & \boldsymbol{x}_1^{(2)} & \cdots & \boldsymbol{x}_1^{(n-r)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_r^{(1)} & \boldsymbol{x}_r^{(2)} & \cdots & \boldsymbol{x}_r^{(n-r)} \end{pmatrix}_{n\times(n-r)} = \mathbf{0}_{k\times(n-r)}$$

consisting of $k(n-r)$ equations and $r(n-r)+m$ variables. This can be solved by finding the Groebner basis, as we commented in 3.3.2.

5.1.2 Example

From Proposition 4.3.1, we know that HFE is vulnerable to this type of attack. Here we describe an optimized algorithm that uses the solution of this problem to show the plaintext given the public key and the ciphertext.

1. Let $P = (p_1(x_1, ..., x_n), ..., p_n(x_1, ..., x_n))$ be the public key and $ct = (c_1, ..., c_n)$ be the ciphertext, both are elements of $\mathbb{F}_q$
2. Call E the set of all polynomials such that $E_i = p_i - c_i$.
3. Call $I = \langle E, x_0^q - x_0, ..., x_n^q - x_n\rangle$. Here we ensure that the zeros of I are in $\mathbb{F}_q$ and not in an extension of $\mathbb{F}_q$, as stated in 2.4 $x^q = x$, this will decrease the running time of the algorithm.
4. Run a reduced Groebner basis algorithm on I, obtaining G.
5. Run algorithm 3 on G, giving S.
6. One element of S will be the plaintext.

5.2 JIAHUI CHEN ET AL. CRYPTOSYSTEM

In this section it will be described a cryptosystem and an attack to it, as well as tests with different parameters for the key generation.

5.2.1 Construction

Here we describe the construction of Chen et al. encryption (CHEN et al., 2020). Starting from the general construction, we have $\mathbb{F}$ a finite field with q elements, C a $(n+1)\times n$ random matrix with each element $c_{i,j} \in \mathbb{F}$ and $C' \in \mathbb{F}^{n\times n}$ an invertible matrix where elements are $c'_{i,j} = 2(c_{i,j} - c_{i+1,j})$. The trapdoor consists of two random linear transformations S, T and a regular function $F' = (f_1, ..., f_{n+1})$. Each f_i is a quadratic polynomial on n variables:

$$
\begin{aligned}
f_1 &= (x_1 - c_{1,1})^2 + ... + (x_n - c_{1,n})^2;\\
f_2 &= (x_1 - c_{2,1})^2 + ... + (x_n - c_{2,n})^2;\\
&\vdots\\
f_{n+1} &= (x_1 - c_{n+1,1})^2 + ... + (x_n - c_{n+1,n})^2.
\end{aligned}
$$

By the multivariate trapdoor function definition, F' must be easily invertible thus we would like to solve these equations for an element $y = (y_1, ..., y_{n+1}) \in \mathbb{F}^{n+1}$:

$$
\begin{cases}
f_1(x_1, ..., x_n) = y_1;\\
f_2(x_2, ..., x_n) = y_2;\\
\vdots\\
f_{n+1}(x_1, ..., x_n) = y_{n+1}.
\end{cases}
\tag{5.1}
$$

If we take the differences we will have:

$$
f_{i+1}(x_1, ..., x_n) - f_i(x_1, ..., x_n) = 2(c_{i,1} - c_{i+1,1})x_1 + ... + 2(c_{i,1} - c_{i+1,1})x_n + \sum_{k=1}^{n}(c_{i+1,k}^2 - c_{i,k}^2)
$$

$$
\begin{pmatrix} y_2 - y_1 \\ \vdots \\ y_{n+1} - y_n \end{pmatrix} = C' \begin{pmatrix} x_1 \\ \vdots \\ x_{n+1} \end{pmatrix} \begin{pmatrix} \sum_{k=1}^{n}(c_{2,k}^2 - c_{1,k}^2) \\ \vdots \\ \sum_{k=1}^{n}(c_{n+1,k}^2 - c_{n,k}^2) \end{pmatrix}.
$$

Since we have a system of linear equations and C' is invertible, this system can be solved easily. But F' is still not our central map, it is just a general idea of the encryption scheme, so let's define now the central map. Let a, s and $m = n + 1 - a + s$ be positive integers. Let's choose s quadratic polynomials uniformly at random $(g_1, ..., g_s) \in \mathbb{F}$ each of them with n variables. Then our central map is $F = (f_1, ..., f_{n+1-a}, g_1, ..., g_s)$, note that this map is $\mathbb{F}^n \to \mathbb{F}^m$.

With the central map defined, we can define the public key $P = S \circ F \circ T$. Our secret key will be $F' = (f_1, ..., f_{n+1}), S, T$ and the polynomials $(g_1, ..., g_s)$. Note that we don't really need the random polynomials, but this will make the decryption easier.

Code 1 – Key generation implementation

```
import sys
q,n,a,s = (17,59,10,25)
m = n+1-a+s
FF = GF(q)
R = PolynomialRing(FF, ["x{}".format(i) for i in range(n)])
xs = R.gens()

def keygen():
  while True:
    C = random_matrix(FF, n+1, n)
    if matrix(FF, [2*C[i]-2*C[i+1] for i in range(n)]).is_invertible():
      break

  FC = []
  for i in range(n+1):
    p = 0
    for j in range(n):
      p += (xs[j] - C[i][j])^2
    FC.append(p)

  while True:
    S_lin = random_matrix(FF, n, n)
    if S_lin.is_invertible():
      break
  S_trans = (FF^n).random_element()
  S = (S_lin, S_trans)

  while True:
    T_lin = random_matrix(FF, m, m)
    if T_lin.is_invertible():
      break
  T_trans = (FF^m).random_element()
  T = (T_lin, T_trans)

  G = []
  for i in range(s):
    G.append(R.random_element(degree=2, terms=Infinity))
  F = FC[:n+1-a] + G

  P = S[0]*vector(xs) + S[1]
  v = []
  for i in range(len(F)):
    v.append(F[i](*P))
  P = T[0]*vector(v) + T[1]

  return (P, (C, G, S, T))
```

Key generation. Code 1. Now we go through the steps to generate a key pair for the scheme.

1. Set the parameters q, n, a, s and create the algebraic structures, such as $\mathbb{F}$.
2. Generate a random matrix C such that C' is invertible.
3. Compute two random linear transformations S, T, this can be done by choosing a random matrix and checking if it is invertible.
4. Compute F' and let F'_α be a list with the first $n+1-a$ polynomials. Compute $G = (g_1, ..., g_s)$.
5. $F = F' \cup G$.
6. Compute $P = S \circ F \circ T$
7. Return P as the public key and C, G, S, T as the secret key.

Encryption. The encryption is simply $P(x_1, ..., x_n)$ for a plaintext $x_1, ..., x_n$. Imagine having a plaintext "Hello World". Then we need to convert the string to a numeric representation and convert again to a polynomial in $\mathbb{F}^n$. Sometimes the message is too big to do this, then we need to separate it into blocks.

Decryption. Code 2. To decrypt the ciphertext ct we do the following, this seems to be really slow, the authors mentions that the time complexity of decryption is $O(q^a n^3)$ and analyzing the algorithm below we can conclude something close to this.

1. Compute a vector $cv = (\sum_{i=1}^{n} c_{2,i}^2 - c_{1,i}^2, ..., \sum_{i=1}^{n} c_{n+1,i}^2 - c_{n,i}^2)$.
2. Compute $aux = T^{-1} * ct$.
3. For each element k in $\mathbb{F}^a$:
   - 3.1 Let A to be the first $n+1-a$ elements of aux and $G = A + k$.
   - 3.2 Compute a vector $Gd = (G_2 - G_1, ..., G_m - G_{m-1})$.
   - 3.3 Compute $d = C'^{-1}(Gd - cv)$
   - 3.4 Now we check if all polynomials from G evaluate to the right solution, if that is true, the decryption is $S^{-1}d$.

Code 2 – Decryption implementation

```
def decrypt(cipher, sk):
  C, G, S, T = sk
  C2I = matrix(FF, [2*C[i]-2*C[i+1] for i in range(n)]).inverse()
  cv = []
  for i in range(n):
    cc = 0
    for j in range(n):
      cc += C[i+1][j]^2 - C[i][j]^2
    cv.append(cc)
  cv = vector(cv)
  g1 = T[0].inverse()*(cipher - T[1])
  for g2 in FF^a:
    g = vector(list(g1)[:n+1-a]+list(g2))
    g_diff = vector([g[i+1]-g[i] for i in range(len(g)-1)])
    d = C2I * (g_diff-cv)
    for i,j in zip(G,g1[n+1-a:]):
      if i(*d) != j:
        break
      else:
        return S[0].inverse() * (d - S[1])
```

With a quick analysis we see that we need to go through all elements in $\mathbb{F}^a$, and for each of them we need to evaluate all polynomials, therefore the complexity is something like $O(q^a(m+n+sn^2)) = O(q^a sn^2)$.

5.2.2 Algebraic Attack

In this subsection, we describe the algebraic attack proposed by (IKEMATSU; NAKAMURA, 2022). It will reduce the general MQ problem to one that is easier to solve and show that we need a bigger a value in order to increase the security of this scheme.

Let's name $h_i = f_{i+1} - f_i$, for each i from 1 to $n-a$. These polynomials will have degree one because of the definition of f_i and the set $\{h_1, ..., h_{n-a}\}$ is linear independent because C' is invertible.

Result 1. The subspace $SpanF = \{f_1, ..., f_{n+1-a}, g_1, ..., g_s\}$ is generated by

$$\{h_1, ..., h_{n-a}, f_1, g_1, ..., g_s\},$$

hence we have $n-a$ degree one polynomials and $s+1$ quadratic polynomials.

Let $SpanP$ be the subspace generated by $\{p_1, ..., p_m\}$. Since S and T are invertible, we have $SpanP = SpanF \circ S$ and, by result 1, $SpanP$ is generated by

$$\{h_1 \circ S, ..., h_{n-a} \circ S, f_1 \circ S, g_1 \circ S, ..., g_s \circ S\},$$

$SpanP$ is linear independent because S is invertible.

This means that we can get a linear independent set of $n-a$ degree one polynomials and s quadratic polynomials from the public key P. Hence, this system of equations can be remodeled into an easier one.

Let $y = (y_1, ..., y_m)$ be the ciphertext. The first thing to do is to find the linear equations, to do this we have to solve the following system of m variables $z_1, ..., z_m$:

$$\sum_{i=1}^{m} z_i \cdot \text{Quad}(p_i) = 0 \iff \sum_{i=1}^{m} z_i p_i \quad \text{has degree 1} \tag{5.2}$$

Where $\text{Quad}(p_i)$ is the quadratic part of p_i, to find this part we loop through the polynomial and get the coefficient c_i for each term $c_i x_i^2$.

Let $a_1, ..., a_m$ be a solution of equation (5.2). Then $\sum_{i=1}^{m} a_i p_i$ is a degree one polynomial and as we saw before the dimension of the space generated by such polynomials is $n-a$. This also means that the dimension of the solution space is $n-a$ too.

Let's choose a basis $\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(n-a)} \in \mathbb{F}^n$ of the solution space that we got from solving (5.2) then we obtain:

$$\begin{gathered} k_1(x_1, ..., x_n) = \boldsymbol{z}_1^{(1)} p_1 + ... + \boldsymbol{z}_m^{(1)} p_m \\ \vdots \\ k_{n-a}(x_1, ..., x_n) = \boldsymbol{z}_1^{(n-a)} p_1 + ... + \boldsymbol{z}_m^{(n-a)} p_m \end{gathered}$$

Then $SpanP$ is generated by $k_1, ..., k_{n-a}$ and other $s+1$ polynomials. If we plug into the direct attack modeling, we will have the following system:

$$\begin{gathered} p_1 - y_1 = 0 \\ \vdots \\ p_m - y_m = 0 \\ \sum_{i=1}^{m} \boldsymbol{z}_i^{(1)} p_i - \sum_{i=1}^{m} \boldsymbol{z}_i^{(1)} y_i = 0 \\ \vdots \\ \sum_{i=1}^{m} \boldsymbol{z}_i^{(n-a)} p_i - \sum_{i=1}^{m} \boldsymbol{z}_i^{(n-a)} y_i = 0 \end{gathered}$$

We can solve this system with a Groebner basis algorithm (see Section 3.3.2) and it will be faster than the direct attack. The full algorithm is below and the implementation is at Code 3.

1. Given $P = (p_1, ..., p_m)$ as the public key and $d = (d_1, ..., d_m)$ the ciphertext.
2. Get the basis $\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(n-a)}$, where $\boldsymbol{z}^{(i)} = (z_1^{(i)}, ..., z_m^{(i)}) \in \mathbb{F}^m$, of the kernel of the matrix $\mathbb{F}^{m \times n}$ given by the quadratic part of the public key.

3. Get the Groebner basis G of the ideal $I = \langle p_1 - d_1, ..., p_m - d_m, \sum_{i=1}^{m} \boldsymbol{z}_1^{(1)} p_i - \sum_{i=1}^{m} \boldsymbol{z}_i^{(1)} d_i, ..., \sum_{i=1}^{m} \boldsymbol{z}_i^{(n-a)} p_i - \sum_{i=1}^{m} \boldsymbol{z}_i^{(n-a)} d_i \rangle$.
4. The reduced Groebner basis G will already give us a valid result, but one can run algorithm 3 on G to effectively find the plaintext.

Code 3 – Implementation of the algebraic attack

```
q,n,a,s = (3,59,10,25)
m = n+1-a+s
FF = GF(q)
R = PolynomialRing(FF, ["x{}".format(i) for i in range(n)])
xs = R.gens()

## get quadratic part of poly
def Quad(p):
  return [p.monomial_coefficient(x**2) for x in xs]

quadz = [Quad(p) for p in P]

## solve sum(z_i*quad(pi)) = 0
mat = matrix(FF, m, n, lambda i, j: quadz[i][j])
krnl = mat.kernel()

def decrypt_block(d):
  ks = [
    sum((z[i] * P[i] for i in range(m))) - sum((z[i] * d[i] for i in
       range(m)))
    for z in krnl.basis()
  ]
  RI = R.ideal([p - v for p, v in zip(P, d)] + ks)
  # solve quadratic
  solution = RI.groebner_basis()
  ans = []
  for term, x in zip(solution, xs):
    ans.append(FF(x-term))
  return ans
```

5.2.3 Experiments and results

These are the time that the attack spent and parameters used to generate the key pairs. The attack runs faster than the decryption algorithm in the general case. All tests were run on Intel I7 4720 2.6GHz Notebook CPU with SageMath 9.6 (The Sage Developers, 2022), the Groebner basis algorithm was either Buchberger's using his criteria and F4 with FGLM (specified on table header), we ran 200 times each algorithm and we show the worst performed test out of 99% of them.

Parameters (q,n,a,s)	KeyGen + Enc	Buchberger	F4&FGLM
$\mathbf{(3, 59, 10, 25)}$	6.29s	4.48s	0.42s
$(3, 59, 15, 25)$	6.29s	200.2s	51.1s
$(3, 59, 12, 47)$	11.5s	6.58s	0.58s
$\mathbf{(3, 83, 12, 27)}$	62.8s	42.5s	1.81s
$(5, 83, 10, 47)$	122.3s	21.6s	2.99s
$(3, 83, 15, 27)$	32.3s	305.2s	127s

Table 1 – Running time of the algebraic attack on Jiahui Chen et al. cryptosystem (in bold the authors parameters for 80 and 128 bit security respectively)

As we can see on Table 1, the algebraic attack increases the complexity when we increase the value of a. $a \leq s$ is always true, because if not the system will be degenerate and the decryption failure rate would be $\geq \frac{1}{q}$ as the authors stated in (CHEN et al., 2020, p. 378). Even the most primitive implementation of the Groebner basis finding algorithm (Buchberger) was able to break the authors parameters in a feasible time and even faster than the key generation time.

To overcome the algebraic attack it was proposed (in (IKEMATSU; NAKAMURA, 2022)) a fortified parameter $(3, 59, 32, 47)$. Even though the algebraic attack time complexity would be of 80 bits, using this parameter, the direct attack would be valid if we use a Groebner basis algorithm implementation that is CPU bound instead of a memory bound.

6 CONCLUSION

In this work we implement and describe a set of attacks to multivariate encryption schemes, not only giving a practical overview but also providing a theoretical base to understand and modify them.

The results presented tell us that some attacks run better than the decryption and key creation algorithms. It is also implicit that it is very hard to build secure and efficient multivariate encryption schemes, in most schemes there is a trade-off between security and efficiency.

We believe that the scheme proposed by Chen can be broken even for the fortified parameters proposed by Ikematsu. Since most Groebner basis algorithms implementations are bounded by RAM, an improvement of the memory usage could lead to a direct attack on the fortified parameters.

REFERENCES

BUSS, J. F.; FRANDSEN, G. S.; SHALLIT, J. O. The computational complexity of some problems of linear algebra. **Journal of Computer and System Sciences**, v. 58, n. 3, p. 572–596, 1999. ISSN 0022-0000. Disponível em: https://www.sciencedirect.com/science/article/pii/S0022000098916087.

CHEN, J. et al. A new encryption scheme for multivariate quadratic systems. **Theoretical Computer Science**, v. 809, p. 372–383, 2020. ISSN 0304-3975. Disponível em: https://www.sciencedirect.com/science/article/pii/S0304397520300025.

COUTINHO, S.; APLICADA, I. N. de Matemática Pura e. **Números inteiros e criptografia RSA**. IMPA, 2005. (Série de computação de matemática). ISBN 9788524401244. Disponível em: https://books.google.com.br/books?id=y9YTYAAACAAJ.

COX, D.; LITTLE, J.; O'SHEA, D. Ideals, varieties, and algorithms. an introduction to computational algebraic geometry and commutative algebra. 2007. Disponível em: https://link.springer.com/book/10.1007/978-0-387-35651-8.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. New York, NY, USA: W. H. Freeman & Co., 1990. ISBN 0716710455.

GOOGLE. **HTTPS encryption on the web**. 2022. Disponível em: https://transparencyreport.google.com/https/overview.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: **Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing**. New York, NY, USA: Association for Computing Machinery, 1996. (STOC '96), p. 212–219. ISBN 0897917855.

IKEMATSU, Y.; NAKAMURA, S. Security analysis via algebraic attack against "a new encryption scheme for multivariate quadratic system". In: GIRI, D. et al. (Ed.). **Proceedings of the Seventh International Conference on Mathematics and Computing**. Singapore: Springer Singapore, 2022. p. 9–21. ISBN 978-981-16-6890-6.

KIPNIS, A.; SHAMIR, A. Cryptanalysis of the hfe public key cryptosystem by relinearization. In: **CRYPTO**. [S.l.: s.n.], 1999.

LANG, S. **Algebra**. 3. ed. New York: Springer-Verlag, 2002.

LIDL, R.; NIEDERREITER, H. Book. **Finite fields / Rudolf Lidl, Harald Niederreiter ; foreword by P.M. Cohn**. 2nd ed.. ed. Cambridge University Press Cambridge ; New York, 1997. xiv, 755 p. : p. ISBN 0521392314. Disponível em: http://www.loc.gov/catdir/toc/cam029/96031467.html.

MATSUMOTO, T.; IMAI, H. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: BARSTOW, D. et al. (Ed.). **Advances in Cryptology — EUROCRYPT '88**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. p. 419–453. ISBN 978-3-540-45961-3.

OLIVEIRA, W. M. F. The jacobian conjecture à la zp. 2018. Disponível em: http://hdl.handle.net/1843/EABA-AVQES5.

PATARIN, J. Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): Two new families of asymmetric algorithms. In: MAURER, U. (Ed.). **Advances in Cryptology—EUROCRYPT 96**. [S.l.]: Springer-Verlag, 1996. (Lecture Notes in Computer Science, v. 1070), p. 33–48.

PORRAS, J.; BAENA, J.; DING, J. Zhfe, a new multivariate public key encryption scheme. In: . [S.l.: s.n.], 2014. v. 8772, p. 229–245. ISBN 978-3-319-11658-7.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM Rev.**, v. 41, n. 2, p. 303–332 (electronic), 1999. ISSN 0036-1445.

The Sage Developers. **SageMath, the Sage Mathematics Software System (Version 9.6)**. [S.l.], 2022. Disponível em: https://www.sagemath.org.