UNIVERSITÀ DEGLI STUDI
DI GENOVA

DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA MAGISTRALE IN MATEMATICA

TESI DI LAUREA
IN
MATEMATICA

Description and algebraic cryptanalysis of the RYDE signature scheme

RELATORE:
Dr. rer. nat. Alessio Caminata

LAUREANDO:
Andrea Coppo

CORRELATORE:
Prof. Alessandro Oneto

ANNO ACCADEMICO 2025 - 2026

Introduction

In modern society, where digital communication is widespread, guaranteeing its security is of crucial importance. Moreover, the constant development of new technologies enables the construction of new attacks which threaten the cryptographic schemes currently in use. With the advent of large-scale quantum computers, problems such as integer factorization or discrete logarithm problem will become solvable exponentially faster than is possible with classical computers. Consequently, currently used public-key cryptography, such as RSA and ECC, will become obsolete. Hence, the new goal of finding and standardizing new cryptographic schemes that can resist quantum attacks has assumed a primary role. For this reason, in December 2016 the National Institute of Standards and Technology (NIST) launched a Post-Quantum Cryptography Standardization process. After several evaluation rounds, NIST selected the first set of post-quantum algorithms for standardization in both encryption and digital signatures. However, the approved signature schemes are based on lattice problems and hash functions. To diversify the mathematical structures underlying signature schemes, in September 2022 NIST launched an ongoing call for additional digital signature schemes.

RYDE is a digital signature scheme submitted in response to this call. It successfully reached the second round of evaluation, but did not advance to the third round in May 2026. Despite this, it remains of great interest as it is based on code-based cryptography. Code-based cryptography is a branch of cryptography that relies on problems from algebraic coding theory, which is the study of algebraic structures that enable the detection and correction of errors when a message is transmitted through a noisy channel. In particular, RYDE relies on a restriction of the Rank Syndrome Decoding (RSD) problem. Intuitively, this consists of recovering the original codeword x from a received word $y = x + e$, where the error vector e is required to have a low weight with respect to the rank metric. To build a signature from the RSD problem,

RYDE first utilizes an interactive protocol between two parties: a Prover and a Verifier. The Prover wants to convince the Verifier that he knows a solution for a specified RSD problem without revealing any information about the solution itself. A protocol of such type is known as a zero-knowledge proof. Finally, the Fiat-Shamir transform is applied to convert this interactive zero-knowledge protocol into a non-interactive digital signature scheme.

The goal of this thesis is to formalize the RYDE signature scheme and analyze its security. In particular, we focus on the hardness of the restriction of RSD problem underlying the scheme. To evaluate its computational complexity we first present the MaxMinors modeling, an algebraic technique that allow to convert the restriction of the RSD problem into a multivariate polynomial system. Finally, we estimate the complexity of solving the resulting system using Gröbner bases.

Section 1 describes preliminary notions in cryptography, zero-knowledge protocols and the Fiat-Shamir Transform.

Section 2 provides the necessary coding theory background, as well as a description and analysis of the properties of the RSD problem and its restriction RSD_s.

In Section 3, we describe the proof system upon which RYDE is built. We first present a generalized version that encodes the secret as roots of quadratic polynomials. We then present the proof system encoding RSD_s instances and we describe the conversion into the RYDE signature scheme.

Section 4 covers the theory of Gröbner bases required to estimate the solving complexity of the RSD_s problem. Finally, Section 5 introduces the MaxMinors modeling. We discuss two different approaches, present the experimental results of our complexity estimations and outline future research directions. Appendix A details the theoretical foundations of the final section.

Contents

Notation

Throughout this document, we adopt the following standard notation.

- **Sets and Sampling:** For a set S, we denote the cardinality of the set as $|S|$. The notation $x \leftarrow_{\$} S$ indicates that an element x is sampled uniformly at random from the set S.
- **Finite Fields:** For q a power of a prime, $\mathbb{F}_q$ denotes the finite field of cardinality q.
- **Binary Operations:** The symbol $\oplus$ denotes the standard addition over the field $\mathbb{F}_2$. The concatenation of two binary strings a and b is denoted as $a \parallel b$.
- **Assignments:** We use the symbol $\coloneqq$ to denote a definition and the symbol $\leftarrow$ to denote the assignment of a computed value to a variable.
- **Ring of integers modulo n:** Given a natural number $n > 1$, we use the symbol $\mathbb{Z}_n$ to denote the ring of integers modulo n;
- **Euler's function:** We use the symbol ϕ to denote the function that maps a natural number n into the cardinality of invertible elements in $\mathbb{Z}_n$.

1 Zero-Knowledge Protocols and the Fiat-Shamir Transform

In this section we explore the connection between zero-knowledge protocols and digital signature schemes. We begin by recalling the concept of a digital signature scheme, alongside the fundamental notions of hash function and pseudorandom generator. Subsequently, we introduce the concept of zero-knowledge protocols and describe how to construct a signature scheme out of them. This process was first introduced by Fiat and Shamir in [FS87]. We will present their original identification scheme and the deriving signature scheme. Finally, we will generalize this construction to general 5 round identification schemes.

1.1 Digital signature scheme

Given a public key k', a digital signature scheme allows the signer, who knows a secret key corresponding to k', to generate a signature for a given document, respecting the following:

1. the signature cannot be forged by anyone who does not know the private key;
2. anyone who has the public key k' can check that the signature is correct.

Formally, a digital signature scheme consists of:

- a set of possible documents $\mathcal{M}$;
- a set of possible signatures $\mathcal{A}$;
- a set of possible keys $\mathcal{K}$;
- a signing function $sig : \mathcal{K} \times \mathcal{M} \to \mathcal{A}$;

- a verification function $ver : \mathcal{K} \times \mathcal{M} \times \mathcal{A} \to \{\text{true}, \text{false}\}$;
- a set $S \subset \mathcal{K} \times \mathcal{K}$ such that:

$$\text{ver}(k', M, y) = \begin{cases} \text{true} & \text{if } y = \text{sig}(k, M) \text{ for a pair } (k, k') \in S \\ \text{false} & \text{if } y \neq \text{sig}(k, M) \text{ for all pairs } (k, k') \in S \end{cases}$$

To sign a document $M \in \mathcal{M}$, the signer generates a pair of keys $(k, k') \in S$, where k is the private key known only to the signer and k' is the public key. The signer computes the signature $\sigma = \text{sig}(k, M)$ and the verifier checks whether $\text{ver}(k', M, \sigma) = \text{true}$. The property of non-forgeability relies on the hardness of obtaining a private key k corresponding to a public key k'. Consequently, digital signature schemes are based on problems that are computationally hard to solve. One of the most famous problems of this type in modern cryptography is the RSA problem. The RSA problem is deeply connected with the problem of factoring a sufficiently large integer n. As an example of a digital signature scheme, we present the RSA scheme.

1.1 Example (RSA signature scheme)
Let p, q be two distinct prime numbers and let $n = pq$, $M = A = \mathbb{Z}_n, K = \mathbb{Z}$ and

$$S = \{(e, d) \in \mathbb{Z} \times \mathbb{Z} \mid ed \equiv 1 \pmod{\phi(n)}\}.$$

We define the signature function as

$$\begin{aligned} \text{sig} : \mathbb{Z} \times \mathbb{Z}_n &\to \mathbb{Z}_n \\ (d, x) &\mapsto x^d \pmod{n} \end{aligned}$$

and the verification function as

$$\begin{aligned} \text{ver} : \mathbb{Z} \times \mathbb{Z}_n \times \mathbb{Z}_n &\to \{True, False\} \\ (e, x, y) &\mapsto \left(y^e \stackrel{?}{\equiv} x \pmod{n}\right) \end{aligned}$$

The secret parameters are $d, p, q, \phi(n)$. The public ones are n, e.

To prove the correctness of the scheme, we evaluate the verification function on a valid signature $y = x^d \pmod{n}$. *Consequently, we need to show that* $y^e \equiv x \pmod{n}$*:*

$$y^e \equiv (x^d)^e \equiv x^{ed} \stackrel{?}{\equiv} x \pmod{n}$$

We analyze the congruence both modulo p *and* q*.*

- *If* $p \mid x$*, then* $x \equiv 0 \pmod{p}$*. Trivially,* $x^{ed} \equiv 0^{ed} \equiv 0 \equiv x \pmod{p}$*.*
- *If* $p \nmid x$*, then* $\gcd(x, p) = 1$*. By Fermat's Little Theorem, we have* $x^{p-1} \equiv 1 \pmod{p}$*. Since* $ed = 1 + k(p-1)(q-1)$*, we obtain:*

$$x^{ed} \equiv x^{1+k(p-1)(q-1)} \equiv x \cdot (x^{p-1})^{k(q-1)} \equiv x \cdot 1^{k(q-1)} \equiv x \pmod{p}$$

In both cases, we establish that $x^{ed} \equiv x \pmod{p}$*.*

By applying the same logic for the prime q*, we obtain* $x^{ed} \equiv x \pmod{q}$*.*

Since p *and* q *are distinct primes, the Chinese Remainder Theorem (CRT) guarantees that if the congruence holds modulo* p *and modulo* q*, it must also hold modulo their product* $n = pq$*. Therefore:*

$$x^{ed} \equiv x \pmod{n} \quad \forall x \in \mathbb{Z}_n$$

A malicious signer, who possesses the factorization of n*, is able to reconstruct* $\phi(n)$*, hence using the Euclidean algorithm is able to reconstruct the inverse modulo* $\phi(n)$ *and, consequently, to forge the signature.*

In 1994 Peter Shor developed an algorithm that computes the factorization of an integer n in polynomial time, meaning the number of steps required is polynomial in $\log n$. Consequently, the security of various cryptographic schemes, such as RSA, were compromised. Although current quantum computers are not yet powerful enough to execute successful attacks, the threat has motivated the transition to post-quantum cryptography.

In order to construct signature schemes using new paradigms, such as those derived from zero-knowledge protocols, we first need to introduce the concept of hash functions.

1.2 Hash function

A cryptographic hash function h is a function that maps arbitrary long strings of bits into small ones of fixed length called digests and satisfies the following properties of security:

1. **Preimage Resistant**: given a digest y it is computationally difficult to find a string of bits x such that $h(x) = y$;
2. **Second Preimage Resistant**: given a string of bits x it is computationally difficult finding another string of bits x', $x \neq x'$ such that $h(x) = h(x')$;
3. **Collision Resistant**: it is computationally difficult finding $x, x' \in X, x \neq x'$ such that $h(x) = h(x')$.

Here we introduce the hash function used in RYDE.

SHA-3

In 2004 and 2005, attacks against numerous hash functions were published; in particular, the NIST-approved SHA-1 suffered critical vulnerabilities. In 2007, concerned about potential attacks on SHA-2, NIST announced the SHA-3 Cryptographic Hash Algorithm Competition. The Keccak function, designed by Joan Daemen, Michaël Peeters, Gilles Van Assche, and Guido Bertoni, was declared the winner of the competition in 2012. Below, we describe its functioning.

The SHA-3 function is based on two constructions: a bijective permutation σ (that is the Keccak-p permutation) acting on a bit string S of length b, called the state, and a sponge construction.

The **sponge construction** partitions the state into two components: $S = (S_r, S_c)$, where $S_r \in \mathbb{F}_2^r$ is called the *rate* (the part interacting with the message) and $S_c \in \mathbb{F}_2^c$ is the *capacity* (the strictly hidden part).

The evaluation of the function on a message M consists of three phases:

1. The message M is padded to a multiple of r and split into blocks $M_1, M_2, \ldots, M_k \in \mathbb{F}_2^r$.

2. The state is initialized as $S = (0^r, 0^c)$. For each block M_i, the state is updated by XORing the message block into the rate component and applying the permutation:

$$S \leftarrow \sigma(S_r \oplus M_i, S_c) \quad \text{for } i = 1, \ldots, k$$

3. To produce the output, the construction extracts the first r bits of S. If further bits are required, it applies the permutation $S \leftarrow \sigma(S)$, concatenates the new first r bits of S to the current output, and repeats this check until the target length is reached.

In the Keccak-p permutation, the state is conceptualized as a 3-dimensional array $A \in \mathbb{F}_2^{5\times5\times w}$, where $w = \lceil \frac{b}{25} \rceil$.

The permutation consists of a sequence of rounds, where each round updates the state A by applying five different algebraic operations.

1. The first operation consists of XORing each bit with the parities of two columns in the array.

2. The second operation consists of a rotation of the bits along the z-axis.

3. The third operation consists of a permutation of the lanes depending on the position (x, y).

4. The fourth operation modifies each bit based on the values of the next two bits in the same row.

5. The fifth operation consists of XORing the first lane of the state with a constant depending on the round.

The combination of the two constructions described above ensures the required security properties. While a security analysis is beyond the scope of this work, detailed security investigation can be found in [Ber+11a; Ber+11b]. A detailed description of SHA-3 can be found in [Dwo15].

An important application of hash functions are commitment schemes.

Commitment scheme

A commitment scheme is a two-party protocol where one party can lock in a value m by sending to the other party a commitment com and then in a second phase unlock it by revealing m. Moreover, the scheme needs to satisfy the following two properties:

- **Hiding:** An adversary observing only *com* gains zero knowledge about the secret m.
- **Binding:** The committer cannot cheat by finding a way to open the same commitment *com* to two different values, m and m'.

A hash function (or simply a preimage and second preimage resistant function) can be used to build a commitment scheme satisfying the two properties above. However, if the message space is small, simply committing $h(m)$ does not guarantee the hiding of the message. To solve this, one samples a random string r and computes $com = h(m \parallel r)$ and then, in the second phase, reveals both the message m and the string r.

1.2 Example (Coin flipping over the telephone)
An elementary example of a commitment scheme is the following protocol, which allows Alice and Bob to play a fair game of coin flipping over the telephone.

The protocol proceeds as follows:

1. *Alice chooses a move* $m \in \{0, 1\}$ *and a random string* r*, computes* $\mathrm{com} = h(m \parallel r)$ *using a hash function* h*, and sends* com *to Bob;*

2. *Bob makes his guess* $b \in \{0, 1\}$ *and sends it to Alice;*

3. *Alice reveals the original move* m *and the string* r*;*

4. *Bob computes* $\tilde{\text{com}} := h(m \parallel r)$ *and checks if* $\tilde{\text{com}} = \text{com}$*. If Alice's move is valid, then the winner is determined by checking* $b = m$*.*

Alongside hash functions, Pseudorandom Generator constitutes another important block for constructing secure signature schemes.

1.3 Pseudorandom Generator (PRG)

A Pseudorandom Generator (PRG) is a function PRG that maps a short input string (called the *seed*) to a longer output string satisfying the following property:

- **Pseudorandomness:** For a seed s chosen uniformly at random, there is only a negligible probability that any polynomial-time adversary can distinguish the output of PRG(s) from a string chosen uniformly at random.

Examples of PRGs are the SHAKE functions.

1.3 Example (SHAKE)
The SHAKE functions are derived from the same Keccak-p permutation and sponge construction used in SHA-3, with the difference that, while SHA-3 repeats the sponge construction until a fixed size (e.g. 256 bits) is reached, SHAKE repeats it until the requested arbitrary output length is fulfilled.

1.4 The Fiat-Shamir identification scheme

We want to define an interactive protocol between two parties: a Prover and a Verifier. Given a public statement x, the Prover wants to demonstrate knowledge of a witness w for x to the Verifier. Formally, we say that $w \in W$

is a **witness** for $x \in X$ if $R(x, w) = 1$ for a public polynomial-time relation $R : X \times W \to \{0, 1\}$.

Let $n = pq$ be a public odd integer, where p and q are two distinct prime numbers satisfying $p \equiv q \equiv 3 \pmod 4$ only known by the Prover. As we will show, this condition enables the Prover to compute square roots modulo n, allowing the generation of the secret keys. Let f be a public function that maps strings to natural numbers and k a public positive integer. We will denote by $\mathcal{P}$ the Prover and by $\mathcal{V}$ the Verifier.

The protocol proceeds as follows:

Setup Phase:

1. $\mathcal{P}$ chooses the identity string I and initializes a counter $d = 0$.
2. $\mathcal{P}$ computes $v_j \coloneqq f(I, j, d)$ for all $j \in \{1, \ldots, k\}$.
3. $\mathcal{P}$ checks that v_j is invertible and that the inverse of every v_j is a quadratic residue modulo n. If this does not hold, $\mathcal{P}$ increments d by 1 and repeats step 2 until valid values are found.
4. $\mathcal{P}$ computes the secret values s_j satisfying $s_j^2 \cdot v_j \equiv 1 \pmod n$ (using the factorization of n).
5. The public parameters are now $(I, d, v_1, \ldots, v_k)$. The secret key is $(s_1, \ldots, s_k)$.

Interactive protocol:

1. $\mathcal{P}$ picks a random $r \in \{0, \ldots, n-1\}$ and sends $x \coloneqq r^2 \pmod n$;
2. $\mathcal{V}$ generates a random challenge $(c_1, \ldots, c_k) \in \mathbb{F}_2^k$ and sends it to $\mathcal{P}$;
3. $\mathcal{P}$ computes and sends $y \coloneqq r \prod_{j:c_j=1} s_j \pmod n$;
4. $\mathcal{V}$ accepts the proof if and only if $x \equiv y^2 \prod_{j:c_j=1} v_j \pmod n$.

1.4 Remark (Probability of success)
Here, we analyze the probability of successfully generating valid secret keys s_j. For the protocol to proceed, two condition must hold for all $j \in \{1, \ldots, n\}$:

1. ***Invertibility:*** *For the setup phase it is necessary that v_j is invertible modulo n and, assuming that f outputs values uniformly at random, for p, q sufficiently large primes, this occurs with overwhelming probability $\mathbb{P}(\gcd(v_j, n) = 1) = \frac{\phi(n)}{n} = \frac{(p-1)(q-1)}{n} = 1 - \frac{p+q-1}{n}$. Since the Prover needs to compute s_j for all k components, this condition must hold for all $j \in \{1, \ldots, n\}$, which occurs with probability $(1 - \frac{p+q-1}{n})^k$.*

2. ***Quadratic Residues:*** *Let now $z_j \equiv v_j^{-1} \pmod{n}$. By the CRT (Chinese Remainder Theorem) z_j is a quadratic residue modulo n if and only if it is also a quadratic residue both modulo p and q. Since z_j is an invertible random element in $\mathbb{Z}_n^*$, this occurs with probability $\frac{(p-1)(q-1)}{4(p-1)(q-1)} = \frac{1}{4}$. Hence, the probability that all k inverses are quadratic residues is $(\frac{1}{4})^k$. The check of being a quadratic residue can be achieved efficiently using the Legendre symbol.*

   Since this probability drops exponentially with k, $\mathcal{P}$ repeats the process until all the s_j can be successfully computed.

1.5 Remark (Computation of the secret keys)
Once valid v_j are found, the Prover can compute the secret keys s_j efficiently proceeding as follows:

1. ***Inversion:*** *The inverse of v_j can be computed efficiently using the Euclidean algorithm.*

2. ***Square roots:*** *To find the square roots of z_j, one computes its square roots modulo p, denoted as $\pm x_p$, and modulo q, denoted as $\pm x_q$. Since $p \equiv q \equiv 3 \pmod{4}$, it holds that:*

$$x_p = z_j^{\frac{p+1}{4}} \qquad x_q = z_j^{\frac{q+1}{4}}.$$

Indeed, $z_j^{\frac{p+1}{2}} = z_j^{\frac{p-1}{2}+1} = z_j^{\frac{p-1}{2}} z_j$ *and, since* $\mathbb{F}_p$ *is a field and* z_j *is a quadratic residue modulo* p*, Euler's criterion states that* $z_j^{\frac{p-1}{2}} \equiv 1 \pmod{p}$*. Let now* x *be a square root of* $z_j \pmod{n}$*, then*

$$\begin{cases} x \equiv \pm x_p \pmod{p} \\ x \equiv \pm x_q \pmod{q} \end{cases}$$

Finally by CRT (Chinese Remainder Theorem) we obtain

$$x \in \{\pm(x_p uq \pm x_q vp) : uq + vp = 1\}.$$

Security:

Let $\tilde{\mathcal{P}}$ be a malicious prover who does not know the secret keys $(s_1, \ldots, s_k)$ and cannot compute square roots modulo n in polynomial time. If $\mathcal{V}$ follows the protocol then, $\mathcal{V}$ would accept the proof with a probability of at most $1/2^k$. This is because $\tilde{\mathcal{P}}$ can successfully cheat only by guessing the verifier's challenge vector $c \in \{0,1\}^k$ and sending

$$x = r^2 \prod_{j:c_j=1} v_j \pmod{n} \text{ and } y = r.$$

Indeed, to increase this probability, $\mathcal{P}$ would need to craft x that allow him to finding two responses y, y' to two possible different challenges $c, c' \in \mathbb{F}_2^k$. This is equivalent to require that

$$y^2 \prod_{j:c_j=1} v_j \equiv x \equiv (y')^2 \prod_{j:c'_j=1} v_j \pmod{n}.$$

By rearranging the terms, we obtain:

$$\left(\frac{y}{y'}\right)^2 \equiv \frac{\prod_{j:c'_j=1} v_j}{\prod_{j:c_j=1} v_j} \pmod{n}.$$

Subsequently, the fraction $\frac{y}{y'}$ constitutes a square root of a product of public keys v_j and their inverses. Hence $\tilde{\mathcal{P}}$ has successfully computed a square root modulo n, that contradicts the assumption on $\tilde{\mathcal{P}}$.

Zero-knowledge:
Intuitively, the protocol does not leak any information because $\mathcal{V}$ obtains only $x = r^2$ and $y = r\prod_{j:c_j=1} s_j$. Consequently, the multiplication by r masks the product $\prod_{j:c_j=1} s_j$ and $\mathcal{V}$ does not learns any information about the secret keys $(s_1, \ldots, s_k)$.

Fiat-Shamir Identification Protocol

Prover $\mathcal{P}(s_1, \ldots, s_k)$		**Verifier** $\mathcal{V}(v_1, \ldots, v_k)$
$r \leftarrow_\$ \{1, \ldots, n-1\}$		
$x \leftarrow r^2 \bmod n$		
	$\xrightarrow{\quad x \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \leftarrow_\$ \{0,1\}^k$
$y \leftarrow r \prod_{j:c_j=1} s_j \bmod n$		
	$\xrightarrow{\quad y \quad}$	
		Check $x \stackrel{?}{\equiv} y^2 \prod_{j:c_j=1} v_j \pmod n$

1.6 Example
Let $p = 19$ and $q = 31$ be the two primes, only known by $\mathcal{P}$, and $n = 589$ the public modulus. Moreover, let $k = 5$.

$\mathcal{P}$ finds the public keys $v = [235, 225, 283, 1, 180]$, which are quadratic residues modulo n, and computes the secret keys $s = [102, 157, 16, 1, 25]$.

Interactive Protocol:

1. *$\mathcal{P}$ picks a random $r = 73$ and sends $x \equiv 73^2 \equiv 28 \pmod{589}$;*
2. *$\mathcal{V}$ sends the challenge $c = [1, 0, 1, 0, 1]$;*

3. $\mathcal{P}$ computes and sends the response

$$y \equiv r \prod_{j:c_j=1} s_j \equiv 73 \cdot 102 \cdot 16 \cdot 25 \equiv 416 \pmod{589};$$

4. $\mathcal{V}$ verifies the proof by checking if $x \equiv y^2 \prod_{j:c_j=1} v_j \pmod{n}$. That is:

$$416^2 \cdot (235 \cdot 283 \cdot 180) \equiv 479 \cdot 64 \equiv 28 \pmod{589}.$$

The interactive scheme introduced above can be converted into a non-interactive digital signature scheme.

1.4.1 Fiat-Shamir Signature

This can be done by substituting the random challenge generated by $\mathcal{V}$ with a cryptographic hash function h. In this setting, $\mathcal{P}$ acts as the signer of a document and $\mathcal{V}$ acts as the verifier checking the validity of the signature.
Setup Phase:

1. $\mathcal{P}$ chooses the a message I and initializes a counter $d = 0$.
2. $\mathcal{P}$ computes $v_j \coloneqq f(I, j, d)$ for all $j \in \{1, \ldots, k\}$.
3. $\mathcal{P}$ checks that v_j is invertible and that the inverse of every v_j is a quadratic residue modulo n. If this does not hold, $\mathcal{P}$ increments d by 1 and repeats step 2 until valid values are found.
4. $\mathcal{P}$ computes the secret values s_j satisfying $s_j^2 \cdot v_j \equiv 1 \pmod{n}$ (using the factorization of n).
5. The public parameters are now $(I, d, v_1, \ldots, v_k)$. The secret key is $(s_1, \ldots, s_k)$.

Sign Algorithm:

1. $\mathcal{P}$ picks a random $r \in \{0, \ldots, n-1\}$ and computes $x = r^2 \pmod{n}$.

2. $\mathcal{P}$ generates a random challenge by hashing x, I, d and the document M, i.e. $h(x, I, d, M) = (c_1, \ldots, c_k) \in \mathbb{F}_2^k$.

3. $\mathcal{P}$ computes $y \coloneqq r \prod_{j:c_j=1} s_j \pmod{n}$ and signs the document by sending $(M, I, d, c_1, \ldots, c_k, y)$.

Verify Algorithm:

1. $\mathcal{V}$ computes $v_j = f(I, j, d)$ for $j \in \{1, \ldots, k\}$.

2. $\mathcal{V}$ computes $\tilde{x} \coloneqq y^2 \prod_{j:c_j=1} v_j \pmod{n}$.

3. $\mathcal{V}$ verifies that $h(\tilde{x}, I, d, M) = (c_1, \ldots, c_k)$.

1.7 Example

We describe the Fiat-Shamir signature scheme derived from the example described in 1.6. Let $p = 19$ *and* $q = 31$ *be the two primes, only known by* $\mathcal{P}$*, and* $n = 589$ *the public modulus. Moreover, let* $k = 5$.

Suppose $\mathcal{P}$ *wants to sign a document* M *with identity* I *and counter* d*. The public keys are* $v = [235, 225, 283, 1, 180]$*, and the corresponding secret keys are* $s = [102, 157, 16, 1, 25]$.

Sign Algorithm:

1. $\mathcal{P}$ *picks a random* $r = 73$ *and computes* $x \equiv 73^2 \pmod{589} \equiv 28$.

2. $\mathcal{P}$ *generates the challenge by hashing* x, I, d, *and the document* M*. Assume for this example that the hash function outputs* $h(28, I, d, M) = c = (1, 0, 1, 0, 1)$.

3. $\mathcal{P}$ *computes the response* $y \equiv r \prod_{j:c_j=1} s_j \equiv 416$. $\mathcal{P}$ *signs the document by outputting the signature:* $(M, I, d, c, 416)$.

Verify Algorithm:

1. $\mathcal{V}$ *receives the signature* $(M, I, d, c = (1, 0, 1, 0, 1), y = 416)$ *and derives the public values* $v = [235, 225, 283, 1, 180]$ *using* $f(I, j, d)$.

2. $\mathcal{V}$ *computes* $\tilde{x} \equiv y^2 \prod_{j:c_j=1} v_j \pmod{n} \equiv 28 \pmod{589}$.

3. $\mathcal{V}$ *verifies the signature by checking if* $h(\tilde{x}, I, d, M)$ *matches the* c *provided in the signature. Since* $h(28, I, d, M) = (1, 0, 1, 0, 1)$*, the signature is accepted as valid.*

1.5 Zero knowledge identification schemes

We want to generalize the argument above. In this work, rather than relying on the heavy formalism of Interactive Turing Machines we model our interactive proofs as identification schemes. We will extend the definition of 3-round identification scheme to a 5-round one as required by the proof system used in RYDE protocol.

Let $x \in \mathcal{X}$ be a public statement and $w \in \mathcal{W}$ the corresponding private witness. The Prover $\mathcal{P}$ is specified by a tuple of three deterministic algorithms:

$$P_1 : \mathcal{X} \times \mathcal{W} \to \mathcal{M}_1 \times \mathcal{S}_1$$
$$P_2 : \mathcal{S}_1 \times \mathcal{C}_1 \to \mathcal{M}_2 \times \mathcal{S}_2$$
$$P_3 : \mathcal{S}_2 \times \mathcal{C}_2 \to \mathcal{M}_3$$

where $\mathcal{M}_i$ denote the public message spaces, $\mathcal{C}_i$ the challenge spaces from which $\mathcal{V}$ samples and $\mathcal{S}_i$ the private state spaces accessible only by $\mathcal{P}$.

The core objective is for $\mathcal{P}$ to convince $\mathcal{V}$ that he knows a witness w for the public statement x, without revealing any information regarding the secret itself. Consequently, the algorithms P_1, P_2 output public messages m_i to be transmitted to $\mathcal{V}$, alongside private states st_i, which contain information about w, and for this reason are kept secret by $\mathcal{P}$.

Similarly, the Verifier $\mathcal{V}$ consists of a random generator for the challenges and a deterministic verification algorithm

$$V : \mathcal{X} \times \mathcal{M}_1 \times \mathcal{C}_1 \times \mathcal{M}_2 \times \mathcal{C}_2 \times \mathcal{M}_3 \to \{0, 1\}$$

A 5-round identification scheme proceeds as follows:

1. **Round 1 (Commitment):** The Prover executes $P_1(x, w)$ to obtain a message m_1 and a private state st_1. It sends m_1 to the Verifier.

2. **Round 2 (First Challenge):** The verifier samples a random challenge c_1 from a challenge space $\mathcal{C}_1$ and sends it to the prover.

3. **Round 3 (First Response):** The prover executes $P_2(st_1, c_1)$ to compute a second message m_2 and a state st_2. It sends m_2 to the verifier.

4. **Round 4 (Second Challenge):** The verifier samples a second random challenge c_2 from a challenge space $\mathcal{C}_2$ and sends it to the prover.

5. **Round 5 (Final Response):** The prover executes $P_3(st_2, c_2)$ to compute the final response m_3, which is sent to the verifier.

Finally, the verifier computes the deterministic algorithm $V(x, m_1, c_1, m_2, c_2, m_3)$, which outputs 1 if the transcript is mathematically consistent, and 0 otherwise. We would call $(m_1, c_1, m_2, c_2, m_3)$ the transcript between $\mathcal{P}$ and $\mathcal{V}$.

Moreover, we require the protocol to satisfy:

- **completeness**, if w is a witness for x and both parties follow the protocol honestly, then $V(x, m_1, c_1, m_2, c_2, m_3) = 1$ with probability 1;

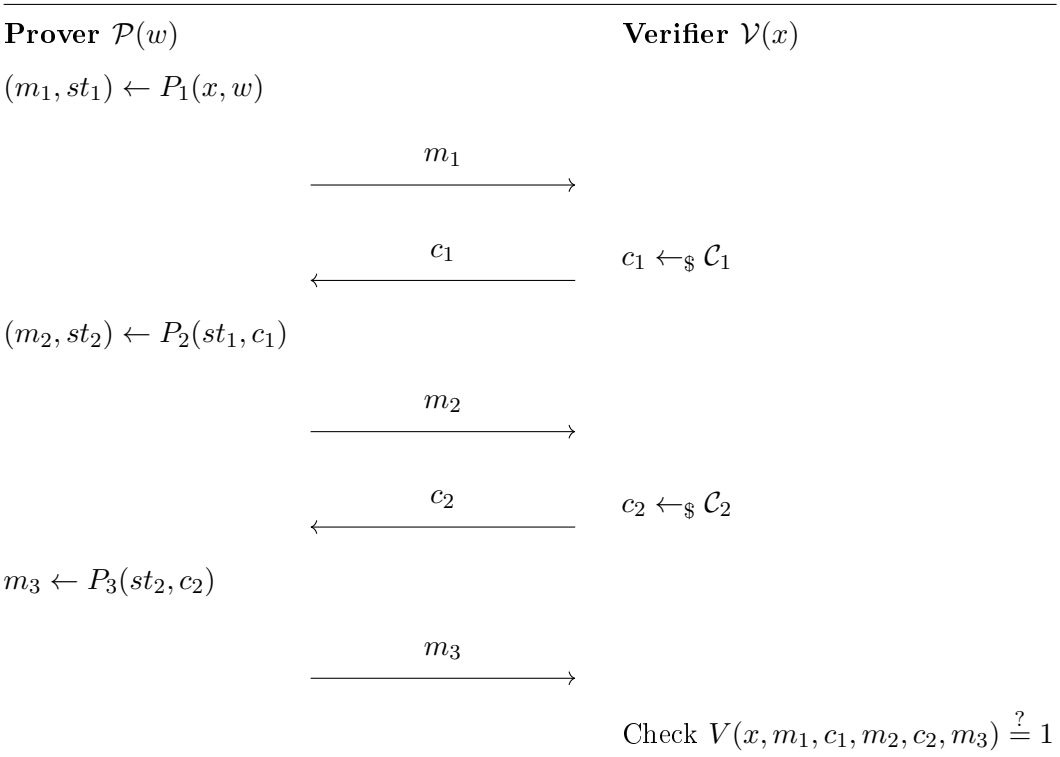
- **soundness**: if the prover does not know a witness for x, then, for every strategy a possibly malicious prover could adopt, the probability that $\mathcal{V}$ accepts the proof is negligible.

In addition, we require that the protocol has the **zero-knowledge property**: after the execution of the protocol $\mathcal{V}$ does not know anything about the secret witness w beyond the fact that it is a witness for x.

A protocol as above with this three properties is called a **zero-knowledge proof**.

While the above properties can be formalized using rigorous notions from probability and complexity theory, as detailed in [Gol01], the intuitive definitions provided here are sufficient to understand the cryptographic mechanisms underlying the RYDE signature scheme.

5-Round Interactive Protocol



1.6 Fiat-Shamir Transform

The Fiat-Shamir Transform is what allows to derive the signature scheme presented in Subsection 1.4. In this section we generalize this concept. We generalize the Fiat-Shamir Transform to demonstrate how it can be adapted to derive a non interactive digital signature scheme from an identification protocol. Assume that we have a 5-round protocol and we want to define

a Sign Algorithm and a Verify Algorithm. To this purpose, we remove the interactions by replacing the Verifier's random challenges with the output of a public cryptographic hash function h. Below, $\mathcal{P}$ wants to sign the document M and $\mathcal{V}$ verifies if the signature is correct.

Sign Algorithm:

1. $\mathcal{P}$ executes $P_1(x, w)$ to obtain the first message m_1 and a private state st_1.
2. $\mathcal{P}$ compute a random challenge by hashing the public statement, the document, and the first message, that is $c_1 = h(x, m_1, M)$.
3. $\mathcal{P}$ executes $P_2(st_1, c_1)$ to compute a second message m_2 and a private state st_2.
4. $\mathcal{P}$ compute a random challenge by hashing x, m_2 and the document M, that is $c_2 = h(x, M, m_1, c_1, m_2)$.
5. $\mathcal{P}$ executes $P_3(st_2, c_2)$ to compute the final message m_3. The final signature σ consists of the tuple (m_1, m_2, m_3).

Verify Algorithm:

1. $\mathcal{V}$ receives the public statement x, the document M, and the signature $\sigma = (m_1, m_2, m_3)$.
2. $\mathcal{V}$ recomputes the first challenge exactly as the prover did: $c_1 = h(x, M, m_1)$.
3. $\mathcal{V}$ recomputes the second challenge: $c_2 = h(x, M, m_1, c_1, m_2)$.
4. $\mathcal{V}$ executes the deterministic verification algorithm $V(x, m_1, c_1, m_2, c_2, m_3)$. If it outputs 1, the signature is accepted; otherwise, it is rejected.

2 Coding Theory

Coding theory is the branch of mathematics studying algebraic structures that allow detection and correction of errors occurring whenever a message passes through a noisy channel. The messages are viewed as vectors with coordinates over a finite field $\mathbb{F}_q$ where q is a power of a prime number. Moreover, instead of using all possible vectors in $\mathbb{F}_q^n$ as valid messages, we restrict ourselves to a specific subset $C \subseteq \mathbb{F}_q^n$, which we call a **code**. The elements of C are the only permitted messages, called **codewords**. The noisy channel introduces an error $e \in \mathbb{F}^n$ during the transmission and, consequently, the receiver gets a vector $y = c + e \in \mathbb{F}_q^n$ where $c \in C$ is the sent message. The mathematical structure of C is what allows us to efficiently detect and correct these errors.

Code-based cryptography uses hard problems coming from coding theory in order to construct secure cryptographic schemes. In particular, RYDE relies on a variant of the Rank Syndrome Decoding (RSD) problem called RSD_s.

In this section we will first introduce the fundamental notions regarding linear codes. Subsequently, we will discuss rank-metric codes. Finally, we will focus on the analysis of the connection between the RSD problem and the RSD_s variant.

2.1 Linear codes

We say that $C \subset \mathbb{F}_q^n$ is a **linear code** of length n over $\mathbb{F}_q$ if it is a $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^n$.

Once the algebraic structure of the code is established, we need a mathematical tool to quantify how far the received message is from the original codeword. To this purpose we equip the space with a metric.

Let $(X, +)$ be an abelian group. A **metric** (or distance) on X is a function $d : X \times X \to \mathbb{R}_+$ such that for all $x, y, z \in X$, the following axioms hold:

1. Definiteness: $d(x,y) = 0 \iff x = y$;

2. Symmetry: $d(x,y) = d(y,x)$;

3. Triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$.

Given a metric d on the group X, we define the **weight** of $x \in X$ as $w(x) \coloneqq d(x,0)$, where 0 is the identity element of the group.

The most classical metric used in coding theory is the Hamming metric which counts the number of coordinate errors introduced by the channel. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$. The **Hamming distance** between x and y is defined as the number of coordinates in which the two vectors differ:

$$d_H(x,y) = |\{i \in \{1,\ldots,n\} \mid x_i \neq y_i\}|.$$

The function d_H satisfies all the axioms of a metric, indeed the first two properties follows directly; the triangle inequality holds because if two vectors x and z differ in the i-th coordinate, any third vector y must differ from at least one of them in that same coordinate.

The error correcting capability of a linear code is influenced by how dispersed its codeword are. This brings us to the concept of minimal distance.

Given C a linear code of length n over $\mathbb{F}_q$ and a metric over $\mathbb{F}_q$ we will say that d is the **minimal distance** of C if

$$d = \min\{d(x,y) \,|\, (x,y) \in C^2 \text{ with } x \neq y\}.$$

Notation: We will say that C is a $[n,k,d]$-linear code if C has length n, minimal distance d and dimension k over $\mathbb{F}_q$.
If the distance is not specified, we would say that C is a $[n,k]$-linear code.

2.1 Remark (Translation invariance)
If a metric d is translation invariant, that is $d(x,y) = w(x-y)$ for all x, y,

then the minimal distance coincides with the minimal weight among all non-zero codewords:

$$d = \min\{d(x,y) \,|\, (x,y) \in C^2 \text{ with } x \neq y\} = \min\{w(z) \,|\, z \in C \setminus \{0\}\}.$$

The Hamming metric trivially satisfies this condition.

The minimal distance of a code determines its error-detecting and error-correcting capabilities.

Let C be a linear code with minimal distance d. The **error-correcting capability** of $\mathcal{C}$ is the maximum number ϵ of errors that the code can guarantee to correct in any received word. It is given by:

$$\epsilon = \left\lfloor \frac{d-1}{2} \right\rfloor$$

If we surround each codeword with a ball of radius t, with respect to the metric d, all these ball will be disjoint. If the distance between the original codeword and the received word is less or equal than ϵ, then the received word falls uniquely inside the ball of the original codeword, allowing a correct decoding.

Let C be an $[n,k]$-linear code over $\mathbb{F}_q$. Since C is a vector space of dimension k over $\mathbb{F}_q$, it can be completely described by choosing a basis of k linearly independent vectors. By collecting these vectors as rows, we construct a $k \times n$ matrix G, called a **generator matrix** for C.

The matrix G provides an encoding mechanism, mapping any k-dimensional message $m \in \mathbb{F}_q^k$ to its corresponding n-dimensional codeword $c \in C$. Thus, the entire code is given by the image of G:

$$C = \{mG \mid m \in \mathbb{F}_q^k\}.$$

A generator matrix G is **in standard form** if $G = [I_k \mid P]$ where I_k is the identity matrix of size $k \times k$ and P is a $k \times (n-k)$ matrix with entries in $\mathbb{F}_q$.

A code that admits a generator matrix in standard form is called a **systematic code**.

The dual notion to the generator matrix is the parity-check matrix, which allows us to verify if a received word belongs to the code.

A matrix $H \in \mathrm{Mat}_{(n-k)\times n}(\mathbb{F}_q)$ is called a **parity-check matrix** for C if, for any vector $x \in \mathbb{F}_q^n$, the following condition holds:

$$x \in C \iff Hx^\top = 0.$$

The following proposition describes the connection between generator and parity-check matrices.

2.2 Proposition
Let $C \subseteq \mathbb{F}_q^n$ be an $[n,k]$ code.

1. *H is a parity-check matrix for C if and only if C is the kernel of the linear map:*

$$\begin{aligned} L_H : \mathbb{F}_q^n &\to \mathbb{F}_q^{n-k} \\ x &\mapsto Hx^\top \end{aligned}$$

2. *If G and H are a generator matrix and a parity-check matrix for C, respectively, then:*
$$H^\top G = 0 \in \mathrm{Mat}_{n\times k}(\mathbb{F}_q)$$

3. *Let G be a generator matrix for C. If $H \in \mathrm{Mat}_{n-k,n}(\mathbb{F}_q)$ satisfies $H^\top G = 0$ and $\mathrm{rank}(H) = n-k$, then H is a parity-check matrix for C.*

4. *If C is a systematic code with generator matrix $G = [I_k \mid P]$, then $H = [-P^\top \mid I_{n-k}]$ is a parity-check matrix for C.*

Proof. 1. It follows by definition.

2. It follows from the fact that the rows of G are vectors in C.

3. The condition $H^\top G = 0$ implies that $C \subseteq \ker(L_H)$. Moreover, by the Rank-Nullity Theorem, $\dim(\ker L_H) = k$. Consequently, $C = \ker(L_H)$.

4. Let H and G be in the form described. We compute:

$$\begin{aligned} H^\top G &= [-P^\top \mid I_{n-k}] \begin{bmatrix} I_k \\ P^\top \end{bmatrix} \\ &= (-P^\top)(I_k) + (I_{n-k})(P^\top) \\ &= -P^\top + P^\top \\ &= 0 \in \mathrm{Mat}_{n-k,k}(\mathbb{F}_q) \end{aligned}$$

   We rely on point 3 to conclude.

□

2.3 Example

Let $q = 2$, $n = 4$ and $k = 2$. We define the generator matrix in standard form

$$G = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right) \in \mathrm{Mat}_{2,4}(\mathbb{F}_2)$$

Since G is in standard form, the code C is systematic.

The linear code C is the subspace of $\mathbb{F}_2^4$ generated by the rows of G, i.e.

$$C = \langle (1,0,1,1), (0,1,1,0) \rangle_{\mathbb{F}_2} = \{(0,0,0,0), (1,0,1,1), (0,1,1,0), (1,1,0,1)\}$$

Since $w(1,0,1,1) = 3$, $w(0,1,1,0) = 2$ and $w(1,1,0,1) = 3$ the minimal distance of the code C is $d = 2$. Consequently, C is a $[4,2,2]$-linear code $\mathbb{F}_2$.

By applying the Proposition 2.2 the following matrix

$$H = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right) \in \mathrm{Mat}_{2,4}(\mathbb{F}_2)$$

is a parity-check matrix for C.

Let C a $[n,k]$-linear with parity check matrix H. Given a received word $y \in \mathbb{F}_q^n$, we define the **syndrome** of y as the vector

$$s = Hy^\top \in \mathbb{F}_q^{n-k}.$$

2.4 Remark

By definition, the parity-check matrix allow us to verify if a vector is in the code, since $Hy^\top = 0 \iff y \in C$.

Moreover, if $c \in C$ *is transmitted and we receive the word* $y = c + e$*, then we can compute the syndrome of* y *as follows:*

$$Hy^\top = H(c^\top + e^\top) = He^\top.$$

This shows that the syndrome does not depend from the codeword that was sent. Consequently, we can say that the syndrome measures the error of the transmitted message.

2.2 Rank metric codes

Let m be a positive integer, $x = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ and let $\mathcal{B} = \{b_1, \ldots, b_m\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then, there exists a matrix $M_{\mathcal{B}}(x) \in \mathrm{Mat}_{n\times m}(\mathbb{F}_q)$ such that:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = M_{\mathcal{B}}(x) \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Note that while the matrix $M_{\mathcal{B}}(x)$ depends on the choice of the basis $\mathcal{B}$, its rank, denoted by $\mathrm{rk}(M_{\mathcal{B}}(x))$, is invariant under a change of basis. Therefore, we define the **rank weight** of x as $w_{\mathrm{rk}}(x) \coloneqq \mathrm{rk}(M_{\mathcal{B}}(x))$ with respect to any basis $\mathcal{B}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Equivalently, we could define the rank weight of x as the dimension of the $\mathbb{F}_q$-vector space generated by its coordinates, that is $w_{\mathrm{rk}}(x) = \dim\langle x_1, \ldots, x_n\rangle_{\mathbb{F}_q}$.

Consequently, we define the **rank metric** as the function:

$$\begin{aligned} d_{\mathrm{rk}} \colon \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n &\to \mathbb{R}_{\geq 0} \\ (x, y) &\mapsto w_{\mathrm{rk}}(x - y) \end{aligned}$$

In this setting, a **vector rank-metric** code of length n is a $\mathbb{F}_{q^m}$-linear subspace of $\mathbb{F}_{q^m}^n$.

2.5 Remark

The terminology vector rank-metric code derives from the fact that there exists a notion of (matrix) rank-metric code that is defined as a $\mathbb{F}_q$-linear subspace of $\mathrm{Mat}_{n\times m}(\mathbb{F}_q)$. A detailed investigation of rank-metric codes can be found in [Gor19].

2.6 Example

Consider the field $\mathbb{F}_{2^3} \cong \mathbb{F}_2[x]/(x^3 + x + 1)$ and denote $\alpha \coloneqq \bar{x}$. Let $C = \langle(\alpha, \alpha^2 + 1)\rangle_{\mathbb{F}_{2^3}}$ be a vector rank-metric code of length 2 over $\mathbb{F}_{2^3}$. Using the basis $\Gamma = \{1, \alpha, \alpha^2\}$ of $\mathbb{F}_{2^3}$ over $\mathbb{F}_2$, we can represent any vector in $\mathbb{F}_{2^3}^2$ as a 3×2 matrix over $\mathbb{F}_2$. For instance, the generator $v = (\alpha, \alpha^2 + 1)$ expands to the matrix:

$$\Gamma(v) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \in \mathrm{Mat}_{3\times 2}(\mathbb{F}_2)$$

Moreover, the code C can also be viewed as a 3-dimensional subspace over $\mathbb{F}_2$ generated by $\{v, \alpha v, \alpha^2 v\}$. Consequently, we can consider the associated matrix rank-metric code w.r.t. Γ, that is:

$$\Gamma(C) = \left\langle \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{pmatrix} \right\rangle_{\mathbb{F}_2}.$$

2.7 Remark (Rank factorization)

We observe that a vector $x \in \mathbb{F}_{q^m}^n$ has rank r if and only if $\mathrm{rk}(M_{\mathcal{B}}(x)) = r$

for any fixed basis $\mathcal{B}$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Furthermore, this holds if and only if the matrix $M_{\mathcal{B}}(x)$ admits a rank factorization $M_{\mathcal{B}}(x) = A \cdot G$, where $G \in \mathrm{Mat}_{r\times n}(\mathbb{F}_q)$ and $A \in \mathrm{Mat}_{m\times r}(\mathbb{F}_q)$ both have full-rank r. We can explicitly construct a factorization as follows. Since $M_{\mathcal{B}}(x)$ has rank r, it contains exactly r linearly independent rows. Without loss of generality, assume these are the first r rows and let them form the matrix G. Since the rows of G form a basis for the row space of $M_{\mathcal{B}}(x)$, any j-th row of $M_{\mathcal{B}}(x)$ can be expressed as a linear combination of the rows of G. That is, for each $j \in \{1, \ldots, m\}$, there exist coefficients $a_{j,1}, \ldots, a_{j,r} \in \mathbb{F}_q$ such that the j-th row of $M_{\mathcal{B}}(x)$ is given by:

$$a_{j,1}G_1 + \cdots + a_{j,r}G_r$$

where G_i denotes the i-th row of G. The matrix A is then simply defined as the $m \times r$ matrix whose (j, i)-th entry is $a_{j,i}$.

Let n, r be two positive integers with $n \geq r$, we define the **Gaussian Binomial coefficient** of n over r as

$$\binom{n}{r}_q = \frac{(q^n-1)(q^{n-1}-1)\cdots(q^{n-r+1}-1)}{(q-1)(q^2-1)\cdots(q^r-1)}$$

2.8 Remark

The Gaussian Binomial coefficient of n over r measures the number of $\mathbb{F}_q$-linear subspaces of $\mathbb{F}_q^n$ with dimension r. Indeed, the numerator of $\binom{n}{r}_q$ represents the number of possible ordered sets of r $\mathbb{F}_q$-linearly independent vectors in $\mathbb{F}_q^n$, since:

- *removing the null vector, there are $q^n - 1$ possibilities for the first vector v_1;*
- *removing the one-dimensional linear space $\langle v_1 \rangle_{\mathbb{F}_q}$, there are $q^n - q$ possibilities for the second vector v_2;*
- *repeating this reasoning in total r times we obtain the product*

$$(q^n-1)(q^n-q)\cdots(q^n-q^{r-1})$$

However, this product counts the number of possible vector basis, not the subspaces themselves. Each r-dimensional space has multiple basis. The number of different ordered bases for a single r-dimensional space is obtained by applying the same reasoning in dimension r, yielding $(q^r - 1)(q^r - q) \cdots (q^r - q^{r-1})$.

Dividing the number of linearly independent r vectors by the number of bases per subspace we obtain the Gaussian Binomial coefficient.

2.9 Proposition

Let S^r_{rk} be the sphere with respect to the rank metric of radius r in $\mathbb{F}^n_{q^m}$, that is

$$S^r_{\mathrm{rk}} \coloneqq \{a \in \mathbb{F}^n_{q^m} \mid w_{\mathrm{rk}}(a) = r\}.$$

The cardinality of S^r_{rk} is

$$S^r_{\mathrm{rk}} = \binom{n}{r}_q \cdot \prod_{j=0}^{r-1} (q^m - q^j)$$

Proof. By representing the vectors in $\mathbb{F}^n_{q^m}$ as matrices in $\mathbb{F}^{m \times n}_q$ the problem turns into counting the number of matrices with rank r.

As observed in Remark 2.7, any such matrix can be factored as a product $A \cdot G$ of two full-rank matrices $G \in \mathrm{Mat}_{r \times n}(\mathbb{F}_q)$ and $A \in \mathrm{Mat}_{m \times r}(\mathbb{F}_q)$. The number of full-rank matrices G, up to multiplication by an invertible matrix $U \in \mathrm{GL}_r(\mathbb{F}_q)$, corresponds to the number of r-dimensional subspaces of $\mathbb{F}^n_q$, and is given by the Gaussian Binomial coefficient $\binom{n}{r}_q$.

Once the matrix G is selected, the matrix A determines the linear combinations of the rows of G. In order for the resulting matrix to have rank r, the matrix $A \in \mathrm{Mat}_{m \times r}(\mathbb{F}_q)$ must have full column rank. Counting the valid matrices A corresponds to counting the number of ordered sets of r $\mathbb{F}_q$-linearly independent vectors in $\mathbb{F}^m_q$, which is exactly $\prod_{j=0}^{r-1}(q^m - q^j)$.

By multiplying the number of choices for the r-dimensional subspaces by the choices for the coefficients, we obtain the exact cardinality. □

2.3 The Decoding problems

In this subsection we describe the underlying mathematical problems formalizing the error-correction process.

Let q be a power of a prime number, let m, n, k, r be positive integers and let the space $\mathbb{F}_{q^m}$ be equipped with a metric d.

The most natural formulation of the problem is to recover the original codeword from the received message. This is formalized as follows:

2.10 Definition (The Decoding Problem)
Let $G \in \mathrm{Mat}_{k\times n}(\mathbb{F}_{q^m})$ be a generator matrix chosen uniformly at random and let $x \in \mathbb{F}_{q^m}^k$ be a uniformly random message. Suppose we are given a received word $z = xG + e \in \mathbb{F}_{q^m}^n$, where the error vector e is sampled uniformly at random from the sphere of radius r.
Problem Statement: *Given the public pair (G, z), find the secret error vector e.*

There exist also a dual version of the problem described. By multiplying the received word $z = xG + e$ by the parity-check matrix H, the codeword xG is annihilated, leaving only the syndrome $y^\top = Hz^\top = He^\top \in \mathbb{F}_{q^m}^{n-k}$. This yields the following problem:

2.11 Definition (The Syndrome Decoding Problem)
Let $H \in \mathrm{Mat}_{(n-k)\times n}(\mathbb{F}_{q^m})$ be a parity-check matrix chosen uniformly at random and let $e \in \mathbb{F}_{q^m}^n$ be a secret error vector sampled uniformly from the sphere of radius r.
Problem Statement: *Given the public pair (H, y), where $y^\top = He^\top \in \mathbb{F}_{q^m}^{n-k}$, find the secret vector e.*

When the underlying metric d is instantiated as the rank metric over the extension field $\mathbb{F}_{q^m}$, this problem takes the name of **Rank Syndrome Decoding (RSD)**.

However, RYDE relies on a variant of this problem, denoted as RSD_s. This variant restricts the search of the error vector by forcing specific conditions on its coordinates.

2.12 Definition (The RSD_s Restriction)
An instance of the RSD_s problem requires finding an error vector $e \in \mathbb{F}_{q^m}^n$ of rank weight $w_{\mathrm{rk}}(e) = r$ that satisfies two additional conditions:

- *The first coordinate is fixed to $e_1 = 1 \in \mathbb{F}_{q^m}$.*
- *The first r coordinates $\{e_1, \ldots, e_r\}$ are $\mathbb{F}_q$-linearly independent (or equivalently generate the space $\langle e_1, \ldots, e_n \rangle_{\mathbb{F}_q}$).*

This restriction allow us to factorize the error vector e into two smaller components. Following the rank decomposition construction in Remark 2.7, e can be uniquely decomposed as the product $e = sC$, where:

$$s = (1 \parallel s') \in \mathbb{F}_{q^m}^r \quad \text{and} \quad C = [I_r \parallel C'] \in \mathrm{Mat}_{r \times n}(\mathbb{F}_q)$$

Here, $s = (e_1, \ldots, e_r)$, while C contains the coordinates of e with respect to this basis. The uniqueness follows from the fact that, for any other valid decomposition $e = \tilde{s}\tilde{C}$, the structure of the matrix $\tilde{C} = [I_r \parallel \tilde{C}']$ forces the first r coordinates of the product $\tilde{s}\tilde{C}$ to be exactly $\tilde{s}$. Once $\tilde{s}$ is uniquely determined, the entries of $\tilde{C}$ must be constructed as the coefficients of the linear combination, yielding $C = \tilde{C}$.

Consequently, the goal of RDS_s can be stated as follows:

The New RSD_s Goal: Find a pair $(s', C') \in \mathbb{F}_{q^m}^{r-1} \times \mathrm{Mat}_{r \times (n-r)}(\mathbb{F}_q)$ satisfying:

$$HC^\top s^\top = y^\top$$

and having the r coordinates of s $\mathbb{F}_q$-linearly independent.

A crucial property of the RSD_s variant is that its overall security does not differ from the overall security of RSD problem. More specifically, the existence of an efficient attack for solving RSD_s yields an efficient attack for

a generic RSD instance with a non-negligible success probability. We now present the formal result, originally introduced by Bidoux et al. in [Bid+25]. The subsequent remarks will provide a mathematical justification for the reduction probability and its underlying assumption.

2.13 Proposition
Let m, k, n, r be positive integers with $n > k$. Assume that the code C associated with the given instance contains no codewords of rank weight r. If there exists a probabilistic polynomial-time algorithm A_s that solves RSD$_s$ instances with probability of success P_s, then there exists an algorithm A that solves RSD instances in polynomial time with a success probability:

$$P \geq \left(\prod_{i=0}^{r-1} \frac{q^n - q^{n-r+i}}{q^n - q^i} \right) P_s.$$

Proof. Let (H, y) be an RSD instance, where $H \in \mathrm{Mat}_{(n-k)\times n}(\mathbb{F}_{q^m})$ and $y \in \mathbb{F}_{q^m}^{n-k}$. We want to find a vector $x \in \mathbb{F}_{q^m}^n$ such that $w_{\mathrm{rk}}(x) = r$ and $y^\top = Hx^\top$.

Sample a matrix $U \in \mathrm{GL}_{n\times n}(\mathbb{F}_q)$ and define $H' \coloneqq H \cdot U^\top$. Find z such that $y = H'z^\top$ and define $\hat{H}$ as the parity check matrix of $C + \langle z \rangle$, we obtain that $(\hat{H}, 0)$ is an instance of RSD$_s$ with a probability greater or equal $\epsilon \coloneqq \prod_{i=0}^{r-1}, \frac{q^n - q^{n-r+i}}{q^n - q^i}$. Indeed, as detailed at the end of the proof, the probability that the first r coordinates of $x' \coloneqq xU^{-1}$ are $\mathbb{F}_q$-linearly independent is ϵ. Define now $c \coloneqq x' - z$. It satisfies:

$$H'c^\top = H'(x')^\top - H'z^\top = HU^\top(U^{-\top}x^\top) - y^\top = Hx^\top - y^\top = y^\top - y^\top = 0.$$

Consequently, c is a codeword of C. We define $x'' \coloneqq (x'_1)^{-1}x'$, hence $x'' \in C + \langle z \rangle$. The vector constructed x'' satisfies all the conditions in RSD$_s$.

The algorithm A_s on $(\hat{H}, 0)$ outputs a vector $\hat{x} \in \mathbb{F}_{q^m}^n$ such that $w_{\mathrm{rk}}(\hat{x}) = r$ and $\hat{H}\hat{x}^\top = 0$. Thus, $\hat{x} \in C + \langle z \rangle$ and $H'\hat{x}^\top = \alpha H'z^\top = \alpha y^\top$ where $\alpha \in \mathbb{F}_{q^m}$.

If we have $\alpha = 0$, then $\hat{x} \in C$ and this contradicts the assumption. We

assume that $\alpha \neq 0$ and define $\tilde{x} \coloneqq \alpha^{-1}\hat{x}U$, therefore:

$$w_{\mathrm{rk}}(\tilde{x}) = w_{\mathrm{rk}}(\alpha^{-1}\hat{x}U) = w_{\mathrm{rk}}(\hat{x}) = r,$$
$$H\tilde{x}^\top = H(\alpha^{-1}\hat{x}U)^\top = \alpha^{-1}HU^\top\hat{x}^\top = \alpha^{-1}H'\hat{x}^\top = \alpha^{-1}(\alpha y^\top) = y^\top.$$

This proves that $\tilde{x}$ has the desired rank weight r and yields the correct syndrome $y^\top$, thus successfully solving the original RSD instance (H, y).

Here we justify the probability that the first r coordinates of $x' = xU^{-1}$ are $\mathbb{F}_q$-linearly independent. Let us denote the kernel of multiplication by the vector x as $\ker(x) \coloneqq \{v \in \mathbb{F}_q^n \mid xv^\top = 0\}$. Since x has rank r, this kernel forms an $\mathbb{F}_q$-linear subspace of dimension $n - r$ within $\mathbb{F}_q^n$. Let us denote as u_i the columns of U^{-1}. To obtain the goal U^{-1} is restrained as follows:

1. $u_1 \notin \ker(x)$;
2. $u_2 \notin (\ker(x) + \langle u_1 \rangle_{\mathbb{F}_q})$;
3. $u_i \notin (\ker(x) + \langle u_1, \ldots, u_{i-1} \rangle_{\mathbb{F}_q})$ for any $i \leq r$.

For u_1, we must exclude the q^{n-r} elements of the kernel, leaving $q^n - q^{n-r}$ available choices. For u_2, we exclude the q^{n-r+1} elements of the newly spanned subspace, leaving $q^n - q^{n-r+1}$ choices. By extending this logic, the total number of ways to pick the first r columns is given by the product:

$$\prod_{i=0}^{r-1}(q^n - q^{n-r+i})$$

For the remaining $n - r$ columns, the only requirement is that U^{-1} remains an invertible matrix. The number of valid choices for these final columns is:

$$\prod_{j=r}^{n-1}(q^n - q^j)$$

By multiplying these two products together, we find the total number of successful matrices U^{-1}, which directly allows us to determine the success

probability

$$\epsilon = \frac{\prod_{i=0}^{r-1}(q^n - q^{n-r+i})\prod_{j=r}^{n-1}(q^n - q^j)}{|\mathrm{GL}_n(\mathbb{F}_q)|} = \prod_{i=0}^{r-1}\frac{q^n - q^{n-r+i}}{q^n - q^i}.$$

□

2.14 Example

Let the parameters be $q = 2, m = 3, n = 4, k = 2, r = 2$. *We work in the field* $\mathbb{F}_{2^3}$, *viewed as* $\mathbb{F}_2[x]/(x^3 + x + 1)$.

The input instance is given by the parity-check matrix H *and the syndrome vector* y:

$$H = \begin{bmatrix} 1 & x+1 & 0 & x^2 \\ x^2+1 & 1 & x & 0 \end{bmatrix}, \quad y = \begin{bmatrix} x & x+1 \end{bmatrix}.$$

We apply the algorithm described in the proof step by step:

1. ***Sample an invertible matrix*** $U \in \mathbb{F}_q^{n\times n}$.
   We use the matrix:
   $$U = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$
   Notice that $U^{-1} = U$.

2. ***Compute*** $H' \in \mathbb{F}_{q^m}^{n\times(n-k)}$ ***as*** $HU^\top$.
   We compute the new parity-check matrix:
   $$H' = HU^\top = \begin{bmatrix} 1 & x+1 & 0 & x^2 \\ x^2+1 & 1 & x & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$
   $$= \begin{bmatrix} 1+(x+1) & x+1 & 0 & x^2 \\ (x^2+1)+1 & 1 & x & 0 \end{bmatrix} = \begin{bmatrix} x & x+1 & 0 & x^2 \\ x^2 & 1 & x & 0 \end{bmatrix}.$$

3. ***Find*** z ***such that*** $y^\top = H'z^\top$***.***

We look for a vector $z = (z_1, z_2, z_3, z_4)$ *that is a solution to the system*

$$\begin{bmatrix} x & x+1 & 0 & x^2 \\ x^2 & 1 & x & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} x \\ x+1 \end{bmatrix}.$$

By arbitrarily fixing $z_3 = 0$ *and* $z_4 = 0$*, we obtain the system:*

$$\begin{cases} z_1 x + z_2(x+1) = x, \\ z_1 x^2 + z_2 = x + 1. \end{cases}$$

From the second equation we obtain $z_2 = z_1 x^2 + x + 1$*. Substituting this into the first equation:*

$$\begin{aligned} z_1 x + (z_1 x^2 + x + 1)(x+1) &= x \\ z_1 x + z_1(x^2 + x + 1) + x^2 + 1 &= x \\ z_1(x^2+1) &= x^2 + x + 1. \end{aligned}$$

Multiplying by the inverse of (x^2+1)*, which is* x*, since* $x(x^2+1) = x^3 + x = 1$*, we obtain:*

$$z_1 = x(x^2 + x + 1) = x^3 + x^2 + x = (x+1) + x^2 + x = x^2 + 1$$

Substituting z_1*, we obtain* z_2*:*

$$z_2 = (x^2+1)x^2 + x + 1 = x^4 + x^2 + x + 1 = (x^2 + x) + x^2 + x + 1 = 1.$$

We have found the solution: $z = \begin{bmatrix} x^2+1 & 1 & 0 & 0 \end{bmatrix}$.

4. ***Build*** $\hat{H} \in \mathbb{F}_{q^m}^{(n-k-1)\times n}$ ***as the parity check matrix of*** $\mathcal{C} + \langle z \rangle$***.***

We look for a linear combination of the rows of H' *(let them be* h'_1 *and*

h'_2) *that evaluates to zero on the vector* z. *We know that* $h'_1 z^\top = x$ *and* $h'_2 z^\top = x+1$. *By cross-multiplying we obtain the vectors:*

$$(x+1)h'_1 = \begin{bmatrix}(x+1)x & (x+1)^2 & 0 & (x+1)x^2\end{bmatrix} = \begin{bmatrix}x^2+x & x^2+1 & 0 & x^2+x+1\end{bmatrix},$$
$$xh'_2 = \begin{bmatrix}x^3 & x & x^2 & 0\end{bmatrix} = \begin{bmatrix}x+1 & x & x^2 & 0\end{bmatrix}.$$

Adding them, we obtain the matrix of the code $\mathcal{C} + \langle z \rangle$*:*

$$\hat{H} = \begin{bmatrix}x^2+1 & x^2+x+1 & x^2 & x^2+x+1\end{bmatrix}.$$

5. ***Run*** $\mathcal{A}_s$ ***on input*** $(\hat{H}, 0)$ ***to get*** $\hat{x}$**.**
   The algorithm $\mathcal{A}_s$ *searches for a word of rank weight* $r = 2$ *in the code defined by* $\hat{H}$, *having a* 1 *in the first coordinate:*

$$\hat{x} = \begin{bmatrix}1 & x & 0 & 0\end{bmatrix}.$$

6. ***If*** $\mathcal{A}_s$ ***does not return*** $\hat{x}$**, repeat from 1.**
   $\mathcal{A}_s$ *did not fail, we proceed.*

7. ***Compute*** $\alpha \in \mathbb{F}_{q^m}$ ***such that*** $H'\hat{x}^\top = \alpha y^\top$**.**
   We compute the product $H'\hat{x}^\top = \begin{bmatrix}x^2 \\ x^2+x\end{bmatrix}$. *Thus* $\alpha = x$.

8. ***Compute*** $\tilde{x}$ ***as*** $\alpha^{-1} \cdot \hat{x} \cdot U$**.**
   Since $\alpha^{-1} = x^2+1$, *we compute the matrix multiplication:*

$$\tilde{x} = (x^2+1)\cdot\begin{bmatrix}1 & x & 0 & 0\end{bmatrix}\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} = (x^2+1)\begin{bmatrix}1 & x+1 & 0 & 0\end{bmatrix} = \begin{bmatrix}x^2+1 & x^2 & 0 & 0\end{bmatrix}.$$

9. ***Return*** $\tilde{x}$**.** *Finally, we check that the algorithm has successfully com-*

puted the solution:

$$\begin{bmatrix} x^2+1 & x^2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & x^2+1 \\ x+1 & 1 \\ 0 & x \\ x^2 & 0 \end{bmatrix} = \begin{bmatrix} x & x+1 \end{bmatrix}.$$

2.15 Remark

Let C be a random $[n,k]$ linear code over $\mathbb{F}_{q^m}$, and let r be a positive integer such that $r < m, n$. Then, the probability that there exists a codeword of rank weight r in C is bounded by $4 \cdot q^{r(m+n-r)-m(n-k)}$. Indeed, we have

$$\mathbb{P}\left(\bigcup_{v \in S^r_{\mathrm{rk}}} \{v \in C\} \right) \leq \sum_{v \in S^r_{\mathrm{rk}}} \mathbb{P}(v \in C).$$

Moreover, fix $v \in S^r_{\mathrm{rk}}$. As the parity-check matrix $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ varies uniformly at random, the syndrome $s_H = Hv^\top$ is uniformly distributed in $\mathbb{F}_{q^m}^{n-k}$. To see this, consider the $\mathbb{F}_{q^m}$-linear map

$$\begin{aligned} L^v \colon \mathbb{F}_{q^m}^{(n-k)\times n} &\to \mathbb{F}_{q^m}^{n-k} \\ H &\mapsto Hv^\top. \end{aligned}$$

This map is surjective: for any given syndrome $s \in \mathbb{F}_{q^m}^{n-k}$, let H^s be the matrix with all columns equal to zero except for the j-th column, which is set to $\frac{1}{v_j} s$. Here, v_j is a non-zero coordinate of v (which must exist since $v \in S^r_{\mathrm{rk}}$ has strictly positive rank weight). Consequently, the preimages $(L^v)^{-1}(s) = H^s + \ker L^v$ all have exactly the same cardinality for any s. Thus,

$$\mathbb{P}(v \in C) = \mathbb{P}(Hv^\top = 0) = \frac{1}{q^{m(n-k)}}.$$

Therefore, it holds that:

$$\mathbb{P}\left(\bigcup_{v \in S^r_{\mathrm{rk}}} \{v \in C\} \right) \leq |S^r_{\mathrm{rk}}| \cdot q^{-m(n-k)} = \binom{n}{r}_q \cdot \prod_{j=0}^{r-1} (q^m - q^j) \cdot q^{-m(n-k)}.$$

Observe that $\prod_{j=0}^{r-1}(q^m - q^j) < q^{rm}$. *Furthermore, for* q^n *much greater than* q^r, *we can upper bound the Gaussian binomial coefficient as:*

$$\binom{n}{r}_q = \frac{\prod_{i=1}^{r}(q^{n-r+i}-1)}{\prod_{i=1}^{r}(q^i-1)} \leq \frac{\prod_{i=1}^{r} q^{n-r+i}}{\prod_{i=1}^{r} q^i \cdot \prod_{i=1}^{r}(1-q^{-i})} = \frac{q^{r(n-r)} \cdot q^{\frac{r(r+1)}{2}}}{q^{\frac{r(r+1)}{2}} \prod_{i=1}^{r}(1-q^{-i})}$$
$$\leq q^{r(n-r)} \prod_{i=1}^{r} \frac{1}{(1-q^{-i})}.$$

The product on the right is bounded from above by the infinite product $\prod_{i=1}^{\infty} \frac{1}{(1-q^{-i})}$. *We observe that for each* $i \geq 1$, *the term* $\frac{1}{(1-q^{-i})}$ *is a strictly decreasing function of* q. *Hence, to find an upper bound, it is sufficient to consider the minimum valid parameter* $q = 2$. *As discussed in [Fin03, p. 318], this infinite product converges to a constant strictly less than* 4.

Substituting these approximations, we conclude that:

$$\mathbb{P}\left(\bigcup_{v \in S^r_{\mathrm{rk}}} \{v \in C\}\right) \leq 4 \cdot q^{r(n-r)} q^{rm} q^{-m(n-k)} = 4q^{r(m+n-r)-m(n-k)}.$$

As a concrete example, in the RYDE-1 parameter set (targeting NIST security level 1), the parameters are $q = 2$, $m = 53$, $n = 53$, $k = 45$, *and* $r = 4$. *The probability of finding a codework of weight rank* 4 *is bounded by* $4 \cdot 2^{4(53+53-4)-53(53-45)} = 4 \cdot 2^{408-424} = 2^{-14}$.

2.16 Remark

For the RYDE-1 parameter set the value $\prod_{i=0}^{r-1} \frac{q^n - q^{n-r+i}}{q^n - q^i}$ *in Proposition 2.13 is around* $0,308$.

The proof of Remark 2.15 can be adapted to show that, for the RYDE parameter sets, the probability of multiple solutions is bounded by the same quantity.

2.17 Proposition (Uniqueness of RSD solution)

Given a random RSD instance (H, y) *the probability of the existence of another solution of rank* r *is bounded by* $4 \cdot q^{r(m+n-r)-m(n-k)}$.

Proof. Let e be the secret error vector of rank r satisfying $eH^\top = y$. The set of possible solutions $\mathcal{S}$ can be described as follows

$$\mathcal{S} = e + \ker H^\top.$$

Noting that $\ker H^\top = C$ is the linear code defined by H, any other potential solution $e' \in \mathcal{S}$ must be of the form $e' = e - c$ for some non-zero codeword $c \in C \setminus \{0\}$.

Then the probability of existence of another solution to the RSD instance can be viewed as the probability that there exists a non-zero codeword contained in the rank-metric sphere $S^r_{\mathrm{rk}}(e)$ of radius r centered at e. We can upper bound this probability as follows:

$$\mathbb{P}\left(\bigcup_{v \in S^r_{\mathrm{rk}}(e)\setminus\{0\}} \{v \in C\}\right) \leq \sum_{v \in S^r_{\mathrm{rk}}(e)\setminus\{0\}} \mathbb{P}(v \in C).$$

By the translation invariance of the rank metric, we note that $|S^r_{\mathrm{rk}}(e)| = |S^r_{\mathrm{rk}}|$. Proceeding exactly as in Remark 2.15, we conclude the proof. □

3 RYDE Protocol

In this chapter, we describe RYDE. RYDE is a digital signature scheme based on the RSD_s variant of the RSD problem. Indeed, its name derives directly from the Rank sYndrome DEcoding problem.

In December 2016 the National Institute of Standards and Technology (NIST) started a process in order to select and standardize public-key cryptographic algorithms that can resist to quantum attacks. After three rounds of analysis and evaluations, three digital signature schemes were selected, two of which rely on the computational hardness of problems involving lattices. To diversify the types of problems that signatures rely on, in 2022 NIST called for proposals for additional digital signatures.

RYDE, introduced by Bidoux et al. in [Ara+23], was submitted in response to this call. While it successfully advanced past the first round, it was ultimately not selected to proceed after the conclusion of Round 2 in May 2026.

RYDE was originally designed as a signature scheme based on the Multiparty Computation-in-the-Head (MPC-itH) paradigm. However, in the Round 2 Specification, the scheme transitioned to a Polynomial Interactive Oracle Proof (PIOP) using the Treshold Computation in the Head approach. This new paradigm, introduced by T. Feneuil and M. Rivain and described in [Fen24], allows a description of RYDE independent from the MPC-itH architecture. This thesis will exclusively treat this modern description.

RYDE is constructed by taking a zero-knowledge proof for a solution to an RSD_s instance and applying the Fiat-Shamir transform to convert it into a non-interactive digital signature.

3.1 Generalized RYDE proof system

In this subsection, we present the proof system underlying RYDE. We first introduce the protocol in a generalized setting: proving the knowledge of a

solution to a system of quadratic equations. In the next subsection we will show how RSD_s instances can be encoded in this framework.

Before detailing the protocol, we fix the necessary setup and parameters. Let n, m, μ, N be positive integers satisfying $N \leq q^\mu$. Let $\phi : \{1, \ldots, N\} \hookrightarrow \mathbb{F}_{q^\mu}$ be a public injective function, PRG be a pseudorandom generator and h an hash function.

Core Objective

The Prover $\mathcal{P}$ wants to convince the Verifier $\mathcal{V}$ that he knows a secret witness $w \in \mathbb{F}_q^n$ satisfying:

$$f_j(w) = 0 \quad \text{for every } 1 \leq j \leq m$$

where the public polynomials $f_j \in \mathbb{F}_q[x_1, \ldots, x_n]$ have a maximum total degree of 2.

3.1 Remark

While RYDE instances require quadratic systems, there exist variants of this protocol for polynomials restricted to any maximum degree d.

Protocol

Before detailing the individual steps, the protocol can be summarized as follows:

1. $\mathcal{P}$ constructs and commits to n random polynomials $P_j(X) \in \mathbb{F}_{q^\mu}$ having the witness components w_j as leading coefficients and random constant terms for $j = 1, \ldots, n$ alongside an additional random linear polynomial $P_0(X)$.

2. $\mathcal{V}$ samples random coefficients $\gamma_j \in \mathbb{F}_{q^\mu}$ for $j = 1, \ldots, m$.

3. $\mathcal{P}$ computes and sends to $\mathcal{V}$ the aggregated polynomial

$$Q(X) = P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j^{[h]}(X, P_1(X), \ldots, P_n(X));$$

4. $\mathcal{V}$ samples an evaluation point $r \in \mathrm{Im}(\phi)$ and sends it to $\mathcal{P}$;

5. Finally, $\mathcal{P}$ sends the evaluation of the polynomials from step 1 to $\mathcal{V}$, who checks that these evaluations are consistent with the revealed polynomial $Q(X)$.

Step 1: Prover's Commitment

Before executing the interactive protocol, $\mathcal{P}$ must commit to the witness w.

- $\mathcal{P}$ creates n random masking polynomials of degree 1:

$$P_j(X) = w_j \cdot X + (w_{\text{base}})_j \quad \text{for } j \in \{1, \ldots, n\}$$

  where $(w_{\text{base}})_j \in \mathbb{F}_{q^\mu}$ a uniformly random element.

- $\mathcal{P}$ also creates a random linear polynomial $P_0 \in \mathbb{F}_{q^\mu}[X]$.

- $\mathcal{P}$ commits to these polynomials.

How is the commitment done?

The **first phase of the commitment scheme** proceeds as follows:

1. $\mathcal{P}$ samples N random seeds $\text{seed}_1, \ldots, \text{seed}_N$ and computes their commitments, which corresponds to the list of hash of seeds, i.e. $\text{com}_i = h(\text{seed}_i)$ for $i \in \{1, \ldots, N\}$;

2. $\mathcal{P}$ generates the random vectors $w_{\text{rnd},i} = \text{PRG}(\text{seed}_i) \in \mathbb{F}_q^{n+1}$ and computes

$$w_{\text{acc}} = \sum_{i=1}^{N} w_{\text{rnd},i} \in \mathbb{F}_q^{n+1}$$

$$w_{\text{base}} = -\sum_{i=1}^{N} \phi(i) w_{\text{rnd},i} \in \mathbb{F}_{q^\mu}^{n+1}.$$

3. $\mathcal{P}$ constructs the polynomials

$$P_j(X) = w_j \cdot X + (w_{\text{base}})_j \quad \text{for } j \in \{1, \ldots, n\}$$

and $P_0(X) = (w_{\text{acc}})_0 \cdot X + (w_{\text{base}})_0$;

4. $\mathcal{P}$ defines $\overline{w} = ((w_{\text{acc}})_0, w)$, reflecting the fact that P_0 does not depend on w and compute $\text{aux} = \overline{w} - w_{\text{acc}}$;

5. finally, the global commitment is computed as the hash of all individual commitments and aux:

$$h_1 \coloneqq h(\text{com}_1, \ldots, \text{com}_N, \text{aux}).$$

In the **second phase of the scheme**, $\mathcal{P}$ reveals all seeds except one depending on a challenge index i^* chosen by the Verifier $\mathcal{V}$. Alongside the $N-1$ seeds, $\mathcal{P}$ also reveals aux and com_{i^*}. This allows $\mathcal{V}$ to verify that the commitment h_1 is consistent with the revealed elements without reconstructing the hidden w_{rnd,i^*} and, consequently, keeping w secret.

Step 2: Verifier's First Challenge

- $\mathcal{V}$ samples random coefficients $\gamma_1, \ldots, \gamma_m$ from $\mathbb{F}_{q^\mu}$ and sends them to $\mathcal{P}$.

- $\mathcal{P}$ combines all equations into a single polynomial $Q(X)$ and sends it to $\mathcal{V}$:

$$Q(X) = P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j^{[h]}(X, P_1(X), \ldots, P_n(X))$$

where the homogenized polynomial is defined as:

$$f_j^{[h]}(Y, X_1, \ldots, X_n) \coloneqq Y^d \cdot f_j\left(\frac{X_1}{Y}, \ldots, \frac{X_n}{Y}\right).$$

Remark: Since f_j has degree 2 and the polynomials $P_j(X)$ have degree 1, the composition would normally yield a polynomial of degree 2. However, the leading coefficient of $Q(X)$ is exactly $\sum_{j=1}^{m} \gamma_j f_j(w_1, \ldots, w_n)$. Since w is a valid witness, $f_j(w) = 0$ for all j. Consequently, $Q(X)$ becomes a linear polynomial. This degree decrease allows $\mathcal{V}$ to check the validity of the solution without learning the secret itself. Furthermore, verifying the algebraic relation is not sufficient on its own. To prevent a malicious prover from simply sampling a random linear polynomial, the protocol requires a second challenge. This step ensures that the evaluated polynomial is consistent with the initial commitment.

Step 3: Verifier's Second Challenge

- $\mathcal{V}$ chooses randomly an index $i^* \in \{1, \ldots, N\}$ corresponding to the evaluation point $r = \phi(i^*) \in \mathrm{Im}(\phi)$ and sends i^* to $\mathcal{P}$;

- $\mathcal{P}$ reveals seed_i for all $i \neq i^*$, alongside com_{i^*} and aux. This allows $\mathcal{V}$ to compute the evaluations $P_j(r)$ for all $j \in \{0, \ldots, n\}$. Indeed, the following equalities hold:

$$
\begin{aligned}
&\phi(i^*) \cdot aux_j + \sum_{\substack{i=1 \\ i \neq i^*}}^{N} (\phi(i^*) - \phi(i)) \cdot (w_{rnd,i})_j \\
&= \phi(i^*) \cdot aux_j + \sum_{i=1}^{N} (\phi(i^*) - \phi(i)) \cdot (w_{rnd,i})_j \\
&= \phi(i^*) \cdot \left(aux_j + \sum_{i=1}^{N} (w_{rnd,i})_j \right) - \sum_{i=1}^{N} \phi(i) \cdot (w_{rnd,i})_j \\
&= \phi(i^*) \cdot (aux_j + (w_{acc})_j) + (w_{base})_j \\
&= \phi(i^*) \cdot \overline{w}_j + (w_{base})_j = P_j(\phi(i^*)).
\end{aligned}
$$

Step 4: Verification

- $\mathcal{V}$ reconstruct the commitment by computing $\mathrm{com}'_i = h(\mathrm{seed}_i)$ for all $i \neq i^*$, and setting $\mathrm{com}'_{i^*} = \mathrm{com}_{i^*}$, which was provided by $\mathcal{P}$. $\mathcal{V}$ checks that
$$h_1 \stackrel{?}{=} h(\mathrm{com}'_1, \ldots, \mathrm{com}'_N, \mathrm{aux}).$$
- Finally $\mathcal{V}$ accepts the proof if and only if the polynomial evaluations match $Q(r)$:
$$P_0(r) + \sum_{j=1}^{m} \gamma_j \cdot f_j^{[h]}(r, P_1(r), \ldots, P_n(r)) \stackrel{?}{=} Q(r).$$

RYDE Generalized Proof System

Prover $\mathcal{P}(w)$		**Verifier** $\mathcal{V}(f_1, \ldots, f_m)$
$\text{seed}_1, \ldots, \text{seed}_N \leftarrow_\$ \{0,1\}^\lambda$		
$\text{com}_i \leftarrow h(\text{seed}_i)$ for $i \in \{1, \ldots, N\}$		
$w_{\text{rnd},i} \leftarrow \text{PRG}(\text{seed}_i)$		
Compute $P_1, \ldots, P_n, P_0$ and aux		
$h_1 \leftarrow h(\text{com}_1, \ldots, \text{com}_N, \text{aux})$		
	$\xrightarrow{h_1}$	
	$\xleftarrow{\gamma_1, \ldots, \gamma_m}$	$\gamma_j \leftarrow_\$ \mathbb{F}_{q^\mu}$
$Q(X) \leftarrow P_0(X) + \sum_{j=1}^{m} \gamma_j \cdot f_j^{[h]}(X, \vec{P}(X))$		
	$\xrightarrow{Q(X)}$	
	$\xleftarrow{i^*}$	$i^* \leftarrow_\$ \{1, \ldots, N\}$
		$r \leftarrow \phi(i^*)$
$\text{Resp} \leftarrow (\{\text{seed}_i\}_{i \neq i^*}, \text{com}_{i^*}, \text{aux})$		
	$\xrightarrow{\text{Resp}}$	
		Reconstruct $\text{com}'_1, \ldots, \text{com}'_N$
		Set $\text{com}'_{i^*} := \text{com}_{i^*}$
		Reconstruct $P_j(r)$ for $j \in \{0, \ldots, n\}$
		Check $h_1 \stackrel{?}{=} h(\text{com}'_1, \ldots, \text{com}'_N, \text{aux})$
		$V \leftarrow P_0(r) + \sum_{j=1}^{m} \gamma_j \cdot f_j^{[h]}(r, \vec{P}(r))$
		Check $Q(r) \stackrel{?}{=} V$

3.2 Example

First of all, let us fix the following parameters:

$$\begin{aligned} q &= 2 \text{ (base field size)}, \\ m &= 4 \text{ (number of equations)}, \\ N &= 5 \text{ (number of seeds)}, \\ n &= 4 \text{ (total length)}, \\ \mu &= 3 \text{ (extension field degree)}. \end{aligned}$$

We consider the field $\mathbb{F}_{2^3}$*, using the isomorphism with* $\mathbb{F}_2[x]/(x^3+x+1)$ *and defining* $b \coloneqq \bar{x}$*.*

Setting of the problem: $\mathcal{P}$ *wants to prove to* $\mathcal{V}$ *the knowledge of a root* $w \in \mathbb{F}_2^4$ *of the following polynomials:*

$$\begin{aligned} f_1 &= x_0, \\ f_2 &= x_1 + x_2, \\ f_3 &= x_1 x_2 + 1, \\ f_4 &= x_0 x_1 + x_3. \end{aligned}$$

Assume that $\mathcal{P}$ *knows* $w = (0,1,1,0)$*. We follow the steps explained above:*

Step 1: Prover's Commitment. *First, starting from the seeds*

$$\text{seeds} = [\texttt{"Seed\_1"}, \texttt{"Seed\_2"}, \texttt{"Seed\_3"}, \texttt{"Seed\_4"}, \texttt{"Seed\_5"}].$$

Using the SHAKE256 function as PRG, $\mathcal{P}$ *generates the following vectors:*

- $w_{\text{rnd}} = [(0,1,1,1,0), (1,0,0,0,1), (0,0,1,0,1), (1,0,1,0,0), (1,1,0,1,1)]$;
- $w_{\text{acc}} = (1,0,1,0,1) \in \mathbb{F}_2^5$;
- $w_{\text{base}} = (b^2+1, b^2+1, b^2+b+1, b^2+1, 0) \in \mathbb{F}_{2^3}^5$.

Alongside these, $\mathcal{P}$ computes the commitment vector com *by applying the SHA3-256 hash function componentwise to the elements of* seeds.

The polynomials obtained from the vectors above are

$$P = [X + b^2 + 1, b^2 + 1, X + b^2 + b + 1, X + b^2 + 1, 0].$$

$\mathcal{P}$ computes the auxiliary vector $\text{aux} = (0, 0, 0, 1, 1) \in \mathbb{F}_2^5$.

Finally, the global hash h_1 of aux *and* com *is computed and sent to $\mathcal{V}$ as the following byte string:*

$$h_1 = \texttt{b'\textbackslash xd6s\textbackslash x135\textbackslash xb7\textbackslash x13\textbackslash xc7\textbackslash x1d\{\textbackslash xe6@n\textbackslash xf0T\textbackslash xe1\textbackslash x89}$$
$$\texttt{\$\textbackslash xd9?\textbackslash x04Y\textbackslash x0b9(\textbackslash xa6\textbackslash xe3\textbackslash xb0\textbackslash xe2\textbackslash x00\textbackslash x97\textbackslash xbe'}.$$

This value is computed using the SHA3-256 hash function.

Step 2: Verifier's First Challenge.

$\mathcal{V}$ samples the vector of coefficients

$$\gamma = [1, b^2, 1, b^2 + 1] \in \mathbb{F}_{2^3}^4.$$

$\mathcal{P}$ generates and sends the polynomial

$$Q(X) = b^2 X + b^2 + b \in \mathbb{F}_{2^3}[X].$$

Step 3: Verifier's Second Challenge

$\mathcal{V}$ chooses randomly $i^* = 3$ corresponding to the evaluation point $b+1$. Then $\mathcal{P}$ reveals

$$\begin{aligned} aux &= (0, 0, 0, 1, 1), \\ \text{seeds}_{revealed} &= [\texttt{"Seed\_1"}, \texttt{"Seed\_2"}, None, \texttt{"Seed\_4"}, \texttt{"Seed\_5"}], \\ \text{com}_3 &= h(\texttt{b'Seed\_3'}). \end{aligned}$$

Hence, $\mathcal{V}$ is able to compute the polynomial evaluations in $b+1$, corresponding to:

$$(b^2 + b, b^2 + 1, b^2, b^2 + b, 0).$$

Step 4: Verification

$\mathcal{V}$ reconstruct the commitments $\text{com}'_i = h(\text{seed}_i)$ *for all* $i \neq 3$ *and checks that the hash* h_1 *provided by* $\mathcal{P}$ *corresponds to the hash* $h(\text{com}'_1, \ldots, \text{com}'_5, \text{aux})$. *Finally* $\mathcal{V}$ *checks that the polynomial evaluation are consistent with* $Q(b{+}1) = 1$*:*

$$b^2 + b + 1 \cdot (b^2 + 1) + b^2 \cdot b + 1 \cdot 0 + (b^2 + 1) \cdot b \stackrel{?}{=} 1.$$

Since the two evaluations match, $\mathcal{V}$ *accepts the proof.*

Security Analysis

The completeness property is clearly satisfied. If both parties follow the protocol honestly and $\mathcal{P}$ knows a valid witness w, then $\mathcal{V}$ will always accept the proof.

Soundness: Assume the absolute security of h and PRG and suppose that a malicious prover $\tilde{\mathcal{P}}$ does not have a valid solution w. Hence, suppose that $\tilde{\mathcal{P}}$ chooses a vector $\tilde{w}$ such that $f_{j^*}(\tilde{w}) \neq 0$ for at least one index j^*.

To successfully deceive $\mathcal{V}$, $\tilde{\mathcal{P}}$ must pass both challenges. We analyze this probability by considering the two possible ways that could allow $\tilde{\mathcal{P}}$ to cheat:

- *Accidentally degree decrease:* in order to pass the first challenge, $\tilde{\mathcal{P}}$ must build a linear polynomial $\tilde{\mathcal{Q}}(X)$. If $\tilde{\mathcal{P}}$ follows the commitment procedure with $\tilde{w}$, the probability that $\tilde{Q}(X)$ collapses to degree 1 is exactly the probability that its leading coefficient vanishes, i.e.

  $$\sum_{j=1}^{m} \gamma_j f_j(\tilde{w}_1, \ldots, \tilde{w}_n) = 0.$$

  Given the randomness of γ_j in $\mathbb{F}_{q^\mu}$, this occurs with probability $\frac{1}{q^\mu}$.

- *Forging the Polynomial:* If the degree does not decrease, $\tilde{\mathcal{P}}$ must forge and send a polynomial $\tilde{Q}'(X)$ of degree 1. In the final step, $\mathcal{V}$ evaluates the equations at a randomly chosen point $r \in \text{Im}(\phi)$ and checks if the

result matches $\tilde{Q}(r)$. Thus, $\mathcal{V}$ will accept only if $\tilde{Q}(r) = \tilde{Q}'(r)$, meaning that r must be a root of the difference polynomial $\tilde{Q}(X) - \tilde{Q}'(X)$. Since $\tilde{Q}(X)$ has degree 2 and $\tilde{Q}'(X)$ has degree 1, their difference is a non-zero polynomial of degree 2. Given that the evaluation space is of size N, the probability that r is a root of $\tilde{Q}(X) - \tilde{Q}'(X)$ is at most $\frac{2}{N}$.

The total probability that a malicious prover $\mathcal{P}$ cheats in a single execution of the protocol is bounded by $\frac{1}{q^\mu} + \left(1 - \frac{1}{q^\mu}\right) \frac{2}{N}$.

Zero-Knowledge: After the execution of the protocol, $\mathcal{V}$ obtains the polynomial $Q(X)$, all the seeds except seed_{i^*}, the commitment com_{i^*} and aux. $\mathcal{V}$ does not learn anything about the secret witness w due to the following reasons. Because seed_{i^*} is not revealed to $\mathcal{V}$, the vectors w_{base} and w_{acc} remain unknown to $\mathcal{V}$. Consequently, the polynomial $P_0(X)$ is random and acts as a blinding factor for the linear polynomial $Q(X)$, covering any information about the witness. Moreover, the randomness of w_{base} in $\mathbb{F}_{q^\mu}^{n+1}$ ensures that the evaluations $P_j(r) = w_j \cdot r + (w_{\text{base}})_j$ for $j \in \{1, \ldots, n\}$ cannot be used to extract any information about w.

This interactive protocol can be converted into a digital signature scheme using the Fiat-Shamir transform. However, we will first adapt this to the RSD problem and then we will construct the signature scheme used in RYDE.

3.2 RYDE proof system for RSD_s

Our goal is to adapt the interactive proof system presented in Subsection 3.1 to encode RSD_s instances. Specifically, rather than proving the knowledge of a generic root w for a set of public polynomials, we aim to prove the knowledge of a pair (s', C') that constitutes a valid solution to an RSD_s instance.

Before detailing the protocol, we establish the necessary setup. Let $\phi : \{1, \ldots, N\} \hookrightarrow \mathbb{F}_{q^\mu}$ be a public injective function, PRG be a pseudorandom generator and h be a hash function. Let n, k, r, μ be positive integers with

$k, r \leq n$ and $N \leq q^\mu$.

Core Objective

Given a public RSD_s instance $(H, y) \in \mathrm{Mat}_{n-k\times n}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^\mu}^{n-k}$, the prover $\mathcal{P}$ wants to convince the Verifier $\mathcal{V}$ that he knows a pair $(s', C') \in \mathbb{F}_{q^\mu}^{r-1} \times \mathrm{Mat}_{r\times(n-r)}(\mathbb{F}_q)$ such that:

$$H(sC)^\top = y^\top, \text{ where } s = (1 \parallel s') \in \mathbb{F}_{q^\mu}^r \quad \text{and} \quad C = [I_r \parallel C'] \in \mathrm{Mat}_{r\times n}(\mathbb{F}_q).$$

Step 1: Prover's Commitment

1. $\mathcal{P}$ samples N random seeds $\mathrm{seed}_1, \ldots, \mathrm{seed}_N$ and computes their commitments $\mathrm{com}_i = h(\mathrm{seed}_i)$ for $i \in \{1, \ldots, N\}$.

2. From each seed, $\mathcal{P}$ generates the components using the PRG:

   $$w_{\mathrm{rnd},i} = \mathrm{PRG}(\mathrm{seed}_i) = (v_{\mathrm{rnd},i}, s'_{\mathrm{rnd},i}, C'_{\mathrm{rnd},i}) \in \mathbb{F}_{q^\mu} \times \mathbb{F}_{q^\mu}^{r-1} \times \mathrm{Mat}_{r\times n-r}(\mathbb{F}_{q^\mu})$$

   and computes the vectors:

   $$w_{\mathrm{acc}} = (v_{\mathrm{acc}}, s'_{\mathrm{acc}}, C'_{\mathrm{acc}}) = \sum_{i=1}^{N} w_{\mathrm{rnd},i},$$

   $$w_{\mathrm{base}} = (v_{\mathrm{base}}, s'_{\mathrm{base}}, C'_{\mathrm{base}}) = -\sum_{i=1}^{N} \phi(i) \cdot w_{\mathrm{rnd},i}.$$

3. $\mathcal{P}$ creates the linear polynomials for the witness (s', C') and the masking linear polynomial P_v:

   $$\begin{aligned} P_{s'}(X) &= s' \cdot X + s'_{\mathrm{base}} \in (\mathbb{F}_{q^\mu}[X])^{r-1}, \\ P_{C'}(X) &= C' \cdot X + C'_{\mathrm{base}} \in \mathrm{Mat}_{r\times n-r}(\mathbb{F}_{q^\mu}[X]), \\ P_v(X) &= v_{\mathrm{acc}} \cdot X + v_{\mathrm{base}} \in \mathbb{F}_{q^\mu}[X]. \end{aligned}$$

   The multiplication by X should be intended componentwise.

4. $\mathcal{P}$ computes $\text{aux} = \tilde{w} - w_{\text{acc}}$ where $\tilde{w} = (v_{\text{acc}}, s', C')$. We denote by $\text{aux}_v, \text{aux}_{s'}, \text{aux}_{C'}$ respectively the first, second and third coordinate of aux. Note that by definition $\text{aux}_v = 0$.

5. $\mathcal{P}$ computes the global commitment as $h_1 \coloneqq h(\text{com}_1, \ldots, \text{com}_N, \text{aux})$ and sends it to $\mathcal{V}$.

Step 2: First Challenge

- $\mathcal{V}$ samples a random vector $\gamma \in \mathbb{F}_{q^\mu}^{n-k}$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ combines the challenge and the polynomials into a single polynomial $P_\alpha(X)$:

$$P_\alpha(X) = P_v(X) + \gamma \cdot (H \cdot (P_x(X))^\top - y^\top \cdot X^2) \in \mathbb{F}_{q^\mu}[X]$$

  where $P_x(X) = (X \parallel P_{s'}(X)) \cdot [I_r \cdot X \parallel P_{C'}(X)] \in (\mathbb{F}_{q^\mu}[X])^n$.

- $\mathcal{P}$ sends $P_\alpha(X)$ to $\mathcal{V}$.

3.3 Remark

The construction of $P_\alpha(X)$ ensures that the quadratic term cancels out if and only if the prover holds a valid solution to the RSD_s instance. To see why, let us analyze the leading coefficients of the polynomials involved.

By construction, the polynomial vector $(X \parallel P_{s'}(X))$ has degree 1 and its leading coefficient is the vector $s = (1 \parallel s')$. Similarly, the polynomial matrix $[I_r \cdot X \parallel P_{C'}(X)]$ has degree 1, with its leading coefficient being the matrix $C = [I_r \parallel C']$.

Consequently, the highest degree term of their product $P_x(X)$ is the quadratic term $(sC)X^2$. When this is multiplied by the parity-check matrix H, the quadratic term of $H \cdot P_x(X)$ evaluates exactly to $(H(sC)^\top)X^2$. Since the pair (s', C') is a valid solution, it satisfies the syndrome equation $H(sC)^\top = y^\top$. Thus, the X^2 term in $P_\alpha[X]$ cancels out. As discussed in Subsection 3.1,

this degree reduction allows $\mathcal{V}$ to check that $\mathcal{P}$ knows a valid solution, guaranteeing the zero-knowledge property of the protocol. The second challenge is necessary to prevent a malicious prover from forging a random linear polynomial.

Step 3: Second Challenge

- $\mathcal{V}$ randomly chooses a challenge index $i^* \in \{1, \ldots, N\}$, which corresponds to the evaluation point $r = \phi(i^*)$ and sends it to $\mathcal{P}$.
- $\mathcal{P}$ reveals seed_j for $j \neq i^*$, com_{i^*}, and aux.
- This allows $\mathcal{V}$ to compute the evaluations $s'_{\text{Eval}} = P_{s'}(r)$, $C'_{\text{Eval}} = P_{C'}(r)$, and $v_{\text{Eval}} = P_v(r)$. Indeed, following the same substitution as in Subsection 3.1, $\mathcal{V}$ computes:

  $$P_{s'}(r) = \phi(i^*) \cdot (\text{aux}_{s'} + s'_{\text{acc}}) + s'_{\text{base}}$$

  and analogously for C' and v.

Step 4: Final Verification

- $\mathcal{V}$ computes $\text{com}'_i = h(\text{seed}_i)$ for $j \neq i^*$ and checks that the global commitment h_1 coincides with $h(\text{com}'_1, \ldots, \text{com}'_N, \text{aux})$, where $\text{com}'_{i^*} := \text{com}_{i^*}$.
- $\mathcal{V}$ computes the evaluation vector:

  $$x_{\text{Eval}} = (r \parallel s'_{\text{Eval}}) \cdot [r \cdot I_r \parallel C'_{\text{Eval}}].$$

- $\mathcal{V}$ accepts the proof if and only if the following equation holds:

  $$P_\alpha(r) = v_{\text{Eval}} + \gamma \cdot (H \cdot x_{\text{Eval}} - y^\top \cdot r^2).$$

RYDE Proof System for RSD_s

Prover $\mathcal{P}(s', C')$		**Verifier** $\mathcal{V}(H, y)$
$\mathrm{seed}_1, \ldots, \mathrm{seed}_N \leftarrow_{\$} \{0,1\}^\lambda$		
$\mathrm{com}_i \leftarrow h(\mathrm{seed}_i)$ for $i \in \{1, \ldots, N\}$		
$w_{\mathrm{rnd},i} = (v_{\mathrm{rnd},i}, s'_{\mathrm{rnd},i}, C'_{\mathrm{rnd},i}) \leftarrow \mathrm{PRG}(\mathrm{seed}_i)$		
Compute $P_{s'}(X), P_{C'}(X), P_v(X)$ and aux		
$h_1 \leftarrow h(\mathrm{com}_1, \ldots, \mathrm{com}_N, \mathrm{aux})$		
	$\xrightarrow{\quad h_1 \quad}$	
	$\xleftarrow{\quad \gamma \quad}$	$\gamma \leftarrow_{\$} \mathbb{F}_{q^\mu}^{n-k}$
$P_x(X) \leftarrow (X \parallel P_{s'}(X)) \cdot [I_r \cdot X \parallel P_{C'}(X)]$		
$\Delta(X) \leftarrow H \cdot (P_x(X))^\top - y^\top \cdot X^2$		
$P_\alpha(X) \leftarrow P_v(X) + \gamma \cdot \Delta(X)$		
	$\xrightarrow{\quad P_\alpha(X) \quad}$	
	$\xleftarrow{\quad i^* \quad}$	$i^* \leftarrow_{\$} \{1, \ldots, N\}$
		$r \leftarrow \phi(i^*)$
$\mathrm{Resp} \leftarrow (\{\mathrm{seed}_i\}_{i \neq i^*}, \mathrm{com}_{i^*}, \mathrm{aux})$		
	$\xrightarrow{\quad \mathrm{Resp} \quad}$	
		Reconstruct com'_i for $i \neq i^*$,
		Set $\mathrm{com}'_{i^*} \coloneqq \mathrm{com}_{i^*}$
		Check $h_1 \stackrel{?}{=} h(\mathrm{com}'_1, \ldots, \mathrm{com}'_N, \mathrm{aux})$
		Reconstruct $s'_{\mathrm{Eval}}, C'_{\mathrm{Eval}}, v_{\mathrm{Eval}}$
		$x_{\mathrm{Eval}} \leftarrow (r \parallel s'_{\mathrm{Eval}}) \cdot [r \cdot I_r \parallel C'_{\mathrm{Eval}}]$
		$V \leftarrow v_{\mathrm{Eval}} + \gamma \cdot \left(H \cdot x_{\mathrm{Eval}}^\top - y^\top \cdot r^2\right)$
		Check $P_\alpha(r) \stackrel{?}{=} V$

3.4 Example

Let us fix the following parameters:

$$\begin{aligned} q &= 2 \text{ (base field size)}, \\ \mu &= 4 \text{ (extension field degree)}, \\ N &= 5 \text{ (number of seeds)}, \\ n &= 4 \text{ (total length)}, \\ r &= 2 \text{ (rank weight)} \end{aligned}$$

We consider the field $\mathbb{F}_{2^4}$*, using the isomorphism with* $\mathbb{F}_2[x]/(x^4+x+1)$*. We define* $\alpha \coloneqq \overline{x}$*.*

Setting of the problem: *Given the public syndrome* $y = (\beta^2+\beta, \beta^3+\beta)$ *and the parity-check matrix*

$$H = \begin{bmatrix} \beta^3+\beta & \beta^2+\beta+1 & \beta^3+\beta^2+\beta & 0 \\ \beta^3+\beta^2+\beta & \beta^2+\beta+1 & \beta^2 & \beta+1 \end{bmatrix},$$

the prover $\mathcal{P}$ *wants to convince the verifier* $\mathcal{V}$ *that he knows a secret pair* $(s', C') \in \mathbb{F}_{2^4} \times \mathrm{Mat}_{2\times 2}(\mathbb{F}_2)$ *satisfying*

$$sCH^\top = y,$$

where $s = (1 \parallel s')$ *and* $C = [I_r \parallel C']$*.*

Assume that $\mathcal{P}$ *knows the pair* $s' = (\beta^2)$ *and* $C' = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$*, yielding the product* sC *to be the vector* $(1, \beta^2, \beta^2+1, \beta^2)$ *of rank weight* 2*.*

We follow the steps explained above:

Step 1: Prover's Commitment

For simplicity, let the random seeds be

$$\text{seeds} = [\texttt{"seed\_1"}, \texttt{"seed\_2"}, \texttt{"seed\_3"}, \texttt{"seed\_4"}, \texttt{"seed\_5"}].$$

Using SHAKE256 as PRG, $\mathcal{P}$ *generates the following elements:*

- *the coordinates of* w_{rnd}*:*

$$\begin{aligned} s'_{\mathrm{rnd}} &= \left[(\beta^3), (\beta^3+\beta), (\beta), (\beta^3+\beta+1), (0)\right], \\ v_{\mathrm{rnd}} &= \left[(\beta^3+1), (\beta), (0), (\beta^2+\beta+1), (\beta^3)\right], \\ C'_{\mathrm{rnd}} &= \left[\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 1\\ 0 & 0\end{bmatrix}, \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}, \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\right]. \end{aligned}$$

- *the coordinates of* w_{base}*:*

$$\begin{aligned} s'_{\mathrm{base}} &= (0), \\ v_{\mathrm{base}} &= (\beta^2+1), \\ C'_{\mathrm{base}} &= \begin{bmatrix}\beta^3+\beta^2+\beta & \beta^3+\beta^2+\beta \\ 0 & \beta+1\end{bmatrix}. \end{aligned}$$

- *the coordinates of* w_{acc}*:*

$$\begin{aligned} s'_{\mathrm{acc}} &= (\beta^3+\beta+1), \\ v_{\mathrm{acc}} &= (\beta^2), \\ C'_{\mathrm{acc}} &= \begin{bmatrix}1 & 1\\ 0 & 0\end{bmatrix}. \end{aligned}$$

- *the auxiliary value:*

$$\mathrm{aux} = \left[(\beta^3+\beta^2+\beta+1), \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}, 0\right].$$

The resultant polynomials are:

$$\begin{aligned} P_{s'} &= (\beta^2 X), \\ P_{C'} &= \begin{bmatrix}X+\beta^3+\beta^2+\beta & \beta^3+\beta^2+\beta \\ X & X+\beta+1\end{bmatrix}, \\ P_v &= (\beta^2 X+\beta^2+1). \end{aligned}$$

Furthermore, $\mathcal{P}$ computes the commitment vector com *by applying componentwise the SHA3-256 hash function to* seeds. *Finally, $\mathcal{P}$ sends the global commitment h_1 of* aux *and* com *to $\mathcal{V}$ as the following byte string:*

$$h_1 = \texttt{b'\textbackslash rdv\textbackslash rP\textbackslash x9b5\textbackslash x9b\textbackslash x93\textbackslash x14\textbackslash x9d\textbackslash xff\textbackslash xa8U6\textbackslash xc0\textbackslash xfc\&}$$
$$\texttt{\textbackslash xd4\textbackslash x1b\textbackslash xcd\textbackslash x85\textbackslash x1c0\textbackslash x16\textbackslash xceb\textbackslash xd5\textbackslash xf1au\textbackslash xda'}.$$

Step 2: First Challenge

$\mathcal{V}$ samples the random challenge vector γ:

$$\gamma = \begin{bmatrix} \beta^2 + \beta + 1 & \beta + 1 \end{bmatrix}.$$

Then, $\mathcal{P}$ computes the aggregated polynomial $P_\alpha(X)$:

$$P_\alpha(X) = (\beta^2 + 1)X + \beta^2 + 1.$$

and sends it to $\mathcal{V}$.

Step 3: Second Challenge

Suppose that $\mathcal{V}$ requests to evaluate the polynomials at the challenge index $i^* = 3$, which corresponds to the evaluation point $r = \phi(3) = \beta^3$.

Consequently, $\mathcal{P}$ reveals aux, *the commitment* com_{i^*} *and the seeds* seed_j *for all $j \neq i^*$.*

Using the formulas detailed in the framework, $\mathcal{V}$ evaluates the polynomials $(P_{s'}(X), P_{C'}(X), P_v(X))$ at the point $r = \beta^3$:

$$\begin{aligned} s'_{\text{Eval}} &= (\beta^2 + \beta), \\ C'_{\text{Eval}} &= \begin{bmatrix} \beta^2 + \beta & \beta^3 + \beta^2 + \beta \\ \beta^3 & \beta^3 + \beta + 1 \end{bmatrix}, \\ v_{\text{Eval}} &= (\beta + 1). \end{aligned}$$

Step 4: Final Verification

First, $\mathcal{V}$ verifies the consistency of the global commitment h_1. By recalculating the hashes of the revealed seeds and combining them with com_{i^*} *and* aux, *$\mathcal{V}$ checks the correctness of the commitment h_1.*

$\mathcal{V}$ evaluates the aggregated polynomial $P_\alpha(X)$ at $r = \beta^3$:

$$P_\alpha(\beta^3) = (\beta^3 + \beta^2).$$

Using the evaluations s'_{Eval} and C'_{Eval} obtained in the previous step, $\mathcal{V}$ computes

$$x_{\mathrm{Eval}} = \begin{pmatrix} \beta^3 & \beta^2+\beta \end{pmatrix} \cdot \left[\begin{bmatrix} \beta^3 & 0 \\ 0 & \beta^3 \end{bmatrix} \quad \begin{bmatrix} \beta^2+\beta & \beta^3+\beta^2+\beta \\ \beta^3 & \beta^3+\beta+1 \end{bmatrix} \right] = (\beta^3+\beta^2, \beta^2+1, 0, \beta^2+\beta).$$

Finally, $\mathcal{V}$ checks that

$$v_{\mathrm{Eval}} + \Gamma \cdot (H \cdot x_{\mathrm{Eval}}^\top - y^\top \cdot r^2) \stackrel{?}{=} \beta^3 + \beta^2.$$

Since the evaluations match, $\mathcal{V}$ accepts the proof.

3.5 Remark

The security analysis described in Subsection 3.1 can be adapted to this context. The formal argument for the zero-knowledge property remains analogous. Regarding soundness, the condition $f_j(w) \neq 0$ is replaced by the condition $(H(sC)^\top)_j \neq (y^\top)_j$ for some coordinate j and the probabilistic reasoning follows identically.

3.3 RYDE signature scheme

In this subsection, we convert the interactive RYDE proof system into a non-interactive digital signature scheme. In order to decrease the size of the signature, RYDE protocol does not correspond completely to the scheme obtained by applying directly the procedure described in Subsection 1.4 to

the RYDE proof system. We will first describe the actual RYDE signature scheme and then compare it with the signature scheme obtained by the formal application of the Fiat-Shamir transform.

Before detailing the algorithms, we establish the necessary setup. Let $\phi\colon \{1,\dots,N\} \hookrightarrow \mathbb{F}_{q^\mu}$ be a public injective function, PRG be a pseudorandom generator, and h be a cryptographic hash function. Let n, k, r, μ be positive integers with $k, r \leq n$ and $N \leq q^\mu$.

Let the secret and public keys be a RSD_s instance.

- **Secret Key:** A valid solution $(s', C') \in \mathbb{F}_{q^\mu}^{r-1} \times \mathrm{Mat}_{r\times(n-r)}(\mathbb{F}_q)$.
- **Public Key:** The corresponding syndrome and parity-check matrix $(H, y) \in \mathrm{Mat}_{(n-k)\times n}(\mathbb{F}_{q^\mu}) \times \mathbb{F}_{q^\mu}^{n-k}$ satisfying $H(sC)^\top = y^\top$, where $s = (1 \parallel s')$ and $C = [I_r \parallel C']$.

Sign Algorithm

To sign a message M, the signer computes the signature as follows:

1. **Commitment:**
   - The signer samples N random seeds $\mathrm{seed}_1, \dots, \mathrm{seed}_N$ and computes $\mathrm{com}_i = h(\mathrm{seed}_i)$.
   - Using the PRG, the signer generates the random vectors $w_{\mathrm{rnd},i}$ and computes the accumulated vectors, base vectors and the polynomials $P_{s'}(X), P_{C'}(X), P_v(X)$.
   - Finally, the signer computes aux and the global commitment
   $$h_1 \coloneqq h(\mathrm{com}_1, \dots, \mathrm{com}_N, \mathrm{aux}).$$
2. **First Fiat-Shamir challenge:**
   - Instead of interacting with a verifier, the signer derives the challenge matrix $\gamma \in \mathbb{F}_{q^\mu}^{n-k}$ by applying PRG to the first commitment: $\gamma = \mathrm{PRG}(h_1)$.

3. **Aggregated Polynomial:**

   - The signer computes the polynomial

   $$P_\alpha(X) = P_v(X) + \gamma \cdot (H \cdot (P_x(X))^\top - y^\top \cdot X^2)$$

   Since the quadratic term cancels out, $P_\alpha(X)$ is a linear polynomial.

   - Denoting α_{mid} and α_{base} as its leading and constant terms respectively, the signer computes a second hash h_2:

   $$h_2 = h(h(M), (\alpha_{mid}, \alpha_{\text{base}}), h_1).$$

4. **Second Fiat-Shamir Challenge:**

   - The signer derives the evaluation index challenge by applying the PRG to second hashing $i^* = \text{PRG}(h_2)$. This defines the evaluation point $r = \phi(i^*)$.

5. **Opening:**

   - The signer prepares the response $\text{Resp} = (\{\text{seed}_j\}_{j \neq i^*}, \text{com}_{i^*}, \text{aux})$.
   - The final signature is $\sigma = (h_2, \alpha_{mid}, \text{Resp})$.

Verify Algorithm

Given a signature $\sigma = (h_2, \alpha_{\text{mid}}, \text{Resp})$ for a message M under the public key (H, y), where $\text{Resp} = (\{\text{seed}_j\}_{j \neq i^*}, \text{com}_{i^*}, \text{aux})$, the verifier executes the following steps:

1. **Challenge Reconstruction:**

   - The verifier recomputes the commitments $\text{com}'_j = h(\text{seed}_j)$ for each $j \neq i^*$, sets $\text{com}'_{i^*} := \text{com}_{i^*}$ and computes the first global commitment:

   $$h_1 = h(\text{com}'_1, \ldots, \text{com}'_N, \text{aux}).$$

- Subsequently, the verifier reconstructs the first challenge $\gamma = \mathrm{PRG}(h_1)$ and derives the evaluation index challenge $i^* = \mathrm{PRG}(h_2)$, which yields the evaluation point $r = \phi(i^*)$.

2. **Evaluation reconstruction:**

   - Using the revealed aux and seeds, the verifier reconstructs the polynomial evaluations $s'_{\mathrm{Eval}}, C'_{\mathrm{Eval}}$, and v_{Eval} at the point r.
   - The verifier then computes the evaluation vector x_{Eval} and evaluates the right-hand side of the equation that must correspond to $P_\alpha(r)$:
   $$P_\alpha(r) = v_{\mathrm{Eval}} + \gamma \cdot \big(H \cdot x_{\mathrm{Eval}}^\top - y^\top \cdot r^2\big).$$

3. **Final verification:**

   - Since $P_\alpha(r) = \alpha_{\mathrm{mid}} \cdot r + \alpha_{\mathrm{base}}$, the verifier reconstruct α_{base} as:
   $$P_\alpha(r) - \alpha_{\mathrm{mid}} \cdot r.$$
   - Finally, the verifier accepts the signature if and only if the following check holds:
   $$h_2 \stackrel{?}{=} h\big(h(M), (\alpha_{\mathrm{mid}}, \alpha'_{\mathrm{base}}), h_1\big).$$

3.6 Example

We use the same parameters, setting and challenges described in Example 3.4. Suppose the signer wants to sign a document $M =$ `"Hello"`.

Sign Algorithm:

- ***Commitment:*** *Since the signer generates the same seeds, commitments and polynomials as in the interactive case, the first global commitment* h_1 *matches the previously computed byte string.*

- ***First Fiat-Shamir Challenge:*** *The signer applies PRG to* h_1 *and obtain the challenge matrix*

$$\gamma = \mathrm{PRG}(h_1) = \begin{bmatrix} \beta^2 + \beta + 1 & \beta + 1 \end{bmatrix}.$$

- ***Aggregated Polynomial:*** *The signer computes* $P_\alpha(X) = (\beta^2+1)X + \beta^2 + 1$ *and obtains the coefficients*

$$\alpha_{\mathrm{mid}} = \beta^2 + 1, \quad \alpha_{\mathrm{base}} = \beta^2 + 1.$$

The signer computes $h_2 = h(h(M), (\alpha_{\mathrm{mid}}, \alpha_{\mathrm{base}}), h_1)$*, that is*

$h_2 =$ `b'\xb0\x92\x99-L\x8eZ\x04,\xf43\xab\xd7SA7`
`l\xa6\x08\xc3(m\xc6\xf7\x8c\'\x97\xc78\x0b\x0b\x11'`

- ***Second Fiat-Shamir Challenge:*** *The signer derives the evaluation index by applying PRG to the second hash:*

$$i^* = \mathrm{PRG}(h_2) = 3 \implies r = \phi(3) = \beta^3.$$

- ***Opening:*** *The signer prepares the response* $\mathrm{Resp} = (\{\mathrm{seed}_j\}_{j\neq 3}, \mathrm{com}_3, \mathrm{aux})$*. The output signature is* $\sigma = (h_2, \beta^2 + 1, \mathrm{Resp})$*.*

Verify Algorithm:

Upon receiving σ *and* M*, the verifier proceeds as follows:*

- ***Challenge Reconstruction:*** *The verifier recomputes* $\mathrm{com}'_j = h(\mathrm{seed}_j)$ *for* $j \neq 3$ *and sets* $\mathrm{com}'_3 = \mathrm{com}_3$*. By hashing these together with* aux*, the verifier reconstructs* h_1*. Consequently, the verifier computes* $\gamma = \mathrm{PRG}(h_1) = \begin{bmatrix} \beta^2 + \beta + 1 & \beta + 1 \end{bmatrix}$ *and* $i^* = \mathrm{PRG}(h_2) = 3$*, yielding* $r = \beta^3$*.*

- ***Evaluation Reconstruction:*** *The verifier reconstructs the evaluations* $s'_{\mathrm{Eval}}, C'_{\mathrm{Eval}}, v_{\mathrm{Eval}}$ *and computes* x_{Eval} *as in Step 4 of Example 3.4. Then, the verifier recovers* $P_\alpha(\beta^3)$ *as it follows:*

$$P_\alpha(\beta^3) = v_{\mathrm{Eval}} + \gamma \cdot \left(H \cdot x_{\mathrm{Eval}}^\top - y^\top \cdot r^2\right) = \beta^3 + \beta^2.$$

- ***Final Verification:*** *Knowing* $\alpha_{\text{mid}} = \beta^2 + 1$ *from the signature* σ*, the verifier recovers* α'_{base} *:*

$$\alpha'_{\text{base}} = P_\alpha(\beta^3) - \alpha_{\text{mid}} \cdot \beta^3 = (\beta^3 + \beta^2) - (\beta^2 + 1)\beta^3 = \beta^2 + 1.$$

Finally, the verifier checks if h_2 *matches* $h\big(h(M), (\alpha_{\text{mid}}, \alpha'_{\text{base}}), h_1\big)$*. Since the reconstructed* α'_{base} *matches* α_{base}*, the signature is accepted.*

Formal Fiat-Shamir Sign Algorithm

Let $\mathsf{pk} = (H, y)$ be the public key, $\mathsf{sk} = (s', C')$ be the secret key and M a message. The protocol obtained from the formal application of the Fiat-Shamir transform proceeds as follows:

1. **First Message** m_1**:** The signer samples N random seeds, computes the polynomials $P_{s'}, P_{C'}, P_v$ and the commitments together with aux. The first message is:

$$m_1 = (\text{com}_1, \ldots, \text{com}_N, \text{aux}).$$

2. **First Challenge:** The signer derives the random challenge vector:

$$\gamma = \text{PRG}(\mathsf{pk}, M, m_1).$$

3. **Second Message** m_2**:** The signer computes the aggregated polynomial $P_\alpha(X) = P_v(X) + \gamma \cdot \big(H \cdot (P_x(X))^\top - y^\top \cdot X^2\big)$. The second message is the polynomial:

$$m_2 = P_\alpha(X).$$

4. **Second Challenge:** The signer derives the evaluation index challenge:

$$i^* = \text{PRG}(\mathsf{pk}, M, m_1, \gamma, m_2).$$

5. **Third Message m_3:** The signer derives the third message corresponding to the value Resp:

$$m_3 = \text{Resp} = (\{\text{seed}_j\}_{j \neq i^*}, \text{com}_{i^*}).$$

6. **Signature Output:** The final signature consists of the concatenation of the three messages:

$$\sigma_{FS} = (m_1, m_2, m_3).$$

3.7 Remark (Comparison with RYDE)
The signature σ_{FS} contains some redundancies, indeed m_1 is completely obtainable from m_3. Moreover, the RYDE signature includes only the linear term α_{mid} of $P_\alpha(X)$ alongside the commitment h_2, which depends on both α_{mid} and α_{base}. This allows converting the goal of checking the equation

$$P_\alpha(r) \stackrel{?}{=} v_{\text{Eval}} + \gamma \cdot \left(H \cdot x_{\text{Eval}}^\top - y^\top \cdot r^2\right)$$

into a method for deriving the term α_{base}. Therefore, the verification turns into checking

$$h_2 \stackrel{?}{=} h\big(h(M), (\alpha_{\text{mid}}, \alpha'_{\text{base}}), h_1\big).$$

Note that since the actual RYDE protocol involves multiple repetitions, utilizing a single hash h_2 instead of transmitting several polynomial coefficients significantly decreases the signature size. Furthermore, the dependency by the message M is restricted to the hash h_2, which is sufficient to make the second challenge and consequently the protocol, dependent on M.

4 Gröbner bases

In this section, we present the theoretical tools necessary to develop algebraic attacks against the RSD_s problem.

The problem of finding the roots of a system of multivariate polynomials with coefficients in a field $\mathbb{F}$ is a challenge of crucial importance in many fields of mathematics. In particular, given a polynomial multivariate system $\mathcal{F} = \{f_1, \ldots, f_\ell\} \subset \mathbb{F}[x_1, \ldots, x_n]$, the study of the underlying ideal $I = (f_1, \ldots, f_\ell)$ reveals essential information about solutions to $\mathcal{F}$.

In the univariate case $\mathbb{F}[x]$, monomials are naturally ordered by their degree, every ideal is principal and the Euclidean algorithm allows us to find the generator. When moving to the multivariate case, things become significantly more complex. For example, the division algorithm depends on the order of division and the ideals can be generated by multiple elements. Gröbner basis is a particular generating set for the ideal, introduced by B. Buchberger in 1965, with the purpose of providing a solution to the multivariate case.

Before introducing a formal definition we must establish a way to order the monomials in the multivariate polynomial ring $R = \mathbb{F}[x_1, \ldots, x_n]$. In the multivariate setting, monomials of the same degree cannot be compared by degree alone. We denote a monomial in R as $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. A **term order** on R is a relation $\geq$ on the set of monomials satisfying the following conditions:

1. $\geq$ is a total ordering;
2. $\geq$ is compatible with the ring multiplication: if $x^\alpha \geq x^\beta$, then $x^\alpha \cdot x^\gamma \geq x^\beta \cdot x^\gamma$ for all $\gamma \in \mathbb{N}^n$;
3. $x^\alpha \geq 1$ for all $\alpha \in \mathbb{N}^n \setminus \{0\}$.

A term order $\geq$ allows us to unambiguously identify the leading monomial $\mathrm{lt}_\geq(f)$ and the leading coefficient $\mathrm{lc}_\geq(f)$ of any non-zero polynomial $f \in R$. We denote with $\mathbb{T}$ the set of monomials.

We give here two important examples of term orders. The **lexicographical order** orders monomials by prioritizing the position of the variables based on a fixed variable order, typically $x_1 > x_2 > \cdots > x_n$. Formally, let $\alpha, \beta \in \mathbb{N}^n$. We say that $x^\alpha >_{\text{lex}} x^\beta$ if the left-most non-zero coordinate of $\alpha - \beta \in \mathbb{Z}^n$ is positive. For example, in $\mathbb{F}_2[x, y, z]$ with $x > y > z$, we have:

$$xy^3z >_{\texttt{lex}} y^5z^2 \quad \text{since } (1,3,1) - (0,5,2) = (1,-2,-1).$$

However, $>_{\texttt{lex}}$ does not respect the degree and computing a Gröbner basis w.r.t. $>\texttt{lex}$ is, generally, computationally expensive.

The following term order is called **degree reverse lexicographic order** (degrevlex) and compares the total degree first, then prioritizes the higher power of the last variables with respect to a fixed variable order, reversing the outcome of the lexicographic order. Formally, let $\alpha, \beta \in \mathbb{N}^n$ and let $|\alpha| \coloneqq \sum_{i=1}^n \alpha_i$. We say that $x^\alpha >_{\texttt{degrevlex}} x^\beta$ if $|\alpha| > |\beta|$ or$|\alpha| = |\beta|$ and the right-most non-zero entry of $\alpha - \beta \in \mathbb{Z}^n$ is negative. Consider the monomials of the same degree x^2yz and xy^3. Their exponent vectors are, respectively, $(2,1,1)$ and $(1,3,0)$. The difference is $(2,1,1) - (1,3,0) = (1,-2,1)$. Therefore

$$xy^3 >_{\texttt{degrevlex}} x^2yz.$$

We call the term orders prioritizing the degree first degree-compatible term orders.

Once a term order $\succ$ is fixed, it is possible to generalize the division algorithm to polynomials in multiple variables. By prioritizing the leading terms w.r.t. $\succ$, the multivariate division algorithm allows us to divide a polynomial $f \in R$ by a set of polynomials $\mathcal{F} = \{f_1, \ldots, f_\ell\}$, yielding a representation $f = q_1f_1 + \cdots + q_mf_m + r$, where $q_1, \ldots, q_m \in R$ and no term of the remainder r is divisible by any of the leading terms $\text{lt}(f_i)$.

Algorithm 1 Multivariate Division Algorithm

Input: A polynomial $f \in R$, a list of polynomials $\mathcal{F} = (f_1, \ldots, f_m)$ in R and a fixed term order $\succ$.

Output: Quotients $q_1, \ldots, q_m \in R$ and a remainder $r \in R$ such that $f = q_1 f_1 + \cdots + q_m f_m + r$, where no term of r is divisible by any $\mathrm{lt}(f_i)$.

$q_1 \leftarrow 0, \ldots, q_m \leftarrow 0, r \leftarrow 0$
$p \leftarrow f$
while $p \neq 0$ **do**
    while $p \neq 0$ and $\exists i$ such that $\mathrm{lt}(f_i)$ divides $\mathrm{lt}(p)$ **do**
        $k = \min\{i \in \{1, \ldots, m\}$ such that $\mathrm{lt}(f_i)$ divides $\mathrm{lt}(p)\}$
        $q_k \leftarrow q_k + \frac{\mathrm{lt}(p)}{\mathrm{lt}(f_k)}$
        $p \leftarrow p - \frac{\mathrm{lt}(p)}{\mathrm{lt}(f_k)} f_k$
    end while
    $r \leftarrow r + \mathrm{lt}(p)$
    $p \leftarrow p - \mathrm{lt}(p)$
end while
return $q_1, \ldots, q_m, r$

The proof of termination of this algorithm is not obvious and can be found in [KR00, Theorem 1.6.4].

However, differently from the univariate case, the remainder r is not unique in general, since it depends on the order in which we divide by the divisors f_i.

4.1 Example

Consider $\mathbb{F}_3[x, y]$ *equipped with the lexicographical order (with* $x > y$*). Let* $f = x^2y + x$ *be the dividend and let the divisors be* $f_1 = xy - 1$ *and* $f_2 = x^2 - y$*. Notice that the leading term of the dividend,* $\mathrm{lt}(f) = x^2y$*, is divisible by both* $\mathrm{lt}(f_1) = xy$ *and* $\mathrm{lt}(f_2) = x^2$*.*

If we divide f *first by* f_1 *we obtain*

$$f = x \cdot (xy - 1) + 2x = x \cdot f_1 + 2x.$$

Since no term of $2x$ *is divisible by either* $\mathrm{lt}(f_1)$ *or* $\mathrm{lt}(f_2)$*, the algorithm stops, yielding the remainder* $r_1 = 2x$*.*

If we divide f *by* f_2 *we obtain:*

$$f = y \cdot (x^2 - y) + y^2 + x = y \cdot f_2 + y^2 + x.$$

The term y^2 *is not divisible by* x^2 *or* xy*. The same applies to the term* x*. Consequently, the algorithm terminates with a different remainder,* $r_2 = y^2 + x$*.*

To obtain the uniqueness of the remainder, we need a special generating set of the ideal $I = (f_1, \ldots, f_\ell)$ which describes the behavior of the leading terms of the entire ideal I. This motivates the following definitions.

Given a non-zero ideal $I \subset R$ and a fixed term order over R, we define the ideal of leading terms of I as the monomial ideal (i.e. there exists a generating set of monomials) generated by its leading terms

$$\mathrm{lt}(I) \coloneqq (\mathrm{lt}(f) \,|\, f \in I \setminus \{0\}) \subset \mathbb{F}[x_1, \ldots, x_n]$$

Before introducing Gröbner basis we introduce an important lemma for monomial ideals.

4.2 Lemma
A monomial x^α *belongs to a monomial ideal* $I = (x^{\alpha_1}, \ldots, x^{\alpha_t})$ *if and only if there exists an index* $j \in \{1, \ldots, t\}$ *such that* $x^{\alpha_j} \mid x^\alpha$*.*

Proof. The if direction is clear by definition. Conversely, let $x^\alpha \in I$. We can write $x^\alpha = \sum_{i=1}^{t} h_i x^{\alpha_i}$ for some polynomials $h_i \in \mathbb{F}[x_1, \ldots, x_n]$. Expanding each polynomial h_i into a linear combination of monomials, every term on the right-hand side is of the form $c_i \cdot x^\beta x^{\alpha_i}$ for some non-zero scalar $c_i \in \mathbb{F}$ and some monomial x^{β_i}. Since the left-hand side consists of the single monomial x^α, there exists at least one index j such that $x^\alpha = x^{\beta_j} x^{\alpha_j}$. This concludes the proof. □

4.3 Definition (Gröbner basis)
*A **Gröbner basis** $G = \{g_1, \ldots, g_t\}$ for the ideal I is a set of polynomials in I whose leading terms generate the ideal of leading terms:*

$$(\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t)) = \mathrm{lt}(I).$$

The first property that we show is that a Gröbner basis is a generating set for the ideal. Indeed, the inclusion $(g_1, \ldots, g_t) \subset I$ is clear by definition. Moreover, given a polynomial $f \in I$, the division algorithm produces a remainder $r = f - (q_1 g_1 + \cdots + q_t g_t) \in I$ whose leading term $\mathrm{lt}(r)$ is not divisible by any of $\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t)$. However, by definition of Gröbner basis, $\mathrm{lt}(r) \in (\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t))$. Lemma 4.2 yields a contradiction.

Another property of Gröbner basis is the uniqueness of the remainder.

4.4 Proposition
Fix a term order over R, let $G = \{g_1, \ldots, g_t\}$ be a Gröbner basis of the ideal $I = (g_1, \ldots, g_t)$ and f be a polynomial in R. Then the remainder of the division of f by G is unique.

Proof. Let $r_1 = f - (q_1 g_1 + \cdots + q_t g_t)$ and $r_2 = f - (q'_1 g_1 + \cdots + q'_t g_t)$ be two distinct remainders obtained from the division algorithm. Subtracting the two remainders, we obtain $r_1 - r_2 = (q_1 - q'_1)g_1 + \cdots + (q_t - q'_t)g_t \in I$, which implies $\mathrm{lt}(r_1 - r_2) \in \mathrm{lt}(I)$. Moreover, $\mathrm{lt}(r_1 - r_2)$ is a multiple of a term appearing in r_1 or r_2 and, consequently, does not belong to $\{\mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t)\}$. This contradicts the fact that G is a Gröbner basis. □

4.5 Example
As in Example 4.1 consider the polynomial $f = x^2y + x \in \mathbb{F}_3[x, y]$ and the ideal $I = (xy - 1, x^2 - y)$. We saw that multivariate division by the arbitrary generators $\mathcal{F} = \{xy - 1, x^2 - y\}$ under the lexicographic order ($x > y$) yielded two different remainders, $r_1 = 2x$ and $r_2 = y^2 + x$.

This arises because $\mathcal{F}$ is not a Gröbner basis. A Gröbner basis for this ideal is $G = \{x - y^2, y^3 - 1\}$. If we divide f by G, the uniqueness is guaranteed.

Any step of multivariate division is equivalent to the substitutions $x \to y^2$ and $y^3 \to 1$. Consequently, regardless of the order in which we apply the divisions, the algorithm will always terminate with the unique remainder $r = 2y^2$.

Gröbner bases are not unique in general, in the example above we can simply construct another Gröbner basis from $\{x - y^2, y^3 - 1\}$ by substituting $x - y^2$ with the sum of the two polynomials $\{x - y^2 + y^3 - 1, y^3 - 1\}$. This motivates to define the reduced Gröbner basis.

A Gröbner basis G for a polynomial ideal I is called a **reduced Gröbner basis** if it satisfies the following two conditions:

1. $\mathrm{lc}(g) = 1$ for all $g \in G$.
2. For all $g \in G$, no term appearing in g is divisible by $\mathrm{lt}(h)$ for any $h \in G \setminus \{g\}$.

Any Gröbner basis can be converted in a reduced one and the reduced Gröbner basis is unique. A proof of this can be found in [CLO15, Chapter 2, Section 7].

To algorithmically compute a Gröbner basis from an arbitrary set of generators, B. Buchberger introduced the concept of S-polynomials. Given two polynomials f and g, their S-polynomial is defined as

$$S(f, g) = \frac{\mathrm{mcm}(\mathrm{lm}(f), \mathrm{lm}(g))}{\mathrm{lt}(f)} \cdot f - \frac{\mathrm{mcm}(\mathrm{lm}(f), \mathrm{lm}(g))}{\mathrm{lt}(g)} \cdot g.$$

S-polynomial is designed to cancel out their leading terms and potentially yield new leading terms that belong to $\mathrm{lt}(I)$ but are not generated by $\mathrm{lt}(f)$ and $\mathrm{lt}(g)$. Buchberger's criterion states that a basis G is a Gröbner basis if and only if the S-polynomial of every pair of elements in G reduces to zero under multivariate division by G. Buchberger's algorithm, starting from an arbitrary generating set, computes S-polynomials for all pairs, reduces them and adds any non-zero remainder to the basis until the set obtained satisfies the Buchberger's criterion.

4.6 Example

Let us illustrate the an example of the application of Buchberger's algorithm. Consider the ideal $I = (f_1, f_2) \subset \mathbb{F}_5[x, y]$ *equipped with the* `degrevlex` *order* ($x \succ y$)*, where* $f_1 = x^2 - y$ *and* $f_2 = xy - x$*. The generating set* $\mathcal{F} = \{f_1, f_2\}$ *is not a Gröbner basis. We first compute the S-polynomial of the pair* (f_1, f_2)*, that is*

$$S(f_1, f_2) = \frac{x^2y}{x^2} \cdot f_1 - \frac{x^2y}{xy} \cdot f_2 = y(x^2 - y) - x(xy - x) = -y^2 + x^2 = x^2 - y^2.$$

Next, we reduce this S-polynomial using the current basis F*. Since the leading term is* x^2*, we can divide it by* f_1*:*

$$(x^2 - y^2) - 1 \cdot (x^2 - y) = -y^2 + y.$$

The remainder $-y^2 + y$ *cannot be divided by* $\mathrm{lt}(f_1)$ *or* $\mathrm{lt}(f_2)$*. Since it is non-zero, we must add this new polynomial to our set, forming* $F' = \{f_1, f_2, f_3\}$*.*

The algorithm then proceeds to compute the S-polynomials for the new pairs (f_2, f_3) *and* (f_1, f_3)*. For the pair* (f_2, f_3) *we have*

$$S(f_2, f_3) = \frac{xy^2}{xy} \cdot f_2 - \frac{xy^2}{-y^2} \cdot f_3 = y(xy - x) + x(-y^2 + y) = xy^2 - xy - xy^2 + xy = 0.$$

This S-polynomial reduces to zero. Next, we compute the S-polynomial for the pair (f_1, f_3)*, that is*

$$S(f_1, f_3) = \frac{x^2y^2}{x^2} \cdot f_1 - \frac{x^2y^2}{-y^2} \cdot f_3 = y^2(x^2 - y) + x^2(-y^2 + y) = x^2y - y^3.$$

We must reduce $x^2y - y^3$ *with respect to the set* F'*. We start by dividing the leading term* x^2y *by* $\mathrm{lt}(f_2) = xy$*:*

$$(x^2y - y^3) - x \cdot (xy - x) = x^2 - y^3.$$

The new leading term is x^2*, which can be divided by* $\mathrm{lt}(f_1) = x^2$*:*

$$(x^2 - y^3) - 1 \cdot (x^2 - y) = -y^3 + y.$$

Now the leading term is $-y^3$, which is divisible by $\mathrm{lt}(f_3) = -y^2$:

$$(-y^3 + y) - y \cdot (-y^2 + y) = -y^2 + y.$$

Finally, dividing this remainder once more by f_3 yields:

$$(-y^2 + y) - 1 \cdot (-y^2 + y) = 0.$$

Since all S-polynomials reduce to zero, Buchberger's algorithm terminates. The set $F' = \{x^2 - y, xy - x, -y^2 + y\}$ is a Gröbner basis for the ideal I.

The termination of this algorithm is guaranteed by the Ascending Chain Condition on the monomial ideals generated by the leading terms. A formal proof of this can be found in [CLO15, Chapter 2, Section 7]. This guarantees that any ideal admits a finite Gröbner basis for any given term order. However, since Buchberger's algorithm needs to compute the S-polynomial once at a time, in practice it is computationally expensive. For this reason, modern algorithms for computing Gröbner bases relies on linear algebra techniques, replacing polynomial divisions with Gauss eliminations of matrices.

4.1 Solving polynomial systems with Gröbner bases

Suppose we are given a multivariate polynomial system $\mathcal{F} = \{f_1, \ldots, f_t\} \subset R$ (where $R = \mathbb{F}[x_1, \ldots, x_n]$), we want to describe the solution $z_1, \ldots, z_n$ in a fixed field extension satisfying

$$\begin{cases} f_1(z_1, \ldots, z_n) = 0 \\ f_2(z_1, \ldots, z_n) = 0 \\ \vdots \\ f_t(z_1, \ldots, z_n) = 0. \end{cases}$$

We define the **zero locus** of $I = (f_1, \ldots, f_t) \subset R$ in the algebraic closure $\bar{\mathbb{F}}$ of the field $\mathbb{F}$ as the set

$$Z(I) = \{P \in \bar{\mathbb{F}} \mid \forall f \in I \ f(P) = 0\} = \{P \in \bar{\mathbb{F}} \mid \forall i \in \{1, \ldots, t\} \ f_i(P) = 0\}.$$

If we are interested in solutions over a specific subfield $\mathbb{K}$, we define the zero locus of I in $\mathbb{K}$ as the restriction $Z(I) \cap \mathbb{K}^n$. When the field is not specified we mean the zero locus in the algebraic closure.

The goal of this section is to describe how a Gröbner basis of I with respect to the Lexicographic term order can describe the solution of this system. This aspect is described in what is called the Shape Lemma. Before detailing it, let us introduce some theorems and some definitions.

First we recall two theorems that will be useful for what follows.

4.7 Theorem (Macaulay)
Let $I \subset R$ and fix a term order. A basis for the k-vector space R/I is given by:

$$\{x^\alpha \in \mathbb{T} \mid \forall x^\alpha \notin \mathrm{lt}(I)\}.$$

An immediate consequence of this is that if G is a Gröbner basis for I, then the set $\{t \in \mathbb{T} \mid \forall g \in G, \mathrm{lt}(g) \nmid t\}$ is a basis for the k-vector space R/I. We call this set the **canonical basis** of R/I.

4.8 Theorem
Let $\mathbb{F}$ be an algebraically closed field and $I \subseteq R = \mathbb{F}[x_1, \ldots, x_n]$ be an ideal. Then the following facts are equivalent:

1. $|Z(I)| < +\infty$.
2. $\dim_k(R/I) < +\infty$.

The proofs of these theorems can be found in [CLO15, Chapter 5, Section 3].

We say that the ideal I is **zero-dimensional** if $|Z(I)| < +\infty$ and we call the **degree of the ideal** $d(I) = \dim_{\mathbb{F}}(R/I)$.

We say that the ideal I is in **normal form** w.r.t. the variable x_n if all the points in its zero locus have distinct x_n-coordinates. Formally, for all $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_n)$ in $Z(I)$, if $P \neq Q$ then $p_n \neq q_n$.

Assuming that $\mathbb{F}$ is a finite field or an algebraic closed field and that I is a radical zero-dimensional ideal in normal position with respect to x_n, the **Shape Lemma** states that the reduced Gröbner basis of I with respect to the lexicographic order ($x_1 > x_2 > \cdots > x_n$) has the following form:

$$\mathcal{G} = \{x_1 - g_1(x_n), x_2 - g_2(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), g_n(x_n)\}$$

where g_n is a polynomial of degree $d = |Z(I)|$, and $\deg(g_i) < d$ for all $i \in \{1, \ldots, n-1\}$.

4.9 Remark

Note that if $\mathbb{F}_q$ is a finite field, then for any ideal $I = (f_1, \ldots, f_t) \subset R = \mathbb{F}_q[x_1, \ldots, x_n]$ we can add the field equations $\mathcal{Q} = \{x_1^q - x_1, \ldots, x_n^q - x_n\}$ to I and we obtain the ideal J generated by $\mathcal{Q} \cup \{f_1, \ldots, f_t\}$. The zero-dimensional property is automatically obtained, since by construction the zero locus of J corresponds to $Z(I) \cap \mathbb{F}^n$ and, consequently, $|Z(J)| < +\infty$. Furthermore, J is also radical. Fix a polynomial $f \in R$ such that $f^\ell \in J$. First, we observe that $a^q = a$ for all $a \in \mathbb{F}_q$ and $x_i^q \equiv x_i \pmod{J}$ by construction. Consequently, we have:

$$f(x_1, \ldots, x_n)^q \equiv f(x_1^q, \ldots, x_n^q) \equiv f(x_1, \ldots, x_n) \pmod{J}.$$

This implies that $f^q - f \in J$. By iterating this relation k times, we obtain $f^{q^k} - f \in J$ for any integer $k \geq 1$.

Now, choose an integer k such that $q^k \geq \ell$, then we have that

$$f^{q^k} = f^\ell \cdot f^{q^k - \ell} \in J.$$

Since both f^{q^k} and $f^{q^k} - f$ belong to J, we conclude that $f \in J$.

Adding the field equation to the ideal I often change the complexity of computing a Gröbner basis, but is generally a good choice for fields $\mathbb{F}_q$ of small size. In the next section we will experimentally compare, in our case of interest, the difference between adding the field equations or not.

Observe also that, if we have a unique solution, the normal position condition is automatically satisfied.

The Shape Lemma reduces the task of solving systems to finding a reduced Gröbner basis w.r.t. `lex`, determining the roots of the univariate polynomial $g_n(x_n)$ and, subsequently, substituting them into the other polynomials g_i to obtain each coordinate. However, finding directly a Gröbner basis w.r.t. the lexicographic term order is generically computationally expensive. To overcome this problem, in 1993 Faugére, Gianni, Lazard and Mora introduced the FGLM algorithm in [Fau+93]. This algorithm allows one to convert, in the case of zero-dimensional ideals, a Gröbner basis G_1 w.r.t. a term order into a Gröbner basis G_2 for the lexicographic term order. We will not describe the details of the algorithm, but we present a toy example to give an intuition of how it works. Before doing this, consider the quotient ring R/I. Since I is a zero-dimensional ideal, R/I is a finite $\mathbb{F}$-linear dimensional vector space. By Macaulay's Theorem (Theorem 4.7), a canonical basis for this vector space is given by

$$\mathcal{W}_\prec = \{t \in \mathbb{T} \mid \forall g \in G_1 \ \mathrm{lt}_1(g) \nmid t\}$$

which is the set of monomials that are not divisible by the leading term of any polynomial in the initial Grobner basis $\mathcal{G}_1$. Consequently, the dimension of R/I as an $\mathbb{F}$-vector space is finite and this dimension is called the degree of the ideal, denoted by $D = \deg(I)$. This guarantees for each $i \in \{1, \ldots, n\}$ that the elements

$$\bar{1}, \bar{x_i}, \bar{x_i^2}, \ldots, \bar{x_i^D}$$

are linear dependent, since they are $D + 1$ elements in R/I. We call the elements $\bar{f} \in R/I$ represented as a vector w.r.t. the basis $W_\prec$ the normal form for f in I, denoted by $\mathrm{NF}_\prec(f, I)$.

In what follows, we show how to find the univariate polynomials $p_i(x_i) \in I \cap k[x_i]$.

4.10 Example

Let $I = (x^2 - y, y^2 - 1) \subset \mathbb{F}_5[x, y]$ be a zero-dimensional ideal. We assume that we have already computed its Gröbner basis with respect to the Degree

Reverse Lexicographic (DRL) order with $x >_{DRL} y$*:*

$$\mathcal{G}_{DRL} = \{x^2 - y, y^2 - 1\}$$

The leading terms are x^2 *and* y^2*. The dimension of the quotient ring* $\mathbb{F}_5[x, y]/I$ *as a* $\mathbb{F}_5$*-vector space is* $D = 4$ *and its canonical basis, ordered by DRL, is:*

$$\mathcal{W}_{DRL} = \{1, y, x, xy\} = \{\omega_1, \omega_2, \omega_3, \omega_4\}$$

We want to find the univariate polynomial $p_x(x) \in I \cap \mathbb{F}_5[x]$*. We iteratively compute* $\overline{x^j}$ *with respect to* $\mathcal{G}_{DRL}$ *for* $j = 0, 1, 2, \ldots$ *and express the results as coordinate vectors in the basis* $\mathcal{W}_{DRL}$*:*

- *Let* $j = 0$*: the element* x^0 *is simply* 1*, which corresponds to the vector* $v_0 = (1, 0, 0, 0)$*;*
- *Let* $j = 1$*: we multiply the previous result by* x*. Since* x *cannot be reduced by* $\mathcal{G}_{DRL}$*, we obtain the vector:*

$$v_1 = (0, 0, 1, 0)^T$$

- *Let* $j = 2$*: consider* x^2*. Using the first polynomial* $x^2 - y \in \mathcal{G}_{DRL}$*, we can reduce* x^2 *to* $y = \omega_2$*, obtaining:*

$$v_2 = (0, 1, 0, 0)^T$$

- *Let* $j = 3$*: We multiply the previous result* y *by* x*, obtaining* xy*. This is not reducible by* $\mathcal{G}_{DRL}$*. The coordinate vector is:*

$$v_3 = (0, 0, 0, 1)^T$$

- *Let* $j = 4$*: We multiply the previous result* xy *by* x*, obtaining* x^2y*. Substituting* x^2 *with* y*, we get* $y \cdot y = y^2$*. Then, using the second polynomial* $y^2 - 1 \in \mathcal{G}_{DRL}$*, we reduce* y^2 *to* $1 = \omega_1$*. The coordinate vector is:*

$$v_4 = (1, 0, 0, 0)^T$$

Since $D = 4$, we are guaranteed to find a linear dependency among the 5 vectors $\{v_0, v_1, v_2, v_3, v_4\}$. It is immediate to see that:

$$v_4 - v_0 = (1,0,0,0)^T - (1,0,0,0)^T = (0,0,0,0)^T$$

This vector relation translates into the belonging of $x^4 - 1$ to I.

Now we consider another example in which, we describe how the FGLM algorithm allow us to convert a Gröbner basis w.r.t. the `degrevlex` term order into another one for the lexicographic term order.

4.11 Example

Consider the ideal $I = (x^2 + y, y^2 + x) \subset \mathbb{F}_5[x, y]$. We want to compute its Gröbner basis with respect to the lexicographic order ($x >_{\texttt{lex}} y$).

First, let us assume we have computed a reduced Gröbner basis with respect to the `degrevlex` *order ($x >_{DRL} y$). The generators already form a DRL basis:*

$$\mathcal{G}_{DRL} = \{x^2 + y, y^2 + x\}$$

The leading terms are $\mathrm{lt}_{DRL}(\mathcal{G}_{DRL}) = \{x^2, y^2\}$. The canonical basis of the quotient ring $\mathbb{F}_5[x, y]/I$ consists of

$$\mathcal{W} = \{1, y, x, xy\} = \{\omega_1, \omega_2, \omega_3, \omega_4\}$$

The degree of the ideal is $D = 4$. Before starting the algorithm, we initialize the following variables:

- *$L = [y, x]$ is the list of variables ordered by $<_{\texttt{lex}}$*
- *$S = [1]$*
- *$V = [v_1]$, where $v_1 = (1,0,0,0)^T$*
- *$G = \emptyset$ which is the vector of the current basis.*

Now we begin the execution loop, popping the smallest element from L at each step:

1. ***Pop*** $t = y$***:*** *We compute* $NF_{DRL}(y) = y = \omega_2$. *Its vector is* $v = (0, 1, 0, 0)^T$.
   Since v *is linearly independent from* v_1. *We execute the following steps:*
   (a) *We update* $S = [1, y]$ *and append* v *to* V *as* v_2.
   (b) *We add the the term* y *multiplied by the variables* x, y *to* L *and sort:* $L = [y^2, x, xy]$.

2. ***Pop*** $t = y^2$***:*** *We compute* $NF_{DRL}(y^2) = -x = -\omega_3$. *Its vector is* $v = (0, 0, -1, 0)^T$.
   This is independent from $\{v_1, v_2\}$. *We update* $S = [1, y, y^2]$ *and append* v *to* V *as* v_3. *We add multiples of* y^2 *to* L*:* $L = [y^3, x, xy, xy^2]$.

3. ***Pop*** $t = y^3$***:*** *We compute* $NF_{DRL}(y^3) = NF_{DRL}(y(-x)) = -xy = -\omega_4$. *Its vector is* $v = (0, 0, 0, -1)^T$.
   This is also independent. We update $S = [1, y, y^2, y^3]$ *and append* v *to* V *as* v_4. *We update* $L = [y^4, x, xy, xy^2, xy^3]$.

4. ***Pop*** $t = y^4$***:*** *We compute* $NF_{DRL}(y^4) = NF_{DRL}(x^2) = -y = -\omega_2$.
   The corresponding vector is $v = (0, -1, 0, 0)^T$.
   Notice that $v = -v_2$. *Once we have found a linear dependency, we obtained the relation* $v + v_2 = 0$ *and we add this to the vector of the current basis* G
   We add it to the basis: $G = \{y^4 + y\}$. *We remove all multiples of the leading term* $\mathrm{lt}_{\mathtt{lex}}(y^4 + y) = y^4$ *from* L*, leaving* $L = [x, xy, xy^2, , xy^3]$.

5. ***Pop*** $t = x$***:*** *We compute* $NF_{DRL}(x) = x = \omega_3$.
   Its vector is $v = (0, 0, 1, 0)^T$. *We see that* $v = -v_3$.
   This gives the relation $x = -y^2$*, translating to* $g_2 = x + y^2$.
   We add it to the basis: $G = \{y^4 + y, x + y^2\}$. *We remove all multiples of* x *from* L. *Since all remaining elements are multiples of* x*,* L *becomes empty.*

The algorithm terminates, returning the reduced lexicographical Gröbner basis:

$$\mathcal{G}_{\texttt{lex}} = \{x + y^2, y^4 + y\}.$$

We observe that the first cycle of the algorithm (from 1. to 4.) replicates exactly the steps explained in Example 4.10 in order to obtain an univariate polynomial in the smaller variable w.r.t. `lex`*. The last cycle (step 5.) allows us to find the polynomial* $x + p(y) = x + y^2$ *as expected by the Shape Lemma.*

If the ideal is zero-dimensional, computing a Gröbner basis with respect to the `lex` order directly is often computationally expensive. In practice, it is generically faster to first compute a Gröbner basis with respect to the `degrevlex` order and then use the FGLM algorithm to convert it into a `lex` basis. In what follows, we will discuss the linear algebra methods used for computing these initial Gröbner bases.

4.2 Macaulay matrices

The transition to linear algebra is achieved through Macaulay matrices: the core idea is to view the systems as coefficient vectors of monomials up to a certain degree.

Let $\mathcal{F} = \{f_1, \ldots, f_t\} \subset \mathbb{F}[x_1, \ldots, x_n]$ be a system of polynomials and fix a term order $\succ$. For a fixed degree d, the Macaulay matrix $\mathcal{M}_{\leq d}$ of $\mathcal{F}$ is constructed by considering all possible polynomials of the form $s \cdot f_i$, where s is a monomial such that $\deg(s \cdot f_i) \leq d$. The columns of $\mathcal{M}_{\leq d}$ are indexed by all monomials m_j of degree $\leq d$ ordered decreasingly with respect to $\succ$. The rows are indexed by the polynomials $s \cdot f_i$. The entry in the row corresponding to $s \cdot f_i$ and the column corresponding to m_j is defined as the coefficient of the monomial m_j in the polynomial $s \cdot f_i$.

We introduce first an example. Consider the polynomial ring $\mathbb{F}_5[x, y]$ equipped with the degree reverse lexicographic order where $x \succ y$. Let

$\mathcal{F} = \{f_1, f_2\}$ be a system of polynomials defined as:

$$f_1 = 2x^2 + y,$$
$$f_2 = 3xy + 1.$$

We want to construct the Macaulay matrix $\mathcal{M}_{\leq 3}$ for the fixed degree $d = 3$. First, we list all monomials in $\mathbb{F}_5[x, y]$ of degree ≤ 3 in decreasing order w.r.t. $\succ_{\texttt{degrevlex}}$:

$$x^3 \succ x^2y \succ xy^2 \succ y^3 \succ x^2 \succ xy \succ y^2 \succ x \succ y \succ 1.$$

These monomials correspond to the columns of $\mathcal{M}_{\leq 3}$. We determine the rows by multiplying f_1 and f_2 by all monomials s such that $\deg(s \cdot f_i) \leq 3$. We obtain

$$x \cdot f_1 = 2x^3 + xy,$$
$$y \cdot f_1 = 2x^2y + y^2,$$
$$x \cdot f_2 = 3x^2y + x,$$
$$y \cdot f_2 = 3xy^2 + y,$$
$$1 \cdot f_1 = 2x^2 + y,$$
$$1 \cdot f_2 = 3xy + 1.$$

By extracting the coefficients of these polynomials, we obtain the Macaulay matrix

$$\mathcal{M}_{\leq 3} = \begin{array}{c} \begin{array}{cccccccccc} x^3 & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{c} xf_1 \\ yf_1 \\ xf_2 \\ yf_2 \\ f_1 \\ f_2 \end{array}$$

In the next example, we show how Macaulay matrices allow us to compute a Gröbner basis of $I = (f_1, \ldots, f_t)$.

4.12 Example

Consider the system over $\mathbb{F}_5[x, y]$ *equipped with the* `degrevlex` *order* ($x \succ y$). *Let* $\mathcal{F} = \{f_1, f_2\}$ *be defined as:*

$$f_1 = x^2 + 2xy + y,$$
$$f_2 = x^2 + y^2 + 2.$$

and let $I = (f_1, f_2)$.

We begin by constructing the Macaulay matrix at the maximum degree of the initial generators $d = 2$. *The columns are indexed by the monomials of degree* ≤ 2*:*

$$x^2 \succ xy \succ y^2 \succ x \succ y \succ 1.$$

The Macaulay matrix $\mathcal{M}_{\leq 2}$ *corresponds to:*

$$\mathcal{M}_{\leq 2} = \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 1 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 2 \end{pmatrix} \end{array} \begin{array}{l} f_1 \\ f_2 \end{array}$$

We compute now the Reduced Row Echelon Form (RREF) of $\mathcal{M}_{\leq 2}$. *Subtracting the first row from the second yields the new vector*

$$(1, 0, 1, 0, 0, 2) - (1, 2, 0, 0, 1, 0) = (0, -2, 1, 0, -1, 2) \equiv (0, 3, 1, 0, 4, 2) \pmod 5$$

We observe that this operation corresponds exactly to the computation of the S-polynomial $S(f_1, f_2) = f_2 - f_1$. *The RREF of* $\mathcal{M}_{\leq 2}$ *corresponds to*

$$\tilde{\mathcal{M}}_{\leq 2} = \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 2 & 0 & 3 & 4 \end{pmatrix} \end{array} \begin{array}{c} x^2 + y^2 + 2 \\ xy + 2y^2 + 3y + 4 \end{array}$$

We now consider the polynomials $f_1, f_2 - f_1$ *corresponding to the rows of the RREF* $\tilde{\mathcal{M}}_{\leq 2}$. *These polynomials do not constitute a Gröbner basis of* I

and we repeat the process incrementing the degree to $d = 3$. *Consider the Macaulay matrix* $\mathcal{M}_{\leq 3}$. *The monomials indexing the columns are:*

$$x^3 \succ x^2y \succ xy^2 \succ y^3 \succ x^2 \succ xy \succ y^2 \succ x \succ y \succ 1.$$

The polynomials indexing the rows are:

- $x \cdot f_1 = x^3 + 2x^2y + xy$,
- $y \cdot f_1 = x^2y + 2xy^2 + y^2$,
- $x \cdot f_2 = x^3 + xy^2 + 2x$,
- $y \cdot f_2 = x^2y + y^3 + 2y$,
- $1 \cdot f_1 = x^2 + 2xy + y$,
- $1 \cdot f_2 = x^2 + y^2 + 2$.

The matrix $\mathcal{M}_{\leq 3}$ *corresponds to:*

$$\mathcal{M}_{\leq 3} = \begin{array}{c} \begin{array}{cccccccccc} x^3 & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 2 \end{pmatrix} \end{array} \begin{array}{c} xf_1 \\ yf_1 \\ xf_2 \\ yf_2 \\ f_1 \\ f_2 \end{array}$$

We compute now the Reduced Row Echelon Form (RREF) of $\mathcal{M}_{\leq 3}$:

$$\tilde{\mathcal{M}}_{\leq 3} = \begin{array}{c} \begin{array}{cccccccccc} x^3 & x^2y & xy^2 & y^3 & x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 3 & 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 4 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 2 & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ x^3 + 3y^3 + x + 2y + 3 \\ x^2y + y^3 + 2y \\ xy^2 + 2y^3 + x + 3y + 2 \\ x^2 + 2x + 3y + 1 \\ xy + 4x + 4y + 2 \\ y^2 + 3x + 2y + 1 \end{array}$$

The polynomials $f_1, xy + 4x + 4y + 2, y^2 + 3x + 2y + 1$ *corresponding to the three bottom rows of* $\tilde{\mathcal{M}}_{\leq 3}$ *form a Gröbner basis for the original ideal* I*.*

In general, if we would not have found a Gröbner basis at this step, we should have added to $\mathcal{M}_{\leq 3}$ new rows corresponding to the polynomials $x^\alpha f_j$ where f_j is a polynomial corresponding to a new row in $\tilde{\mathcal{M}}_{\leq 3}$ satisfying $\deg(f_j) < 3$ and x^α is a monomial such that $\deg(x^\alpha f_j) \leq 3$. After this we should have computed the RREF of the new Macaulay matrix. This example illustrates the underlying logic of the Lazard algorithm [Laz83]. Note that to check whether the obtained polynomials form a Gröbner basis, we must rely on other criteria, such as Buchberger's criterion, or have a theoretical justification for the degree d at which we can stop computing Macaulay matrices.

However, as n and d increase, the global Macaulay matrix $\mathcal{M}_{\leq d}$ becomes exponentially large and with a massive amount of zero entries, making this method computationally expensive.

4.3 The F_4 Algorithm

To overcome the problem of the global Macaulay matrix, J. C. Faugère [Fau99] introduced the F_4 algorithm in 1999. Fix a degree-compatible term order. The idea of F_4 is to repeat the process described for Macaulay matrices,

but locally, by constructing submatrices for specific monomials. To this end, we assume we have a selection function Sel which, given a list of pairs P, returns a non-empty subset $P' \subseteq P$. The efficiency of the algorithm depends on the choice of this function. A standard approach is the *normal selection strategy*, which corresponds to restricting B' to the set of all pairs with the minimal degree in P (the degree of a pair is defined as the degree of the least common multiple of their leading term). Starting from the set of polynomials $\mathcal{F} = \{f_1, \ldots, f_t\}$ we define the set of all indices pairs $B \coloneqq \{(i,j) \mid 1 \leq i < j \leq t\}$ and we select a subset $B' = Sel(B)$. Then the algorithm starts two phases: a symbolic preprocessing and a matrix reduction phase.

First, we construct the initial set of polynomials to be reduced. For each pair $(f_i, f_j) \in B'$, we consider the two "halves" of the corresponding S-polynomial:

$$s_{\text{half}}(f_i, f_j) \coloneqq \frac{\text{lcm}(\text{lt}(f_i), \text{lt}(f_j))}{\text{lt}(f_i)} f_i \quad \text{and} \quad s_{\text{half}}(f_j, f_i) \coloneqq \frac{\text{lcm}(\text{lt}(f_i), \text{lt}(f_j))}{\text{lt}(f_j)} f_j.$$

and we create a list of these polynomials

$$L = \{(s_{\text{half}}(f_i, f_j), s_{\text{half}}(f_j, f_i)) \mid (f_i, f_j) \in B'\}.$$

The goal of the symbolic preprocessing phase is to predict the polynomials that will be required for reductions in the linear algebra operations. The phase is described as follows.

Let M be the set of all monomials appearing in the the polynomials in L. For each monomial $m \in M$, the algorithm checks if m is divisible by the leading term $\text{lt}(g)$ of some polynomial $g \in G$. If such a g exists, then, if not yet included, we add the polynomial $x^\alpha g$ to L where $x^\alpha \text{lt}(g) = m$. Now, we add to M the monomials appearing also in this new polynomial. This process is repeated until we cannot find no more monomial in M satisfying the property.

After the symbolic preprocessing phase, we construct a local Macaulay matrix $\mathcal{M}$. The rows of $\mathcal{M}$ correspond to the polynomials in the new expanded set L. The columns correspond to all the distinct monomials in M ordered decreasingly w.r.t. the term order $\succ$.

We compute now the RREF of $\mathcal{M}$, denoted as $\tilde{\mathcal{M}}$. Consider now the non-zero rows of $\tilde{\mathcal{M}}$ whose leading monomials were not leading monomials of any element in L. The polynomials corresponding to these rows are added to the current generating set G. The list of pairs P is updated with the new pairs formed by these newly discovered elements and the algorithm proceeds iteratively. A proof of correctness and termination of this algorithm can be found in [CLO15, Chapter 10, Section 3].

4.13 Example

Consider again the ideal $I = (f_1, f_2) \subset \mathbb{F}_5[x, y]$ equipped with the `degrevlex` *order ($x \succ y$), where $f_1 = x^2 - y$ and $f_2 = xy - x$. We perform a single step of the F_4 algorithm for the pair (f_1, f_2).*

The set L of "halves" of S-polynomials is:

$$L = \left\{ yf_1 = x^2y - y^2, \quad xf_2 = x^2y - x^2 \right\}.$$

The set of monomials appearing in L is $M = \{x^2y, x^2, y^2\}$.

Consider now the symbolic preprocessing phase. We skip the monomial x^2y since by construction yf_1 and xf_2 are already in L. The monomial x^2 is divisible by $\mathrm{lt}(f_1) = x^2$. Since the corresponding f_1 is not in L, we add it. The polynomial f_1 forces to add the monomial y to M. The remaining terms y^2 and y are not divisible by $\mathrm{lt}(f_1)$ and $\mathrm{lt}(f_2)$. The phase terminates, yielding

$$L = \{yf_1, xf_2, f_1\}, \quad M = \{x^2y, x^2, y^2, y\}.$$

Next, we construct the local Macaulay matrix $\mathcal{M}$. The columns correspond to the monomials in M sorted decreasingly. The rows correspond to the

polynomials in L:

$$\mathcal{M} = \begin{array}{c} \begin{matrix} x^2y & x^2 & y^2 & y \end{matrix} \\ \begin{pmatrix} 1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \end{array} \begin{matrix} yf_1 \\ xf_2 \\ f_1 \end{matrix}$$

We compute the RREF:

$$\widetilde{\mathcal{M}} = \begin{array}{c} \begin{matrix} x^2y & x^2 & y^2 & y \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \begin{matrix} x^2y - y \\ x^2 - y \\ y^2 - y \end{matrix}$$

The leading monomials of the polynomials in L are x^2y and x^2. The polynomial corresponding to the third row has a new leading term y^2.

Consequently, the polynomial $y^2 - y$ is added to G and the algorithm should proceed to the next iteration.

During the computation of F_4, we generate polynomials of different degrees. This motivates the following definition.

4.14 Definition (Solving degree)
Let F be a system of multivariate polynomials and fix a degree-compatible term order $\prec$ and a selection function Sel. The ***solving degree*** *of F with respect to the F_4 algorithm with Sel and $\prec$ is defined as the maximum degree among all polynomials generated during the entire Gröbner basis computation.*

This concept is of crucial importance in algebraic cryptanalysis because it is an indicator of the computational complexity of the Gröbner basis extraction through the F_4 algorithm. For instance, the columns in the local

Macaulay matrices are indexed by the monomials appearing during the computation and the solving degree represents the maximum degree reached by these monomials.

4.15 Example

Let us reconsider the computation performed in the previous example. During the symbolic preprocessing phase, the algorithm constructed the polynomial $yf_1 = x^2y - y^2$*, which has total degree* 3*. Consequently, the local Macaulay matrix* $\mathcal{M}$ *contains a column corresponding to the monomial* x^2y *of degree* 3*. If the entire* F_4 *computation terminates without generating any polynomial of degree* 4 *or higher, we conclude that the solving degree of the system* $\mathcal{F} = \{f_1, f_2\}$ *is exactly* 3*.*

5 Algebraic Cryptanalysis: The MaxMinors modeling

Cryptanalysis is the study of mathematical techniques used to evaluate and compromise the security of cryptographic systems. In our setting, the main aspects that need to be considered are the structural security of the signature scheme and the complexity analysis of the RSD_s problem. In this chapter we will focus on the latter. The security analysis of problems arising from coding theory splits into two branches: combinatorial and algebraic attacks. Combinatorial attacks are designed to exploit the structural and algebraic properties of the underlying code rather than its algebraic relations. Algebraic attacks, instead, focus on modeling the problem as a multivariate polynomial system and subsequently computing or estimating the complexity of solving it. The complexity of these attacks relies heavily on the chosen modeling. Consequently a primary objective in this field is to find a representation that best fits the algebraic structure of the problem. Specifically, we will focus on an algebraic attack against RSD_s based on the MaxMinors modeling, an approach introduced by Bardet et al. in [Bar+20].

5.1 The MaxMinors system

Let us first recall our setting. Let $(H, y) \in \mathrm{Mat}_{(n-k)\times n}(\mathbb{F}_{q^m}) \times \mathbb{F}_{q^m}^{n-k}$ be an RSD_s instance, let $s = (1 \parallel s') \in \mathbb{F}_{q^m}^r$ be a vector whose r coordinates are linearly independent over $\mathbb{F}_q$ and $C = [I_r \parallel C'] \in \mathrm{Mat}_{r\times n}(\mathbb{F}_q)$ be a matrix satisfying

$$sCH^\top = y.$$

Our primary goal is to derive a homogeneous (with syndrome 0) RSD_s instance, whose solution corresponds exactly to the solution of the original RSD_s instance. For this purpose we rely on an assumption that is clearly satisfied by the RYDE parameter sets. As discussed in Remark 2.15 and

Proposition 2.17, the small value $4q^{r(m+n-r)-m(n-k)}$ allows us to guarantee the uniqueness of the RSD_s solution and to assume that the linear code defined by H contains no codewords of rank exactly r.

Assume without loss of generality that $y_{n-k} \neq 0$. We construct a new parity-check matrix $\tilde{H}$ as follows. Let $h_i \in \mathbb{F}_{q^m}^n$ denote the rows of H for $i = 1, \ldots, n-k$. We define the rows of $\tilde{H}$ as

$$\tilde{h}_i \coloneqq h_i y_{n-k} - h_{n-k} y_i$$

for $i = 1, \ldots, n-k-1$. Since $y_{n-k} \neq 0$, a linear dependence between the rows of $\tilde{H}$ automatically yields a linear dependence between the rows of H. This justifies that $\tilde{H}$ is indeed a parity-check matrix. Note that $\tilde{H}$ can be constructed by replacing the $n-k$ index with any index j such that $y_j \neq 0$.

The construction of $\tilde{H}$ is equivalent to considering the linear code $\tilde{\mathcal{C}}$ over $\mathbb{F}_{q^m}$ defined by adding the error vector sC yielding the correct syndrome y to the code $\mathcal{C}$ corresponding to the parity-check matrix H. Indeed it is clear that $\mathcal{C} = \ker L_H \subseteq \ker L_{\tilde{H}}$ and, by construction, the error vector sC is contained in the kernel $\ker L_{\tilde{H}}$. Analyzing the dimensions yields $k+1 \leq \dim \tilde{\mathcal{C}} \leq \dim \ker L_{\tilde{H}} = k+1$. Hence $\ker L_{\tilde{H}} = \tilde{\mathcal{C}} = \mathcal{C} + \langle sC \rangle_{\mathbb{F}_{q^m}}$.

Any codeword $z \in \tilde{\mathcal{C}}$ can be expressed as the sum of a codeword from the original code $\mathcal{C}$ and an element of the form $a \cdot sC$, where $a \in \mathbb{F}_{q^m}$. Consequently, multiplying by $H^\top$ yields $zH^\top = a \cdot sCH^\top = a \cdot y$. Suppose that $z = c + a \cdot sC$ is a valid rank r solution with $c \in \mathcal{C}$. We distinguish two cases: if $a = 0$, then $\mathcal{C}$ admits a codeword of rank r; if $a \neq 0$, $a^{-1}z$ is another solution of the RSD_s instance. The assumption allows us to conclude that $z = asC$. Finally, the additional restriction that the error vector must have 1 in its first coordinate forces $a = 1$. A direct consequence is that sC is a solution to the RSD_s instance (H, y) if and only if it is a solution to the homogeneous RSD_s instance $(\tilde{H}, 0)$.

The problem thus reduces to finding (s', C') in the previously described form such that

$$(1 \parallel s')[I_r \parallel C']\tilde{H}^\top = 0_{n-k-1}.$$

This condition implies that a non-trivial linear combination of the r rows of the matrix $[I_r \parallel C']\tilde{H}^\top \in \mathrm{Mat}_{r\times n-k-1}$ evaluates to zero. Consequently all its minors of size r must vanish. The core idea of this system is to express the minors of the matrix $[I_r \parallel C']\tilde{H}^\top$ as a combination between the minors of C and the minors of $\tilde{H}$. The following proposition details this aspect.

5.1 Proposition (Generalized Cauchy-Binet Formula)
Let $\mathbb{F}$ be a field, let n and r be two positive integers with $n \geq r$ and let $A \in \mathrm{Mat}_{r\times n}(\mathbb{F})$ and $B \in \mathrm{Mat}_{n\times r}(\mathbb{F})$ be two matrices over $\mathbb{F}$. Then the following formula holds:

$$\det(AB) = \sum_{T\subset\{1,\ldots,n\},|T|=r} \det(A|_{*,T}) \det(B|_{T,*})$$

where $A|_{*,T}$ denote the square submatrix of A whose rows are indexed by T and analogously $B|_{T,*}$ denote the square submatrix of B whose rows are indexed by T.

In Appendix A, we present a proof of this proposition, introducing the necessary theory of exterior products and determinants.

Returning to the modeling, we first observe that $\tilde{H}^\top$ is an $n \times n-k-1$ matrix. In order to apply directly the proposition, we select the submatrices $\tilde{H}^\top|_{*,J}$ indexed by a subset $J \subset \{1,\ldots n-k-1\}$ of size r and we observe that the product with the matrix $[I_r \parallel C']$ restricts to this submatrix. Applying the formula we obtain $\binom{n-k-1}{r}$ equations over $\mathbb{F}_{q^m}$.

For every subset $T \subset \{1,\ldots,n\}$ of size r, we denote by X_T the determinant of the submatrix $[I_r \parallel C']|_{*,T}$. Note that once we obtain the value of the minors X_T for all T, we can easily recover the coordinates of the matrix $[I_r \parallel C']$ by considering the minors indexed by $r-1$ columns of I_r. The equations derived from this modeling are given by:

$$\sum_{\substack{T\subset\{1,\ldots,n\}\\|T|=r}} X_T \det(\tilde{H}^\top|_{T,J}) = 0 \quad \text{for any subset } J \subset \{1,\ldots,n-k-1\},\ |J| = r.$$

By unfolding these equation over $\mathbb{F}_q$, we obtain a total of $m\binom{n-k-1}{r}$ and, using the X_T as variables, we obtain a linear system over $\mathbb{F}_q$ with a total number of variables $\binom{n}{r} - 1$. The -1 term arises because $c_{\{1,\dots,r\}} = \det(I_r) = 1$. We distinguish two cases: the overdetermined case, where $m\binom{n-k-1}{r} \geq \binom{n}{r} - 1$, and the underdetermined case, where the strict reverse inequality holds. In the first case, the system has at most one solution, which can be recovered using Gaussian elimination. In the underdetermined case, which is the scenario of interest for the RYDE parameter sets, the linear system has multiple solutions. Before detailing the approaches that we have considered to solve the system, consider the following toy example.

5.2 Remark

Note that until now, our goal has been to find the matrix $C = [I_r \parallel C']$. Once we have recovered the matrix C, we can obtain $s = (1 \parallel s')$ by simply solving the linear system $sCH = y$ which is guaranteed to have a unique solution under the assumptions that $r \leq \frac{m+n}{2}$ and that the quantity $4q^{r(m+n-r)-m(n-k)}$ is small. Indeed, this quantity is an increasing function of r for $r < \frac{m+n}{2}$, a condition that holds for RYDE-1 parameters ($n = 53$, $m = 53$, $r = 4$). Consequently, Remark 2.15 allows us to derive that the probability to obtain a codeword of rank $\leq r$ is small. To see why this guarantees the uniqueness of s, suppose we have two distinct vectors s_1 and s_2 with the same coefficient matrix $C = [I_r \parallel C']$ and yielding the same syndrome y. Since $H(s_1C)^\top = y = H(s_2C)^\top$, the vector $(s_1 - s_2)C$ must belong to the code $\mathcal{C} = \ker(L_H)$. Moreover, we can express each of its coordinates as a linear combination of its first r ones. This implies the existence of a non-zero codeword of rank $\leq r$, which contradicts our previous probability assumption.

Moreover, any other possible matrix $\tilde{C} = [I_r \parallel C']$ for which $\tilde{C}\tilde{H}^\top$ does not have full rank would automatically yield an RSD_s solution of rank at most r. As described above, this occurs with probability bounded by $4q^{r(m+n-r)-m(n-k)}$.

5.3 Example

Consider the following parameters $m = 3, r = 2, n = 6$ and $q = 2$. Note that

the aforementioned parameters do not satisfy the condition in Remark 2.15 and, hence we could not guarantee the theoretical overwhelming probability of uniqueness of the RSD_s solution. Indeed, for small parameters, having both the conditions for uniqueness condition and underdetermined equations is rare. For this reason we will assume only the latter. View $\mathbb{F}_{2^3}$ as the quotient field $\mathbb{F}_2[x]/(x^3+x+1)$ and define $a \coloneqq \bar{x}$. Let the RSD_s instance be constructed as follows:

$$y = (0, a+1, 1, a+1).$$

and

$$H = \begin{bmatrix} a+1 & 1 & a^2+1 & a^2+1 & a^2+a & 0 \\ a^2 & a^2+a & a & a & a^2 & 1 \\ a & a^2+a & a & a^2+a+1 & a^2+a+1 & 0 \\ 0 & a & 0 & a^2 & 0 & 0 \end{bmatrix}.$$

We construct the matrix $\widetilde{H}$ as described at the start of this section:

$$\widetilde{H} = \begin{bmatrix} a^2+1 & a+1 & a^2 & a^2 & 1 & 0 \\ a^2+a+1 & a^2+a+1 & a^2+a & 1 & a^2+a+1 & a+1 \\ a^2+a & a+1 & a^2+a & a^2+a & a & 0 \end{bmatrix}$$

and the transpose

$$\widetilde{H}^T = \begin{bmatrix} a^2+1 & a^2+a+1 & a^2+a \\ a+1 & a^2+a+1 & a+1 \\ a^2 & a^2+a & a^2+a \\ a^2 & 1 & a^2+a \\ 1 & a^2+a+1 & a \\ 0 & a+1 & 0 \end{bmatrix}$$

From the latter matrix and the Cauchy Binet Generalized formula (Prop. A.3) we derive the following linear equations over $\mathbb{F}_{q^m}$

$$\begin{aligned} 0 = {} & a^2x_{0,1} + ax_{0,2} + a^2x_{0,3} + x_{0,4} + a^2x_{0,5} + ax_{1,3} + (a^2+1)x_{1,4} + (a^2+1)x_{1,5} \\ & + x_{2,3} + (a^2+a+1)x_{2,4} + (a^2+a+1)x_{2,5} + (a^2+a+1)x_{3,5} + (a+1)x_{4,5}, \end{aligned}$$

$$0 = (a^2+1)x_{0,1} + (a^2+a)x_{0,2} + (a^2+a)x_{0,3} + (a^2+a+1)x_{0,4} + (a^2+a)x_{1,2} + (a^2+a)x_{1,3} + (a^2+1)x_{1,4} + (a^2+1)x_{2,4} + (a^2+1)x_{3,4},$$

$$0 = (a^2+a)x_{0,1} + (a^2+a)x_{0,2} + ax_{0,3} + x_{0,4} + x_{0,5} + (a^2+1)x_{1,2} + (a^2+a+1)x_{1,3} + (a^2+a+1)x_{1,4} + (a^2+1)x_{1,5} + a^2x_{2,3} + (a+1)x_{2,4} + x_{2,5} + (a^2+a)x_{3,4} + x_{3,5} + (a^2+a)x_{4,5}.$$

where with $x_{i,j}$ we denote the minors of C (that is a unknown matrix 2×6 over $\mathbb{F}_q$) indexed by the columns $i+1$ and $j+1$. We can unfold these equations over $\mathbb{F}_q$ and obtain the following:

$$\begin{cases} x_{0,4} + x_{1,4} + x_{1,5} + x_{2,3} + x_{2,4} + x_{2,5} + x_{3,5} + x_{4,5} = 0 \\ x_{0,2} + x_{1,3} + x_{2,4} + x_{2,5} + x_{3,5} + x_{4,5} = 0 \\ x_{0,1} + x_{0,3} + x_{0,5} + x_{1,4} + x_{1,5} + x_{2,4} + x_{2,5} + x_{3,5} = 0 \\ x_{0,1} + x_{0,4} + x_{1,4} + x_{2,4} + x_{3,4} = 0 \\ x_{0,2} + x_{0,3} + x_{0,4} + x_{1,2} + x_{1,3} = 0 \\ x_{0,1} + x_{0,2} + x_{0,3} + x_{0,4} + x_{1,2} + x_{1,3} + x_{1,4} + x_{2,4} + x_{3,4} = 0 \\ x_{0,4} + x_{0,5} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{2,4} + x_{2,5} + x_{3,5} = 0 \\ x_{0,1} + x_{0,2} + x_{0,3} + x_{1,3} + x_{1,4} + x_{2,4} + x_{3,4} + x_{4,5} = 0 \\ x_{0,1} + x_{0,2} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{2,3} + x_{3,4} + x_{4,5} = 0 \end{cases}$$

One can resolve these linear system and find a basis for the solutions of the system above

$$\begin{aligned} v_0 &= (1,1,0,0,0,0,1,0,1,1,0,0,1,0,0), \\ v_1 &= (0,0,1,0,0,0,1,0,0,1,0,0,0,1,0), \\ v_2 &= (0,0,0,0,1,0,0,0,1,1,0,0,0,0,0), \\ v_3 &= (0,0,0,0,0,1,1,0,0,1,0,0,0,0,1), \\ v_4 &= (0,0,0,0,0,0,0,1,1,0,1,0,0,1,0), \\ v_5 &= (0,0,0,0,0,0,0,0,0,0,0,1,0,1,0) \end{aligned}$$

whose coordinates correspond, in order, to the sequence of minors:

$$(x_{0,1},\ x_{0,2},\ x_{0,3},\ x_{0,4},\ x_{0,5},\ x_{1,2},\ x_{1,3},\ x_{1,4},\ x_{1,5},\ x_{2,3},\ x_{2,4},\ x_{2,5},\ x_{3,4},\ x_{3,5},\ x_{4,5}).$$

The minors of a possible matrix $C = [I_r \parallel C']$ being a solution of the RSD_s instance (H, y) must be a linear combination over $\mathbb{F}_2$ of the vectors $v_1, \ldots, v_6$. For instance one can verify that

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } s = (1,\ a^2 + 1)$$

form a solution to the instance (H, y) and the minors of C corresponds to the linear combination $v_1 + v_2 + v_5$.

5.1.1 SageMath code for constructing the MaxMinors system

We also present the SageMath [Ste+26] code used to construct the aforementioned linear system. Assuming the parameters q, m, n, and r have already been initialized, we first detail the algorithm that takes a polynomial over $\mathbb{F}_{q^m}$ and returns the corresponding list of m polynomials over the base field $\mathbb{F}_q$.

Listing 1: Unfolding algorithm for polinomials over $\mathbb{F}_{q^m}$

```
from sage.all import *
import itertools
Fq = GF(q)
Fqm = GF(q**m, 'a')
a = Fqm.gen()
def unfold_polynomial(P, m):
        """
        Takes a polynomial P over F_{q^m} and returns a
           list of m polynomials over F_q
        """
        if P == 0:
```

```
11                  return [0] * m
12          unfolded = [0] * m
13          monomi = P.monomials()
14          coefficienti = P.coefficients()
15          for c, mon in zip(coefficienti, monomi):
16                  for i in range(m):
17                          if c[i] != 0:
18                          unfolded[i] += c[i] * mon
19          return unfolded
```

Assuming we have already defined the matrix H and the syndrome y, we present the implementation for constructing the MaxMinors system. Lines 1–11 perform the initial setup, while Lines 13–21 generate the list of polynomials over the extension field $\mathbb{F}_{q^m}$. Finally, these polynomials are unfolded over $\mathbb{F}_q$ in Lines 23–27.

Listing 2: Generation of the MaxMinors system

```
1  subsets_T = list(itertools.combinations(range(n), r))
2  subsets_J = list(itertools.combinations(range(n - k - 1),
       r))
3  name_minors = [f"x_{'_'.join(map(str, T))}" for T in
       subsets_T]
4  R_linear = PolynomialRing(Fq, name_minors, order='
       degrevlex')
5  R_linear_ext = PolynomialRing(Fqm, name_minors, order='
       degrevlex')
6  variables_x = R_linear.gens()
7  pivot_idx = next(i for i in reversed(range(n-k)) if y[i]
       != 0)
8  Htilde_rows = [H[idx] * y[pivot_idx] - H[pivot_idx] * y[
       idx]
9  for idx in range(n-k) if idx != pivot_idx]
10 Htilde = matrix(Htilde_rows)
```

```
H_tilde_T = Htilde.transpose()

eqs_linear = []
for J in subsets_J:
        eq = R_linear_ext(0)
        for index_T, T in enumerate(subsets_T):
                H_sub = H_tilde_T.
                   matrix_from_rows_and_columns(list(T),
                   list(J))
                det_sub = H_sub.det()
                if det_sub != 0:
                        eq += det_sub * variables_x[
                           index_T]
        eqs_linear.append(eq)

eqs_linear_unf = []
for p in eqs_linear:
        list_eq_unf = unfold_polynomial(p, m)
        eqs_linear_unf.extend(list_eq_unf)
eqs_linear_polSeq = [R_linear(eq) for eq in
   eqs_linear_unf]

system = Sequence(eqs_linear_polSeq)
```

In what follows, we consider two approaches to find a solution matrix C':

1. A direct approach, substituting the coordinates of C' directly into the minors.

2. Solving the linear system and substituting the parameterized solutions (expressed as a linear combination of the kernel vectors of the system) into the relations that the minors must satisfy, known as Plücker relations.

5.2 The direct approach

In this subsection we detail how the direct approach is constructed. Consider the matrix $C' = (c_{i,j})_{i\in\{1,\ldots,r\},j\in\{r+1,\ldots,n\}}$, where $c_{i,j}$ are variables taking values in $\mathbb{F}_q$. The direct approach consists of substituting the variable X_T in the MaxMinors system with the minor of the submatrix of $[I_r \parallel C']$ whose columns are indexed by T. Hence, we obtain a new system having as variables the coordinates of C. The polynomials in the resulting system have generally degree r and our goal is to estimate the computational complexity of solving this system using the F_4 algorithm. In particular, we will compare the resolution of this system with and without field equations and compare with the second modeling that will be presented. Before proceeding, we discuss the direct approach in the previous example.

5.4 Example

Consider again the system

$$\begin{cases} x_{0,4}+x_{1,4}+x_{1,5}+x_{2,3}+x_{2,4}+x_{2,5}+x_{3,5}+x_{4,5}=0 \\ x_{0,2}+x_{1,3}+x_{2,4}+x_{2,5}+x_{3,5}+x_{4,5}=0 \\ x_{0,1}+x_{0,3}+x_{0,5}+x_{1,4}+x_{1,5}+x_{2,4}+x_{2,5}+x_{3,5}=0 \\ x_{0,1}+x_{0,4}+x_{1,4}+x_{2,4}+x_{3,4}=0 \\ x_{0,2}+x_{0,3}+x_{0,4}+x_{1,2}+x_{1,3}=0 \\ x_{0,1}+x_{0,2}+x_{0,3}+x_{0,4}+x_{1,2}+x_{1,3}+x_{1,4}+x_{2,4}+x_{3,4}=0 \\ x_{0,4}+x_{0,5}+x_{1,2}+x_{1,3}+x_{1,4}+x_{1,5}+x_{2,4}+x_{2,5}+x_{3,5}=0 \\ x_{0,1}+x_{0,2}+x_{0,3}+x_{1,3}+x_{1,4}+x_{2,4}+x_{3,4}+x_{4,5}=0 \\ x_{0,1}+x_{0,2}+x_{1,2}+x_{1,3}+x_{1,4}+x_{1,5}+x_{2,3}+x_{3,4}+x_{4,5}=0 \end{cases}$$

We operate the substitution with the entries $c_{i,j}$ and we obtain:

$$\begin{cases} c_{0,3}c_{1,2} + c_{0,4}c_{1,2} + c_{0,5}c_{1,2} + c_{0,2}c_{1,3} + c_{0,5}c_{1,3} \\ \quad + c_{0,2}c_{1,4} + c_{0,5}c_{1,4} + c_{0,2}c_{1,5} + c_{0,3}c_{1,5} \\ \quad + c_{0,4}c_{1,5} + c_{0,4} + c_{0,5} + c_{1,4} = 0 \\ c_{0,4}c_{1,2} + c_{0,5}c_{1,2} + c_{0,5}c_{1,3} + c_{0,2}c_{1,4} + c_{0,5}c_{1,4} + c_{0,2}c_{1,5} + c_{0,3}c_{1,5} + c_{0,4}c_{1,5} + c_{0,3} + c_{1,2} = 0 \\ c_{0,4}c_{1,2} + c_{0,5}c_{1,2} + c_{0,5}c_{1,3} + c_{0,2}c_{1,4} + c_{0,2}c_{1,5} + c_{0,3}c_{1,5} + c_{0,4} + c_{0,5} + c_{1,3} + c_{1,5} + 1 = 0 \\ c_{0,4}c_{1,2} + c_{0,4}c_{1,3} + c_{0,2}c_{1,4} + c_{0,3}c_{1,4} + c_{0,4} + c_{1,4} + 1 = 0 \\ c_{0,2} + c_{0,3} + c_{1,2} + c_{1,3} + c_{1,4} = 0 \\ c_{0,4}c_{1,2} + c_{0,4}c_{1,3} + c_{0,2}c_{1,4} + c_{0,3}c_{1,4} + c_{0,2} + c_{0,3} + c_{0,4} + c_{1,2} + c_{1,3} + c_{1,4} + 1 = 0 \\ c_{0,4}c_{1,2} + c_{0,5}c_{1,2} + c_{0,5}c_{1,3} + c_{0,2}c_{1,4} + c_{0,2}c_{1,5} + c_{0,3}c_{1,5} + c_{0,2} + c_{0,3} + c_{0,4} + c_{0,5} + c_{1,4} + c_{1,5} = 0 \\ c_{0,4}c_{1,2} + c_{0,4}c_{1,3} + c_{0,2}c_{1,4} + c_{0,3}c_{1,4} + c_{0,5}c_{1,4} + c_{0,4}c_{1,5} + c_{0,3} + c_{0,4} + c_{1,2} + c_{1,3} + 1 = 0 \\ c_{0,3}c_{1,2} + c_{0,2}c_{1,3} + c_{0,4}c_{1,3} + c_{0,3}c_{1,4} + c_{0,5}c_{1,4} + c_{0,4}c_{1,5} + c_{0,2} + c_{0,3} + c_{0,4} + c_{0,5} + c_{1,2} + 1 = 0 \end{cases}$$

Let I be the ideal generated by the polynomials appearing on the left-hand side of the equations in the system above. In Subsection 5.4, we estimate the complexity of computing with F_4 algorithm a Gröbner basis of the ideal I w.r.t. the `degrevlex` *term order. Moreover we extend the estimation for the ideal generated by I together with the field equations:*

$$\begin{aligned} c_{0,2}^2 + c_{0,2} &= 0, \\ c_{0,3}^2 + c_{0,3} &= 0, \\ c_{0,4}^2 + c_{0,4} &= 0, \\ c_{0,5}^2 + c_{0,5} &= 0, \\ c_{1,2}^2 + c_{1,2} &= 0, \\ c_{1,3}^2 + c_{1,3} &= 0, \\ c_{1,4}^2 + c_{1,4} &= 0, \\ c_{1,5}^2 + c_{1,5} &= 0. \end{aligned}$$

5.2.1 SageMath code for direct approach

We describe now the SageMath code to construct the ideal coming from the direct approach. Lines 1–6 perform the setup and lines 8–22 substitute the minors of C in the variables of the MaxMinors system. Finally, lines 25–27 construct the two ideals: I_enriched without field equations and I_enriched_fields_eqs with the field equations.

Listing 3: Generation of the ideals from the direct approach

```
1  name_C_entries = [f'c_{i}_{j}' for i in range(r) for j in
       range(r,n)]
2  R = PolynomialRing(Fq, name_C_entries, order='degrevlex')
3  C_symbolic_primeFq = matrix(R, r, n - r, R.gens())
4  I_rFq = identity_matrix(R, r)
5  C_symbolicFq = I_rFq.augment(C_symbolic_primeFq)
6  vector_minors = C_symbolicFq.minors(r)
7
8  equations_evaluated_unfolded = []
9  for eq in eqs_linear_unf:
10         if eq == 0:
11                 continue
12         new_eq = R(0)
13         for monomial_expon, coeff in eq.dict().items():
14                 term = Fq(coeff) * R(1)
15                 for idx_var, exp in enumerate(
                       monomial_expon):
16                       if exp > 0:
17                               term *= (vector_minors[
                                     idx_var])**exp
18                 new_eq += term
19
20         if new_eq != 0:
21                 equations_evaluated_unfolded.append(
```

```
                    new_eq)
22
23
24  # Field Equations and ideals
25  I_enriched = R.ideal(equations_evaluated_unfolded)
26  field_eqs_Fq = [var**2 - var for var in R.gens()]
27  I_enriched_field_eqs_Fq = R.ideal(
       equations_evaluated_unfolded + field_eqs_Fq)
```

5.3 The Plücker relations approach

In this subsection we present the second approach based on the Plücker relations. Informally, these relations correspond to the conditions that maximal minors of any matrix must satisfy. A formal definition and detailed study of these relations are in Subsection A.2 of the Appendix A.

In particular the approach we will present relies on Theorem A.7. Informally, fixed the dimension of the matrix $n \times r$ we denote as $I_{n,r}$ the ideal generated by the Plücker relations. Specifically, we utilize the following result, which provides an explicit Gröbner basis for $I_{n,r}$ with respect to a certain term order $\prec$:

$$\mathcal{G} = \left\{ \sum_{\substack{\pi \text{ is an} \\ (i,r+1-i)\text{-shuffle}}} \operatorname{sign}(\pi) p_{\pi(\sigma)} p_{\pi(\tau)} \;\middle|\; \sigma, \tau \text{ are incomparable} \right\}.$$

where $\sigma = \{\sigma_1 < \cdots < \sigma_r\}$ and $\tau = \{\tau_1 < \cdots < \tau_r\}$ are ordered subsets of $\{1, \ldots, n\}$ of size r and incomparable means that neither $\sigma_i \leq \tau_i \ \forall i$ nor $\tau_i \leq \sigma_i \ \forall i$ holds.

Returning to our modeling, we start from the MaxMinors linear system and apply Gaussian elimination to obtain a basis for its solution space. Now we consider the polynomials in $\mathcal{G}$. Note that since these polynomials do not depend on the specific instance of the problem, they can be precomputed. We

substitute the coordinates of our parameterized solution into the corresponding variables of $\mathcal{G}$, yielding a new polynomial system whose variables are the coefficients of the linear combination. Finally, we compute the Gröbner basis of this system with respect to the `degrevlex` order and we compare the computational cost with or without the inclusion and, also, with the direct approach.

Note that, as we have already discussed, obtaining the values of the minors of C will automatically yield the entries of a solution matrix C'.

5.5 Example

Consider again the MaxMinors system in Example 5.3 in which we have already computed a basis for the solution space

$$\begin{aligned}
v_1 &= (1,1,0,0,0,0,1,0,1,1,0,0,1,0,0),\\
v_2 &= (0,0,1,0,0,0,1,0,0,1,0,0,0,1,0),\\
v_3 &= (0,0,0,0,1,0,0,0,1,1,0,0,0,0,0),\\
v_4 &= (0,0,0,0,0,1,1,0,0,1,0,0,0,0,1),\\
v_5 &= (0,0,0,0,0,0,0,1,1,0,1,0,0,1,0),\\
v_6 &= (0,0,0,0,0,0,0,0,0,0,0,1,0,1,0)
\end{aligned}$$

whose coordinates correspond, in order, to the sequence of minors:

$(x_{0,1},\ x_{0,2},\ x_{0,3},\ x_{0,4},\ x_{0,5},\ x_{1,2},\ x_{1,3},\ x_{1,4},\ x_{1,5},\ x_{2,3},\ x_{2,4},\ x_{2,5},\ x_{3,4},\ x_{3,5},\ x_{4,5})$.

We list now the polynomials in the Gröbner basis of Plücker ideal $I_{6,2}$:

$$\begin{cases} x_{0,3}x_{1,2} + x_{0,2}x_{1,3} + x_{0,1}x_{2,3} = 0, \\ x_{0,4}x_{1,2} + x_{0,2}x_{1,4} + x_{0,1}x_{2,4} = 0, \\ x_{0,4}x_{1,3} + x_{0,3}x_{1,4} + x_{0,1}x_{3,4} = 0, \\ x_{0,4}x_{2,3} + x_{0,3}x_{2,4} + x_{0,2}x_{3,4} = 0, \\ x_{0,5}x_{1,2} + x_{0,2}x_{1,5} + x_{0,1}x_{2,5} = 0, \\ x_{0,5}x_{1,3} + x_{0,3}x_{1,5} + x_{0,1}x_{3,5} = 0, \\ x_{0,5}x_{1,4} + x_{0,4}x_{1,5} + x_{0,1}x_{4,5} = 0, \\ x_{0,5}x_{2,3} + x_{0,3}x_{2,5} + x_{0,2}x_{3,5} = 0, \\ x_{0,5}x_{2,4} + x_{0,4}x_{2,5} + x_{0,2}x_{4,5} = 0, \\ x_{0,5}x_{3,4} + x_{0,4}x_{3,5} + x_{0,3}x_{4,5} = 0, \\ x_{1,4}x_{2,3} + x_{1,3}x_{2,4} + x_{1,2}x_{3,4} = 0, \\ x_{1,5}x_{2,3} + x_{1,3}x_{2,5} + x_{1,2}x_{3,5} = 0, \\ x_{1,5}x_{2,4} + x_{1,4}x_{2,5} + x_{1,2}x_{4,5} = 0, \\ x_{1,5}x_{3,4} + x_{1,4}x_{3,5} + x_{1,3}x_{4,5} = 0, \\ x_{2,5}x_{3,4} + x_{2,4}x_{3,5} + x_{2,3}x_{4,5} = 0. \end{cases}$$

and we substitute each variable with the corresponding coordinate of the parameterized vector $a_0v_0 + a_1v_1 + \cdots + a_5v_5$*, where the coefficients* a_i *are un-*

knowns in $\mathbb{F}_q$*, which gives:*

$$
\begin{cases}
a_0a_2 + a_1a_3 = 0, \\
a_0^2 + a_1a_4 = 0, \\
a_0^2 + a_1a_4 = 0, \\
a_0^2 + a_0a_2 + a_2a_3 + a_0a_4 + a_0a_5 = 0, \\
a_0a_2 + a_2a_3 + a_0a_4 + a_1a_4 + a_0a_5 = 0, \\
a_0a_3 + a_2a_4 = 0, \\
a_0a_1 + a_0a_2 + a_1a_2 + a_2^2 + a_2a_3 + a_0a_4 + a_0a_5 + a_1a_5 = 0, \\
a_0a_3 + a_2a_4 = 0, \\
a_0a_2 + a_1a_3 = 0, \\
a_0a_3 + a_2a_4 = 0, \\
a_0^2 + a_0a_1 + a_1a_2 + a_2^2 + a_0a_3 + a_1a_3 + a_2a_3 + a_0a_4 + a_1a_4 + a_2a_4 \\
\qquad + a_0a_5 + a_1a_5 = 0, \\
a_3^2 + a_0a_4 + a_2a_4 + a_4^2 + a_4a_5 = 0, \\
a_0^2 + a_0a_2 + a_0a_3 + a_1a_3 + a_3^2 + a_0a_4 + a_1a_4 + a_4^2 + a_4a_5 = 0, \\
a_0a_3 + a_1a_3 + a_2a_3 + a_3^2 + a_1a_4 + a_4^2 + a_0a_5 + a_4a_5 = 0, \\
a_0 + 1 = 0.
\end{cases}
$$

Note that we added also a last equation $a_0 + 1 = 0$ *forcing the first minor to be 1. Analogously to the direct approach, in Subsection 5.4, we estimate the computational cost of computing with* F_4 *algorithm a Gröbner basis w.r.t.* `degrevlex` *term order for the ideal generated by these equations both with and without the field equations.*

5.3.1 SageMath code for the Plücker relations approach

Here, we present the SageMath code used to construct the two ideal described in the second approach. Specifically, lines 1–5 define the function for identifying incomparable pairs, while lines 10–18 generate the complete list of these pairs. Note that the dictionary in line 8 maps the tuple T in the corresponding variable $x_{T'}$, where T' corresponds T shifted to the left for each coordinate.

Listing 4: Extract incomparable pairs

```
1  def is_le(sigma, tau):
2          return all(s <= t for s, t in zip(sigma, tau))
3
4  def is_incomparable(sigma, tau):
5          return not is_le(sigma, tau) and not is_le(tau,
               sigma)
6
7  multi_indices = list(Combinations(range(1, n+1), r))
8  dict_variables = {tuple(T): var for T, var in zip(
       multi_indices, variables_x)}
9
10 def extract_incomp_pairs(multi_indices):
11         incomparable_pairs = []
12
13         for sigma, tau in itertools.combinations(
               multi_indices, 2):
14                 if is_incomparable(sigma, tau):
15                         incomparable_pairs.append((sigma,
                               tau))
16
17         return incomparable_pairs
18 incomparable_pairs = extract_incomp_pairs(multi_indices)
```

The next code will construct the polynomials appearing in the Gröbner basis for the Plücker ideal. Lines 1–39, given an incomparable pair, a dictionary as in line 8 of Listing 4 and the ring of the variables x_T generate the polynomial in the form described of the Gröbner basis $\mathcal{G}$ for $I_{n,r}$. Lines 41–49 generate all the polynomials in $\mathcal{G}$.

Listing 5: Generation of the polynomials in the Gröbner basis for Plücker ideal

```
1  def generate_polynomial_plucker(pair, dict_var,
       ring_polynomial):
2          A, B = sorted(pair)
3
4          violation_idx = -1
5          for i in range(len(A)):
6                  if A[i] > B[i]:
7                          violation_idx = i
8                          break
9
10         sigma = A
11         tau = B
12         i = violation_idx
13
14         active_tau = tau[:i+1]
15         active_sigma = sigma[i:]
16         Omega = list(active_tau) + list(active_sigma)
17
18         passive_tau = tau[i+1:]
19         passive_sigma = sigma[:i]
20
21         polynomial = ring_polynomial(0)
22
23         for combo_indices in itertools.combinations(range
```

```
            (len(Omega)), i+1):
24                chosen_for_tau = [Omega[idx] for idx in
                      combo_indices]
25                chosen_for_sigma = [Omega[idx] for idx in
                       range(len(Omega)) if idx not in
                      combo_indices]
26
27                new_tau = tuple(sorted(chosen_for_tau +
                      list(passive_tau)))
28                new_tau = tuple(sorted(chosen_for_tau +
                      list(passive_tau)))
29                new_sigma = tuple(sorted(chosen_for_sigma
                       + list(passive_sigma)))
30
31                if len(set(new_tau)) < len(new_tau) or
                      len(set(new_sigma)) < len(new_sigma):
32                     continue
33
34                var_tau = dict_var[new_tau]
35                var_sigma = dict_var[new_sigma]
36
37                polynomial += var_sigma * var_tau
38
39        return polynomial
40
41  def generate_relations_plucker(incomparable_pairs,
      dict_var, ring_polynomial):
42        relations_plucker = []
43
44        for pair in incomparable_pairs:
45                F = generate_polynomial_plucker(pair,
                      dict_var, ring_polynomial)
```

```
46                  relations_plucker.append(F)
47
48          return relations_plucker
49 relations= generate_relations_plucker(incomparable_pairs,
       dict_variables, R_linear)
```

Given the MaxMinors system, this code applies the substitution of the parameterized solution into the variables of the Plücker relations Lines 1–5 perform the setup and lines 7–15 construct the dictionary for substituting the coordinates of the linear combinations in the variables x_T. Lines 22–29 execute the substitution and in lines 32–35 we append the constraint that the first minor equals 1. Lines 37–39 construct the ideals.

Listing 6: Construction of ideals for Plücker relations approach

```
1  A, monomials = system.coefficients_monomials()
2  vectors_basis = A.right_kernel().basis()
3  t_var = len(vectors_basis)
4  names_a = [f'a_{idx}' for idx in range(t_var)]
5  Ring_Reduced = PolynomialRing(Fq, names_a, order='
       degrevlex')
6
7  dict_subst = {}
8
9  for idx, comb in enumerate(Combinations(range(n), r)):
10         nome_var = "x_" + "_".join(map(str, comb))
11         var_X = R_linear(nome_var)
12         terms = [Ring_Reduced(f'a_{k_idx}') *
               vectors_basis[k_idx][idx]
13         for k_idx in range(t_var) if vectors_basis[k_idx
               ][idx] != 0]
14         comb_a = sum(terms) if terms else Ring_Reduced(0)
15         dict_subst[var_X] = comb_a
16
```

```
17
18  # Substitution map
19  def sub_map(polynomials):
20          return polynomials.subs(dict_subst)
21
22  eq_GB = []
23  map_inverse_plucker = {}
24
25  for eq_P in relations:
26          eq_reduced = sub_map(eq_P)
27          if eq_reduced != 0:
28                  eq_GB.append(eq_reduced)
29                  map_inverse_plucker[eq_reduced] = eq_P
30
31
32  first_columns = tuple(range(r))
33  name_identity = "x_" + "_".join(map(str, first_columns))
34  minor_identity_Reduced = dict_sost[R_linear(name_identity
       )]
35  eq_GB.append(minor_identity_Reduced - 1)
36
37  field_eqs = [var**2 - var for var in Ring_Reduced.gens()]
38  I_PluckerApproach_feqs = Ring_Reduced.ideal(eq_GB +
       field_eqs)
39  I_PluckerApproach = Ring_Reduced.ideal(eq_GB )
```

5.4 Experimental results and comparison

In this subsection we present and make comparisons between the experimental results from the first and second method. Before starting with this, we justify the choice of parameters. Recall first that we want to discuss the underdetermined case where $m\binom{n-k-1}{r} < \binom{n}{r} - 1$. Moreover, in order to not

have multiple solutions affecting the results, we searched parameters satisfying also $4q^{r(m+n-r)-m(n-k)} < \frac{1}{2}$ as in Proposition 2.17. Given these premises, we have considered two parameter sets for evaluating the behavior of the system as n increases:

- The first set $r = 2, m = 14$ and $n \in \{10, \ldots, 14\}$ and $k \in \{6, 7, 8, 9, 10\}$.
- The second set $r = 3, m = 13$, $n \in \{6, 8, 9, 10\}$ and $k \in \{2, 3, 4, 5\}$.

In both scenarios, we increase the dimension k alongside n. This choice ensures that the generated polynomial systems do not become overdetermined.

Note that for the following results we have used the `msolve` package for the F_4 algorithm in SageMath. Moreover, we only considered unfolded modelings because the implementation of F_4 in this package does not work with non-prime finite field extensions. All experiments were conducted on a machine equipped with an AMD Ryzen 5 5500U processor (2.10 GHz) and 8 GB of RAM.

In what follows, we study the percentage of zero-dimensional ideals, the maximal solving degree obtained, the maximal number of rows and columns of the matrices used by the F_4 algorithm and the average time for computing the `degrevlex` Gröbner basis. We considered approximately 100 iterations for every case, with the exception of $r = 2, n = 14$ and $r = 3, n = 10$. For these parameters, the high memory demand of the Gröbner basis computations led to Out-Of-Memory errors that terminated the kernel. Consequently, we reduced the number of iterations for these instances depending on the chosen algebraic modeling. Note that, to mitigate the variance in the following results, we averaged the measurements across multiple independent runs and for this reason, the label "Average" appears in the following figures.

We first consider the study of the percentage of zero-dimensional ideals in the parameter set with $r = 2$. As shown in Figure 1, it is almost 100% for every n in the case of ideals containing the field equations. The "almost"

is probably due to isolated crashes of the algorithm. Regarding ideals without field equations, we observe that the direct minors approach obtains the minimum percentage among all other modelings for $n = 14$. Overall, the percentage is high in all modelings. A non-trivial result is that, in the zero-dimensional case, the solution turns out to be unique even for ideals without field equations. This applies also for $r = 3$.

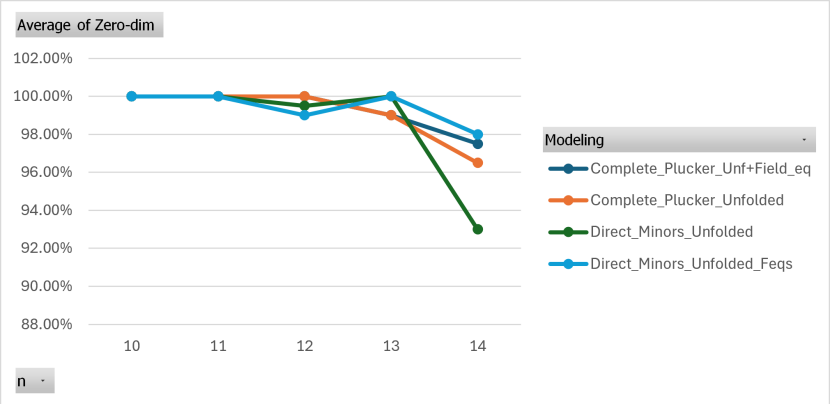


Figure 1: Percentage of zero-dimensional ideals for the parameter set $r = 2$.

For $r = 3$, the percentage remains overall high across all modelings. Only for $n = 6$ we observe a minimum percentage of circa 92% for the ideals without field equations. Note that the Plücker relations approach without field equations stops at $n = 9$, because for $n = 10$ the Gröbner basis computation exhausted the available memory. We will explain a possible reason for this when we discuss the maximal number of rows.

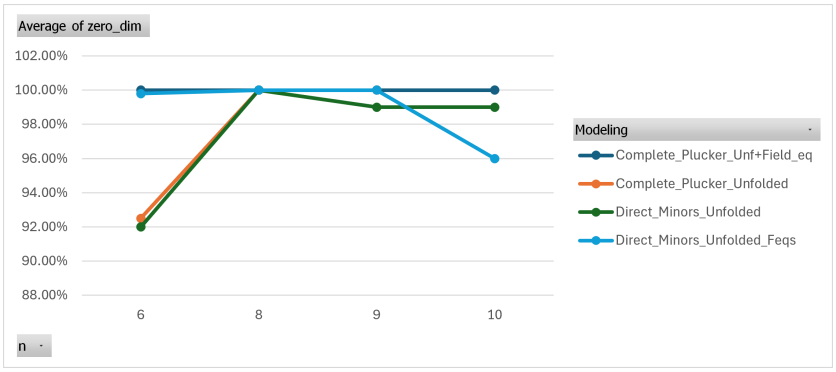


Figure 2: Percentage of zero-dimensional ideals for the parameter set $r = 3$.

Next, we analyze the solving degree achieved by the F_4 algorithm.

For both $r = 2$ and $r = 3$, the solving degree of direct approach results generally higher than that of the methods based on Plücker relations. A plausible explanation is that Plücker relations introduce a significantly large number of quadric equations and this give sufficient constraints to compute the Gröbner basis without allowing the solving degree to exceed 3. Overall the solving degree remains low indicating that it is not the primary cause of the computational cost.

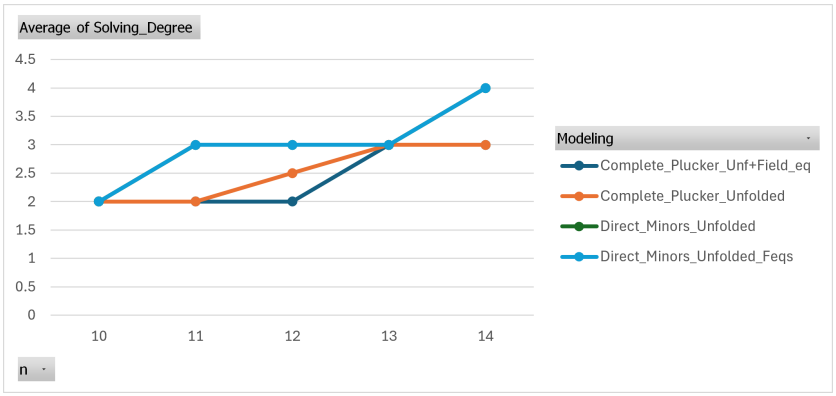


Figure 3: Average of maximal solving degree reached during iterations of F_4 algorithm for $r = 2$.

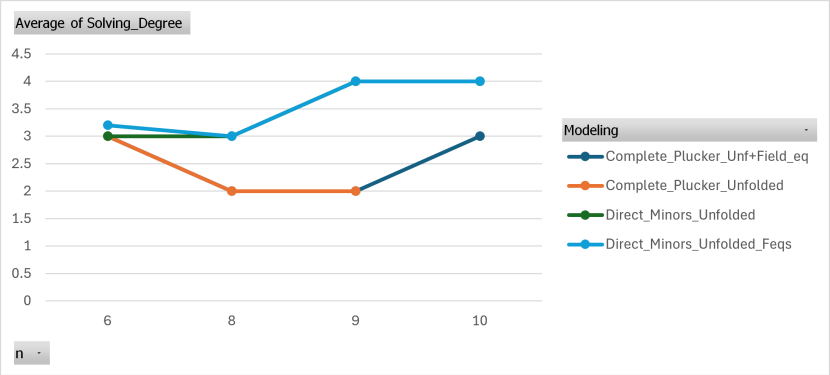


Figure 4: Average of maximal solving degree reached during iterations of F_4 algorithm for $r = 3$.

Regarding the maximal number of matrix rows generated during the F_4 execution, we observe notable differences between the modelings. We conjecture that, due to the high number of equations, the Plücker relations approach shows a significantly faster growth in the number of rows as n assumes high values.

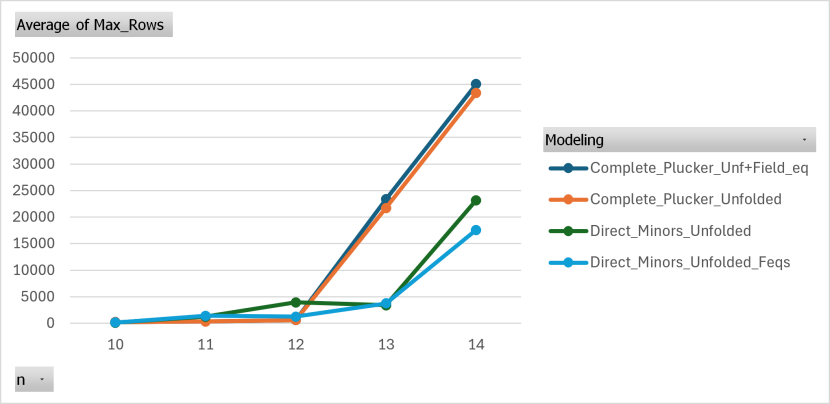


Figure 5: Average of maximal number of matrix rows generated during the F_4 execution for $r = 2$.

Also in the case $r = 3$, the trend described is evident which also explains

the fact that for $n = 10$ the Plücker relations approach without field equations results in an Out-Of-Memory error.

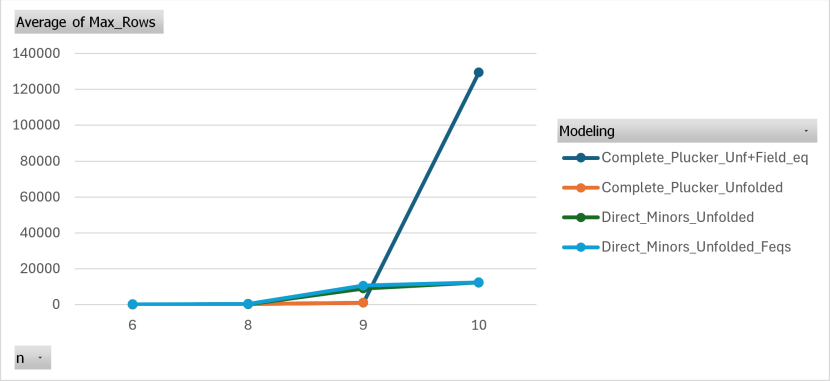


Figure 6: Average of maximal number of matrix rows generated during the F_4 execution for $r = 3$.

Similarly the trend is observed also for the maximal number of columns, especially for $r = 3$. For $r = 2$ this difference is less pronounced.

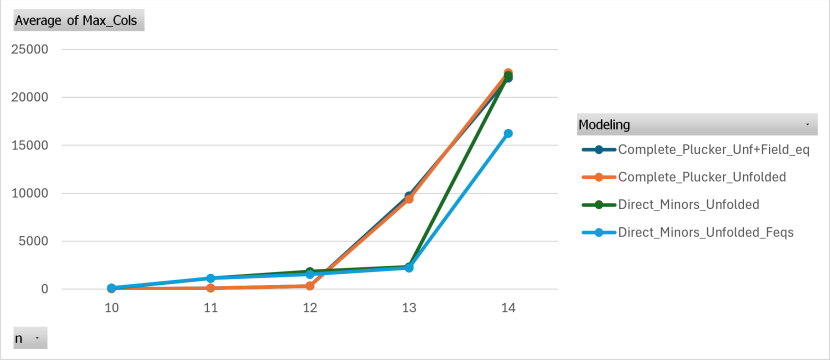


Figure 7: Average of maximal number of matrix columns generated during the F_4 execution for $r = 2$.

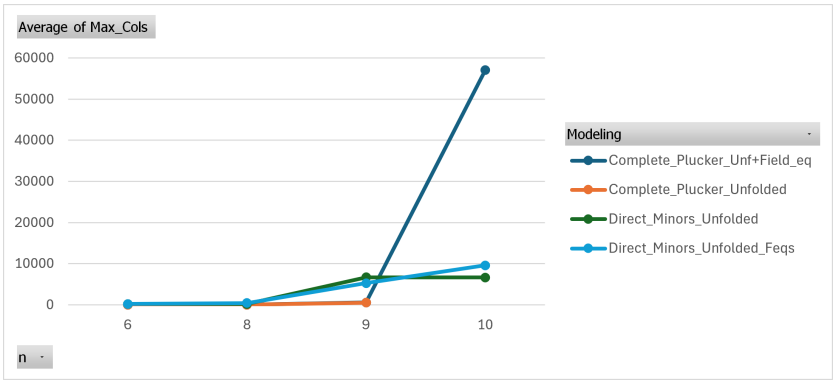


Figure 8: Average of maximal number of matrix columns generated during the F_4 execution for $r = 3$.

Finally, regarding the average time employed for computing a Gröbner basis we observe that for $r = 2$ the two methods without field equations are similar. Moreover the direct minors approach results the fastest.

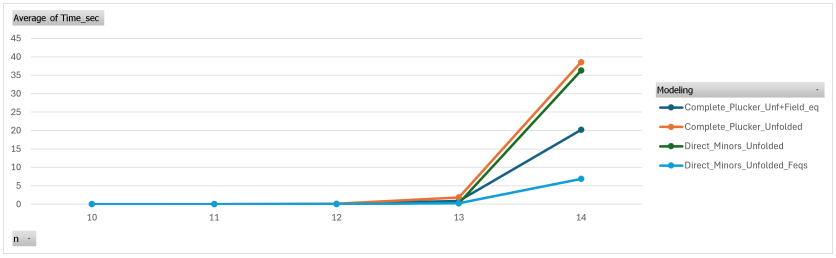


Figure 9: Average of execution time of the F_4 algorithm for $r = 2$.

For the subsequent analysis of the $r = 3$ case, note that the time axis in Figure 10 is presented on a logarithmic scale. In this we observe that the direct minors approach with field equations results to be the most efficient method.

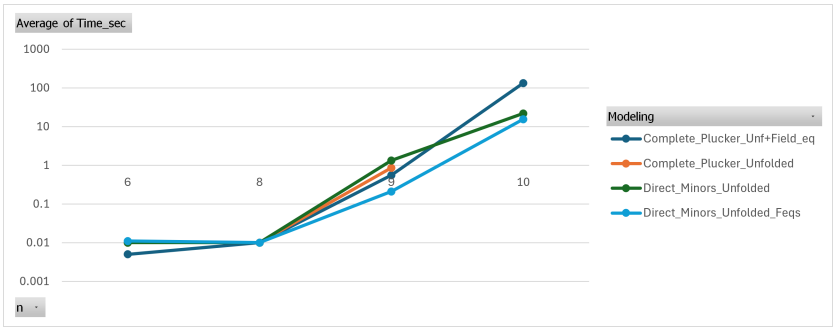


Figure 10: Average of execution time of the F_4 algorithm for $r = 3$.

5.4.1 Percentage of Plücker relations and single-swap relations approaches

In order to test whether the high number of relations causes the high computational cost, we randomly sampled a percentage of Plücker relations for the case $r = 2, n = 14$. To identify a potential threshold between computational efficiency and algebraic correctness, we considered a specific sequence of percentages: 60%, 70%, 80%, 85% and 90%. For each considered percentage, we extracted a subset of Plücker relations and kept it fixed across multiple independent iterations, analyzing the same computational measures detailed in the previous subsection. Moreover, we considered also the same idea of modeling but with single-swap Plücker relations, as presented for $n = 4$ and $r = 2$ in Subsection A.2.

Below, we will not present results for $r = 3$. Due to a high frequency of Out-Of-Memory errors, we consider the data for this parameter set to be unreliable.

In Figure 11, we observe that the execution time increases as the percentage of Plücker relations decreases. However, when we keep at least 85% of the relations, the execution time results almost constant.

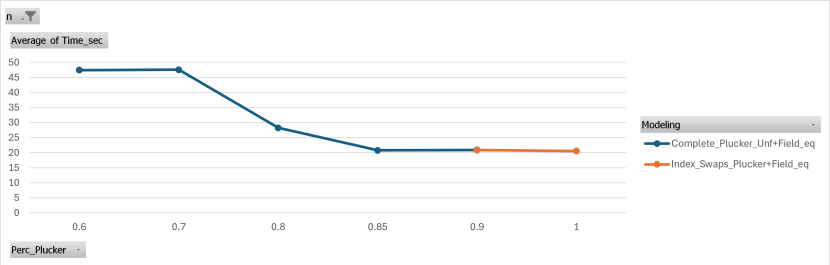


Figure 11: Average of execution time of the F_4 algorithm for $r = 2, n = 14$, varying the percentage of relations.

Interestingly, the maximum number of rows decreases when the percentage drops below 80%. This suggests that a well-chosen subset of Plücker relations can potentially reduce the computational cost.

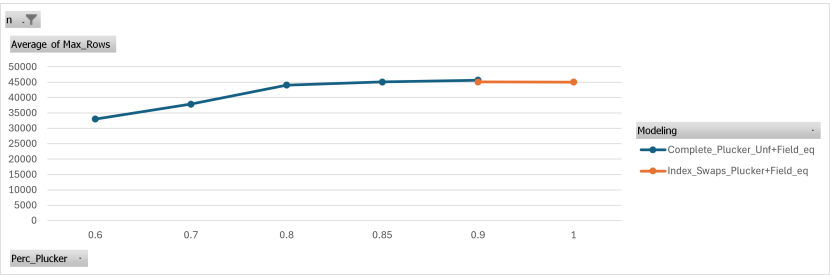


Figure 12: Average of maximal number of matrix rows generated during the F_4 execution for $r = 2, n = 14$, varying the percentage of relations.

Regarding the dimension of the ideals, we observe a high number of zero-dimensional ideals also when we keep only 60% of the relations.

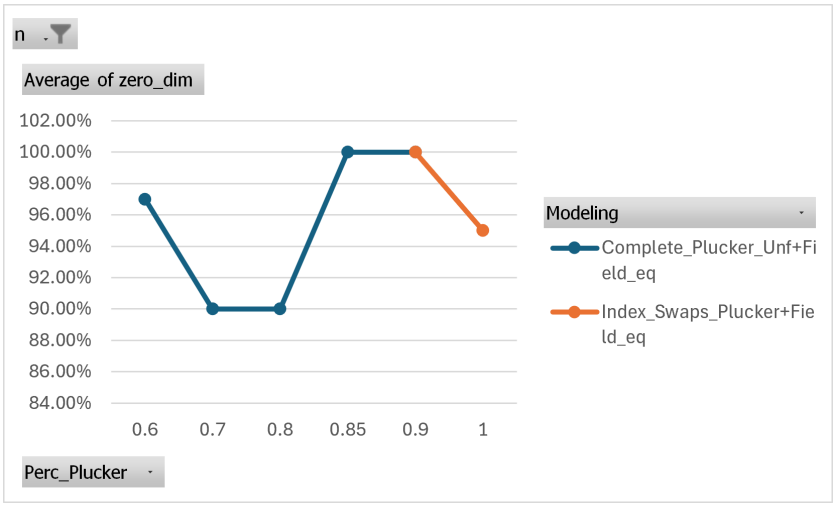


Figure 13: Percentage of zero-dimensional ideals for $r = 2, n = 14$ and varying the percentage of relations.

We conclude that, in general, keeping more than 85% of the equations yields similar results to using the full percentage of generators for Plücker relations.

We also conducted tests on smaller values of n and observed that the required percentage of Plücker relations grows as n increases. This behavior can be attributed to the fact that the system becomes increasingly underdetermined as n increases and, consequently, the number of variables in the ideal grows causing an high computational cost.

Furthermore, we experimented with resampling a different subset of Plücker relations for each iteration. While the overall results are similar to those presented above, we observed a more pronounced increase in execution time as the percentage drops. This behavior suggests a dependence on the chosen subset.

5.4.2 Future Directions

How shown in the previous subsection, randomly extracting a subset of the Plücker relations did not yield a decrease in the overall computational cost.

Future research could explore the following strategies:

- Selecting specific subset of Plücker relations or attributing weighted probabilities to polynomials depending on the basis of the solution of the MaxMinors linear system.
- Investigating the dependence between the parameters of the system, the cardinality of the basis and the percentage of Plücker relations sufficient to solve the system with a high probability.

A Exterior powers, determinants and Plücker relations

In this section, we present the theory and the proofs behind Proposition A.3, namely the generalized Cauchy-Binet formula, and Theorem A.7, which provides a Gröbner basis for the ideal of Plücker relations with respect to a particular term order. The former relies on the theory of exterior powers and the latter is based on the work of Miller and Sturmfels [MS09].

A.1 The generalized Cauchy-Binet formula

The idea of the proof that we will show, consists of leveraging the functoriality of wedge functor in order to decompose the determinant of AB into the formula described. Before defining the wedge product it is necessary to introduce the multilinear and alternating functions.

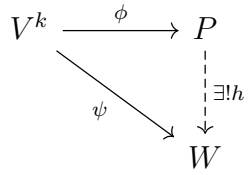
Let $\mathbb{F}$ be a field and let V, W be two $\mathbb{F}$-vector spaces. A map $f : V^k \to W$ is called **multilinear** over $\mathbb{F}$ if it is linear in each of its k variables. That is, for any index $i \in \{1, \ldots, k\}$, any elements $x_1, \ldots, x_k, y \in V$ and any scalars $a, b \in \mathbb{F}$:

$$f(x_1, \ldots, ax_i + by, \ldots, x_k) = af(x_1, \ldots, x_i, \ldots, x_k) + bf(x_1, \ldots, y, \ldots, x_k)$$

We say that a multilinear map $f : V^k \to W$ is **alternating** if $f(x_1, \ldots, x_k) = 0$ whenever there exists $i \neq j$ such that $x_i = x_j$.

There exists a universal property for alternating maps formulated as follows. Given any $\mathbb{F}$-vector space V and a positive integer $k \geq 1$, there exists an $\mathbb{F}$-vector space P (typically denoted as $\bigwedge^k V$) and an alternating map $\phi : V^k \to P$ such that for every $\mathbb{F}$-vector space W and every alternating map $\psi : V^k \to W$, there exists a unique linear map $h : P \to W$ satisfying $h \circ \phi = \psi$.

This universal property can be summarized by the following commutative diagram:



The k-th exterior power of V can be defined as

$$\bigwedge^k(V) := \frac{F_0(V^k)}{U + Z},$$

where

- $F_0(V^k)$ is the $\mathbb{F}$-vector space of the functions with finite support from V^k to $\mathbb{F}$ having as basis the elements $\{e_{(v_1,\dots,v_k)} \mid (v_1,\dots,v_k) \in V^k\}$, where

$$e_{(v_1,\dots,v_k)} : V^k \to \mathbb{F}$$
$$(v'_1,\dots,v'_k) \mapsto \begin{cases} 1 & \text{if } v'_i = v_i \ \forall i \in \{1,\dots,k\} \\ 0 & \text{else} \end{cases}$$

- U is the subspace generated by all the elements of the form

$$e_{(v_1,\dots,v_i+v'_i,\dots,v_k)} - e_{(v_1,\dots,v_i,\dots,v_k)} - e_{(v_1,\dots,v'_i,\dots,v_k)},$$

  and

$$e_{(v_1,\dots,rv_i,\dots,v_k)} - r\, e_{(v_1,\dots,v_i,\dots,v_k)},$$

  for every $r \in \mathbb{F}$;

- Z is the subspace generated by all the elements of the form

$$e_{(v_1,\dots,v_k)}$$

  such that $v_i = v_j$ for some $i \neq j$.

Given this structure, we introduce the notation $v_1 \wedge \cdots \wedge v_k \coloneqq \overline{e_{(v_1,\ldots,v_k)}}$ and we call the elements of this form the elementary wedge products. Moreover, by defining the function ϕ as follows

$$\phi : V^k \to \bigwedge^k(V)$$
$$(v_1, \ldots, v_k) \mapsto v_1 \wedge \cdots \wedge v_k$$

one can verify that the pair $(\bigwedge^k(V), \phi)$ satisfies the universal property.

Let V be an $\mathbb{F}$-vector space with basis $\{e_i \mid i \in I\}$ and let $J = \{j_1, \ldots, j_k\} \subseteq I$ be a finite subset of I. Then, there exists a linear map $\det_J : \bigwedge^k(V) \to \mathbb{F}$ defined as follows

$$\det_J(e_{i_1} \wedge \cdots \wedge e_{i_k}) = \begin{cases} \operatorname{sign}(\sigma) & \text{if } \{i_1, \ldots, i_k\} = J \\ 0 & \text{otherwise} \end{cases}$$

where $\sigma \in S_k$ is the permutation such that $i_m = j_{\sigma(m)}$ for all $m = 1, \ldots, k$, assuming the elements of $J = \{j_1, \ldots, j_k\}$ are sorted strictly as $j_1 < j_2 < \cdots < j_k$.

Indeed, denoting with $\pi_t : V \to \mathbb{F}$ for $t = 1, \ldots, k$ the projection on the coordinate corresponding to the basis element e_{j_t}, since the determinant is multilinear and alternating in its columns, the map $\psi : V^k \to \mathbb{F}$ defined as

$$\psi(\alpha_1, \ldots, \alpha_k) = \det \begin{bmatrix} \pi_1(\alpha_1) & \pi_1(\alpha_2) & \ldots & \pi_1(\alpha_k) \\ \pi_2(\alpha_1) & \pi_2(\alpha_2) & \ldots & \pi_2(\alpha_k) \\ \vdots & \vdots & \vdots & \vdots \\ \pi_k(\alpha_1) & \pi_k(\alpha_2) & \ldots & \pi_k(\alpha_k) \end{bmatrix}$$

is also alternating. Moreover,

$$\psi(e_{i_1}, \ldots, e_{i_k}) = \begin{cases} \operatorname{sign}(\sigma) & \text{if } \{i_1, \ldots, i_k\} = J \\ 0 & \text{otherwise} \end{cases}$$

Indeed, if $\{i_1, \ldots, i_k\} \neq J$, then there exists $j_t \in J$ that does not belong to $\{i_1, \ldots, i_k\}$. Consequently,

$$\pi_t(e_{i_1}) = \cdots = \pi_t(e_{i_k}) = 0,$$

so the t-th row of the matrix is zero, forcing its determinant to vanish. Furthermore, the factor $\operatorname{sign}(\sigma)$ derives from the row swaps required to convert the obtained matrix into the identity matrix.

Finally, by the universal property of the exterior power, the alternating map ψ induces the linear map $\det_J$.

A.1 Proposition
The k-th exterior power $\bigwedge^k V$ of an $\mathbb{F}$-vector space V of finite dimension $\dim(V)$ is an $\mathbb{F}$-vector space of dimension $\binom{\dim(V)}{k}$.

Proof. Let $E = \{e_i \mid i \in I\}$ be an ordered basis of the $\mathbb{F}$-vector space V. We show that

$$E_k = \{e_{i_1} \wedge \cdots \wedge e_{i_k} \mid i_1 < \cdots < i_k\}$$

is a basis of $\bigwedge^k(V)$.

The set E_k generates $\bigwedge^k(V)$, since every elementary wedge product can be rewritten, up to sign, with increasing indices.

It remains to prove linear independence. Suppose that there is a linear relation among elements of E_k and let

$$e_{i_1} \wedge \cdots \wedge e_{i_k}$$

be one of the terms with coefficient $a \in \mathbb{F}$.

Since the relation is finite, only finitely many basis elements of V appear. Let J be this finite set of indices, let $J' = J \setminus \{i_1, \ldots, i_k\}$ and write

$$e_{J'} = \bigwedge_{j \in J'} e_j.$$

The map $\Phi : V^k \to \bigwedge^{k+|J'|} V$ defined as

$$(x_1, \ldots, x_k) \longmapsto x_1 \wedge \cdots \wedge x_k \wedge e_{J'}$$

is multilinear and alternating.

Consequently, it induces a linear map $h : \bigwedge^k(V) \to \bigwedge^{k+|J'|}(V)$.

Composing h with the determinant map on $\bigwedge^{k+|J'|}(V)$, we obtain a linear map that sends

$$e_{i_1} \wedge \cdots \wedge e_{i_k}$$

to ± 1 and every other basis element in the relation to 0. Applying this map to the relation gives

$$a = \det_{k+|J'|}(ae_{i_1} \wedge \cdots \wedge e_{i_k}) = 0.$$

Since the chosen coefficient was arbitrary, all coefficients vanish, which concludes the proof. □

A.2 Lemma

Let V and W be two $\mathbb{F}$-vector spaces with bases respectively $\{e_1, \ldots, e_r\}$ and $\{f_1, \ldots, f_n\}$ where $r \leq n$. Let $L_B : V \to W$ be the linear map associated with the matrix $B \in \mathrm{Mat}_{n\times r}(\mathbb{F})$ w.r.t. the bases introduced above. Then the application induced by the wedge functor $\bigwedge^r L_B : \bigwedge^r V \to \bigwedge^r W$ maps the generator of the space $\bigwedge^r V$ as follows:

$$\bigwedge^r L_B(e_1 \wedge \cdots \wedge e_r) = \sum_{T\subseteq\{1,\ldots,n\}\,|\,|T|=r} \det(B|_{*,T}) f_T$$

where $f_T = f_{t_1} \wedge \cdots \wedge f_{t_r}$ with $1 \leq t_1 < \cdots < t_r \leq n$.

Proof. The elements of the form $f_T = f_{t_1} \wedge \cdots \wedge f_{t_r}$ with $1 \leq t_1 < \cdots < t_r \leq n$ form a basis for the space $\bigwedge^r W$. Consequently, there exist uniquely some coefficients $c_I \in \mathbb{F}$ such that

$$\bigwedge^r L_B(e_1 \wedge \cdots \wedge e_r) = \sum_{I\subseteq\{1,\ldots,n\}\,|\,|I|=r} c_I f_I.$$

To determine the coefficients c_T, we use the linear map $\det_T : \bigwedge^r W \to \mathbb{F}$ defined previously. By applying $\det_T$, we obtain

$$\det_T\left(\bigwedge^r L_B(e_1 \wedge \cdots \wedge e_r)\right) = \det_T\left(\sum_{I\subseteq\{1,\ldots,n\}\,|\,|I|=r} c_I f_I\right)$$

Using linearity, we obtain that the right-hand side of the equation is equal to c_T. Consider now the left-hand side. Since the map $\det_T$ is induced by the alternating map $\psi_T : W^r \to \mathbb{F}$, we obtain

$$c_T = \det_T(L_B(e_1) \wedge \cdots \wedge L_B(e_r)) = \psi_T(L_B(e_1), \ldots, L_B(e_r)).$$

Finally, the map ψ_T is defined as the determinant of the matrix $(\pi_{T_i}(L_B(e_j)))_{i,j\in\{1,\ldots,r\}}$. The vector $L_B(e_j)$ represents the j-th column of B and the projection $\pi_{T_i}(L_B(e_j))$ gives the element corresponding to the t_i-th row and j-th column of B. Consequently, the left-hand side of the equation corresponds exactly to the determinant of the submatrix $B|_{T,*}$, yielding that $c_T = \det(B|_{T,*})$. This concludes the proof. □

A.3 Proposition (Generalized Cauchy-Binet Formula)
Let $\mathbb{F}$ be a field, let n and r be two positive integers with $n \geq r$ and let $A \in \mathrm{Mat}_{r\times n}(\mathbb{F})$ and $B \in \mathrm{Mat}_{n\times r}(\mathbb{F})$ be two matrices over $\mathbb{F}$. Then the following formula holds:

$$\det(AB) = \sum_{T\subset\{1,\ldots,n\},|T|=r} \det(A|_{*,T}) \det(B|_{T,*})$$

where $A|_{*,T}$ denote the square submatrix of A whose rows are indexed by T and analogously $B|_{T,*}$ denote the square submatrix of B whose rows are indexed by T.

Proof. We define the linear maps $L_A : \mathbb{F}^n \to \mathbb{F}^r$, $L_B : \mathbb{F}^r \to \mathbb{F}^n$ associated respectively to the matrices A and B.

$$\bigwedge^r (L_A \circ L_B) = \bigwedge^r (L_{AB}) = \det(AB)$$

where L_{AB} is the linear map associated to AB and the last equality holds by Lemma A.2. On the other hand the functoriality of wedge product yields $\bigwedge^r (L_A \circ L_B) = \bigwedge^r (L_A) \circ \bigwedge^r (L_B)$. Moreover, from Lemma A.2 we obtain

$$\bigwedge^r L_B(e_1 \wedge \cdots \wedge e_r) = \sum_{T\subseteq\{1,\ldots,n\}\,|\,|T|=r} \det(B|_{*,T}) f_T.$$

By applying $\bigwedge^r(L_A)$ we deduce that

$$\bigwedge^r(L_A) \circ \bigwedge^r(L_B)(e_1 \wedge \cdots \wedge e_r) = \sum_{T \subseteq \{1,\ldots,n\} \,|\, |T|=r} \det(B|_{*,T}) \bigwedge^r(L_A)(f_T).$$

Now, since $\bigwedge^r \mathbb{F}^r \cong \mathbb{F}$ and following the same reasoning as in the proof of the Lemma A.2, the element $\bigwedge^r(L_A)(f_T)$ corresponds to the determinant of the square submatrix $(A|_{*,T})$ multiplied by the generator $e_1 \wedge \cdots \wedge e_r$. Unifying the results, we obtain the thesis. □

A.4 Example

Set $r = 2$ and $n = 3$ and consider the matrices $A \in \mathrm{Mat}_{2\times 3}(\mathbb{F}_5)$ and $B \in \mathrm{Mat}_{3\times 2}(\mathbb{F}_5)$ defined as:

$$A = \begin{pmatrix} 1 & 2 & 0 \\ -1 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & -1 \\ 0 & 2 \\ 1 & 1 \end{pmatrix}.$$

First, we compute the product matrix $AB \in \mathrm{Mat}_{2\times 2}(\mathbb{F}_5)$:

$$AB = \begin{pmatrix} 1 & 3 \\ 0 & 2 \end{pmatrix},$$

which yields $\det(AB) = 2$.

Now, we apply the Cauchy-Binet formula. The subsets $T \subset \{1,2,3\}$ with $|T| = 2$ are $\{1,2\}, \{1,3\}$ and $\{2,3\}$. We compute the corresponding maximal minors for both matrices:

- *For $T = \{1,2\}$:*

$$\det(A|_{*,\{1,2\}}) = \det \begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix} = 2, \quad \det(B|_{\{1,2\},*}) = \det \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} = 2.$$

- *For $T = \{1,3\}$:*

$$\det(A|_{*,\{1,3\}}) = \det \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = 1, \quad \det(B|_{\{1,3\},*}) = \det \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = 2.$$

- *For $T = \{2, 3\}$:*

$$\det(A|_{*,\{2,3\}}) = \det\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} = 2, \quad \det(B|_{\{2,3\},*}) = \det\begin{pmatrix} 0 & 2 \\ 1 & 1 \end{pmatrix} = -2.$$

Summing the products of these corresponding minors, we obtain:

$$\sum_{\substack{T \subset \{1,2,3\} \\ |T|=2}} \det(A|_{*,T}) \det(B|_{T,*}) = (2 \cdot 2) + (1 \cdot 2) + (2 \cdot -2)$$

$$= 4 + 2 - 4 = 2.$$

This matches the determinant of the product AB, confirming the formula.

A.2 Plücker relations

This Subsection is based on the work of Miller and Sturmfels [MS09].

Informally, Plücker relations are the algebraic conditions that the maximal minors of a matrix must satisfy, not depending on the specific matrix, but only on the condition of being minors. An introductive example is as follows. Let

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} \end{bmatrix}$$

be a generic matrix of size 2×4. Consider now the 3×3 matrix $C_1^{1,2,3}$, constructed by appending a copy of the first row of C as its third row and restricting the matrix to the first three columns:

$$C_1^{1,2,3} = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{1,1} & c_{1,2} & c_{1,3} \end{bmatrix}$$

Since $C_{1,2,1}$ has two identical rows, its determinant is necessarily zero. Consequently, denoting as X_T the maximal minor of C indexed by the columns in T, we derive that

$$c_{1,1}X_{2,3} - c_{1,2}X_{1,3} + c_{1,3}X_{1,2} = 0.$$

By repeating this process also for the second row, we derive that

$$C_1X_{2,3} - C_2X_{1,3} + C_3X_{1,2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where C_i denotes the i-th column of C. By appending the fourth column of C and computing the determinant of the resulting matrix, we obtain the condition

$$X_{1,4}X_{2,3} - X_{2,4}X_{1,3} + X_{3,4}X_{1,2} = 0.$$

The argument detailed above can be adapted to obtain the relations arising from single-index swaps for maximal $r \times r$ minors of an $r \times n$ matrix. However, these specific equations are not sufficient, in general, to describe all the algebraic relations satisfied by the maximal minors.

Formally, let $C = (c_{i,j})$ be an arbitrary $r \times n$ matrix of variables, we denote by X_T the minors of the submatrix of C whose columns are indexed by T, that is $X_T = \det(C|_{*,T})$.

Let $\mathbb{F}$ be a field, we define the Plücker algebra over $\mathbb{F}$ as the subalgebra $\mathbb{F}[X] \subset \mathbb{F}[c_{i,j}]$ as the polynomial ring over $\mathbb{F}$ generated by the maximal minors of C, that is

$$\mathbb{F}[X] \coloneqq \mathbb{F}[X_T \,|\, T \subset \{1, \ldots, n\},\, |T| = r].$$

We define a second polynomial ring by introducing the $\binom{n}{r}$ variables p_T where T varies over the subsets of $\{1, \ldots, n\}$ of size r. We consider each subset T as the strictly increasing string of its elements. Then we extend the definition of these variables to arbitrary strings of length r, forcing the condition for alternating sequences. Specifically, swapping any two indices flips the sign of the variable and any repeated indices annihilates the variable in $\mathbb{F}[p]$. Now we establish the ring homomorphism $\phi_{n\times r}$ defined as

$$\begin{aligned} \phi_{n,r} : \mathbb{F}[p] &\to \mathbb{F}[X] \\ p_T &\mapsto X_T = \det(C|_{*,T}) \end{aligned}$$

Since the determinant of a submatrix respects the exact same conventions introduced above, this map is well-defined for any arbitrary string of indices. Moreover, the homomorphism $\phi_{n,r}$ is surjective by construction. The elements of its kernel ideal $I_{n,r} = \ker(\phi_{n,r})$ are called the **Plücker relations**.

The objective now is to find a set of generators for this ideal. Specifically, at the end of this section we will describe a Grobner basis with respect to a specific term order over $\mathbb{F}[p]$. First let us introduce a partial order on the set of variables $P = \{p_T \,|\, T \subset \{1, \ldots, n\},\, |T| = r\}$ we say that $p_T \leq p_R$ if $t_i \leq r_i$ for all $i \in \{1, \ldots, r\}$.

Second we define a variable order $\prec$ which corresponds to the lexicographical order on the variables in $\mathbb{F}[p]$. Finally we extend $\prec$ to the `degrevlex` term order induced by the variable order $\prec$.

Let us consider the case $n = 4$, $r = 2$, we first describe the poset P. The elements of P are indexed by all possible subsets of 2 columns chosen from $\{1, 2, 3, 4\}$, which correspond to

$$p_{12}, p_{13}, p_{14}, p_{23}, p_{24}, p_{34}$$

We observe that the following chains hold:

$$p_{12} \leq p_{13} \leq p_{14} \leq p_{24} \leq p_{34}$$
$$p_{12} \leq p_{13} \leq p_{23} \leq p_{24} \leq p_{34}$$

Moreover, the pair p_{14} and p_{23} are incomparable w.r.t. $\leq$, since we have an inequality in the first component $1 \leq 2$ and the reversed inequality in the second coordinate $3 \geq 4$. An interesting aspect is that the Plücker relation we have introduced at the start of this subchapter

$$p_{14}p_{23} - p_{12}p_{34} + p_{13}p_{24} = 0$$

has as leading term w.r.t. $\prec$ the term $p_{14}p_{23}$. In particular, in this case the ideal $I_{4,2}$ is principal and generated by the relation $p_{14}p_{23} - p_{12}p_{34} + p_{13}p_{24}$. The fact that its leading term is incomparable w.r.t. $\leq$ is not a coincidence

and in Theorem A.7 we will prove that the homogeneous quadrics in $I_{n,r}$ having as leading term an incomparable pair form a Gröbner basis w.r.t. $\prec$.

For the proof of the theorem will be useful the notion of (p,q)-shuffle. The simple idea of shuffles is the same as cutting a deck of cards in two subdecks and riffle shuffle the two subdecks together. This shuffle will not change the order of the two subdecks. Let $N = p + q$. We call permutation $\pi \in S_N$ a (p,q)-shuffle of the chain

$$\sigma_1 < \cdots < \sigma_p < \sigma_{p+1} < \cdots < \sigma_{p+q} \tag{1}$$

if it preserves the order of the first p elements and of the last q elements. That is, if π respects both these conditions:

1. $\sigma_{\pi(1)} < \sigma_{\pi(2)} < \cdots < \sigma_{\pi(p)}$;
2. $\sigma_{\pi(p+1)} < \sigma_{\pi(p+2)} < \cdots < \sigma_{\pi(p+q)}$.

For proving the next Theorem, we need some preliminary lemmas.

A.5 Lemma
Let $\Omega \coloneqq \{1, 2, \ldots, p+q\}$ be the set of indices corresponding to the chain (1). *Given a (p,q)-shuffle $\pi \in S_{p+q}$ we denote with $B_1(\pi)$ the first block of indexes permuted $B_1(\pi) = \{\pi(1), \ldots, \pi(p)\}$ and, analogously, the second block $B_2(\pi) = \{\pi(p+1), \ldots, \pi(p+q)\}$.*

Let $a, b \in \Omega$ with $b > a$ be two indices separated by π, that is $a \in B_1(\pi)$ and $b \in B_2(\pi)$. Then, there exists a unique (p,q)-shuffle π' of (1) *yielding the following blocks:*

$$B_1(\pi') = (B_1(\pi) \setminus \{a\}) \cup \{b\}$$
$$B_2(\pi') = (B_2(\pi) \setminus \{b\}) \cup \{a\}$$

reordered increasingly. Moreover the sign of this shuffle is given by $\operatorname{sign}(\pi') = (-1)^{b-a} \operatorname{sign}(\pi)$.

Proof. First, we observe that, by definition, any (p, q)-shuffle π is uniquely determined by selecting a subset S of size p of Ω. Indeed, π is constructed by placing the element of S in $B_1(\pi)$ and the other elements in the second block in increasing order. This justifies the uniqueness of π'.

Let π be a (p, q)-shuffle separating a and b. By reordering the obtained blocks $B_1(\pi')$ and $B_2(\pi')$, the permutation π' constructed as in the statement of the Proposition is clearly a (p, q)-shuffle. Finally, we must check that $\operatorname{sign}(\pi') = (-1)^{b-a} \operatorname{sign}(\pi)$ holds. Between a and b in the chain (1) there are elements belonging to $B_1(\pi)$ and $B_2(\pi)$. Let k be the number of elements of $B_1(\pi)$ strictly between a and b and, analogously m for $B_2(\pi)$, thus $k + m = b-a-1$. After swapping a and b, the element b in the first block is larger than all these k elements, requiring exactly k transpositions to yield the increasing reordering. Symmetrically, the element a in the second block requires m transpositions. Counting the total number of transposition alongside with the initial swap between a and b, we obtain

$$k + m + 1 = (b - a - 1) + 1 = b - a.$$

This concludes the proof. □

A.6 Lemma

Let V be a vector space over a field $\mathbb{F}$ and let $f : V^n \to k$ be an alternating function. If $v_1, \ldots, v_n \in V$ is a linearly dependent sequence of vectors, then

$$f(v_1, \ldots, v_n) = 0.$$

Proof. For some non-all zero coefficients, let $\sum_{i=1}^{n} c_i v_i = 0$ be a linear dependency of $v_1, \ldots, v_n$. Without loss of generality, assume that $c_n \neq 0$. Substituting the expression $v_n = -c_n^{-1} \sum_{i=1}^{n-1} c_i v_i$ into f, we obtain:

$$f(v_1, \ldots, v_{n-1}, v_n) = f\left(v_1, \ldots, v_{n-1}, -c_n^{-1} \sum_{i=1}^{n-1} c_i v_i\right).$$

By the multilinearity of f we obtain:

$$f(v_1, \ldots, v_n) = -c_n^{-1} \sum_{i=1}^{n-1} c_i f(v_1, \ldots, v_{n-1}, v_i).$$

The alternating property of f yields $f(v_1, \ldots, v_n) = 0$, concluding the proof. □

The following theorem was originally proved by Sturmfels and White in [SW89]. Here, we follow the proof provided in [MS09, Theorem 14.6].

A.7 Theorem

Let $I_{r,n}$ be the ideal of Plücker relations. The ideal $I_{r,n}$ admits a Gröbner basis under the term order $\prec$ consisting of homogeneous quadrics having as leading terms the products $p_\sigma p_\tau$ of incomparable pairs, where σ and τ are (increasingly ordered) subsets of $\{1, \ldots, n\}$ of cardinality r.

Proof. In this proof we first construct for each incomparable pair (σ, τ), a polynomial $F \in I_{r,n}$ whose leading term is exactly $p_\sigma p_\tau$. This will prove the inclusion $(p_\sigma p_\tau \mid \sigma, \tau \text{ incomparable}) \subseteq \text{in}_\prec(I_{r,n})$. Fix a pair $p_\sigma p_\tau$. From the incomparability, we know that there exists an index $i \in \{1, \ldots, r\}$ satisfying $\tau_i < \sigma_i$. Without less of generality assume that i is the smallest index such that $\tau_i < \sigma_i$.

Using this index i, we extract a strictly increasing sequence of $r+1$ column indices by concatenating the first i elements of τ with the remaining $r-i+1$ elements of σ:

$$\tau_1 < \tau_2 < \cdots < \tau_i < \sigma_i < \sigma_{i+1} < \cdots < \sigma_r. \tag{2}$$

We now define the specific polynomial F by mixing these $r+1$ indices. To this end, we introduce $(i, r+1-i)$-shuffles $\pi \in S_{r+1}$ acting on the indices of the chain (2).

We extend the action of these shuffles to the full indices τ and σ as follows. The first i components of $\pi(\tau)$ are given by the first i elements of

the permuted chain, while its remaining $r-i$ components are left untouched as the original $\tau_{i+1}, \ldots, \tau_r$. Symmetrically, $\pi(\sigma)$ keeps its first $i-1$ original components $\sigma_1, \ldots, \sigma_{i-1}$ fixed, while its remaining $r-i+1$ components are replaced by the elements of the permuted chain. Formally, for every $j \in \{1, \ldots, r\}$, the elements of the new index $\pi(\tau)$ are explicitly defined by:

$$\pi(\tau)_j = \begin{cases} \tau_j & \text{if } j > i \\ \tau_{\pi(j)} & \text{if } j \leq i \text{ and } \pi(j) \leq i \\ \sigma_{\pi(j)-1} & \text{if } j \leq i \text{ and } \pi(j) > i \end{cases}$$

Symmetrically, the components of $\pi(\sigma)$ are defined by:

$$\pi(\sigma)_j = \begin{cases} \sigma_j & \text{if } j < i \\ \tau_{\pi(j+1)} & \text{if } j \geq i \text{ and } \pi(j+1) \leq i \\ \sigma_{\pi(j+1)-1} & \text{if } j \geq i \text{ and } \pi(j+1) > i \end{cases}$$

Using all the $(i, r+1-i)$-shuffles of (2) we form the polynomial

$$F = \sum_{\substack{\pi \text{ is an} \\ (i,r+1-i)\text{-shuffle}}} \operatorname{sign}(\pi) p_{\pi(\sigma)} p_{\pi(\tau)}.$$

First observe that, by definition of shuffle, each term in the sum is distinct and, consequently $F \neq 0$. We now prove that $p_\sigma p_\tau = \operatorname{lt}_\prec(F)$. To see this, consider any non-identity shuffle π. By definition, there exists at least one index $j \in \{i, \ldots, r\}$ such that $\pi(\sigma_j) < \tau_i < \sigma_i$. Since the shuffle reorder the sequence increasingly we obtain that $\pi(\sigma_t) \leq \sigma_t$ for every index $t \in \{i, \ldots, r\}$, yielding $p_{\pi(\sigma)} \prec p_\sigma$. Therefore, by definition of degree reverse lexicographic order $p_\sigma p_\tau$ is indeed the initial term of F.

Finally, we prove that the polynomial F belongs to the ideal $I_{r,n} = \ker(\phi_{n,r})$. To this end we observe that

$$\phi_{n,r}(F) = \sum_{\substack{\pi \text{ is an} \\ (i,r+1-i)\text{-shuffle}}} \operatorname{sign}(\pi) \cdot \det(C|_{*,\pi(\sigma)}) \det(C|_{*,\pi(\tau)})$$

is multilinear and alternating as a function on the columns of C indexed by elements of the chain (2). Indeed, the multilinearity follows directly from the multilinearity of the determinant in its columns. To prove that $\phi_{n,r}(F)$ is an alternating function, we must show that $\phi_{n,r}(F) = 0$ whenever two columns located at the positional indices $a, b \in \{1, \ldots, r+1\}$, with $a < b$, are identical. As in Lemma A.5, we denote with $B_1(\pi) = \{\pi(1), \ldots, \pi(i)\}$ the first block of π and, with $B_2(\pi) = \{\pi(i+1), \ldots, \pi(r)\}$ the second block. We can partition the sum into two main cases:

- *Both elements go to the same subset:* Suppose a shuffle π assigns both the sequence to the same block $B_1(\pi)$ or $B_2(\pi)$. In this case, the matrix $C|_{*,\pi(\tau)}$ or $C|_{*,\pi(\sigma)}$ will contain two identical columns, causing its determinant to vanish to zero. Thus, all such shuffles contribute zero to the total sum.

- *The elements are split:* Suppose a shuffle π splits a in $B_1(\pi)$ and b in $B_2(\pi)$. Then Lemma A.5 guarantees the existence of a unique shuffle π' with sign $\operatorname{sign}(\pi') = (-1)^{b-a} \operatorname{sign}(\pi)$ and constructed by swapping the set assignments of the positions a and b from each block. Moreover, the shuffle π' sets the blocks obtained after the swap $B_1(\pi')$ and $B_2(\pi')$ in increasing order. As explained in Lemma A.5, this reordering yields a total of subsequent $b - a - 1$ transpositions. In order to reobtain the minors of the permutation π we must compose the inverses of those transpositions, yielding

  $$\det(C|_{*,\pi'(\sigma)}) \det(C|_{*,\pi'(\tau)}) = (-1)^{b-a-1} \det(C|_{*,\pi(\sigma)}) \det(C|_{*,\pi(\tau)}).$$

  Consequently, the term of $\phi_{n,r}(F)$ obtained from π' corresponds exactly to

  $$(-1)^{b-a}(-1)^{b-a-1} \det(C|_{*,\pi(\sigma)}) \det(C|_{*,\pi(\tau)}) = (-1) \det(C|_{*,\pi(\sigma)}) \det(C|_{*,\pi(\tau)}).$$

  Thus also the sum of terms in the second case contributes zero to the total sum.

This proves that $\phi_{n,r}(F)$ is an alternating multilinear form on the $r+1$ columns indexed by $\tau_1, \ldots, \tau_i, \sigma_i, \ldots, \sigma_r$. Furthermore, since these columns are vectors of length r over $\mathbb{F}$, any set of $r+1$ such vectors is strictly linearly dependent. Lemma A.6 allows us to conclude that

$$\sum_{\substack{\pi \text{ is an} \\ (i,r+1-i)\text{-shuffle}}} \operatorname{sign}(\pi) \det(C|_{*,\pi(\sigma)}) \det(C|_{*,\pi(\tau)}) = 0.$$

We now must show that the incomparable products generate the initial ideal. By contradiction let us assume that there exists a polynomial f in $I_{n,r}$ such that its leading term $\operatorname{lt}_\prec(f)$ is not a multiple of any incomparable product. Then, the variables appearing in $\operatorname{lt}_\prec(f)$ form a chain

$$\sigma^{(1)} \leq \sigma^{(2)} \leq \cdots \leq \sigma^{(t)} \tag{3}$$

where each $\sigma^{(k)} = (\sigma_1^{(k)} < \cdots < \sigma_r^{(k)})$. Let $M = \operatorname{lt}_\prec(f)$ be this leading monomial

$$M = (p_{\sigma^{(1)}})^{a_1} \cdot (p_{\sigma^{(2)}})^{a_2} \cdots (p_{\sigma^{(r)}})^{a_r}$$

for some positive exponents $a_1, \ldots, a_t \in \mathbb{N}$. When evaluated under the homomorphism $\phi_{r,n}$, the initial term $\phi_{r,n}(M)$ yields the monomial

$$(c_{1\sigma_1^{(1)}} \cdots c_{r\sigma_r^{(1)}})^{a_1} \cdot (c_{1\sigma_1^{(2)}} \cdots c_{r\sigma_r^{(2)}})^{a_2} \cdot (c_{1\sigma_1^{(t)}} \cdots c_{r\sigma_r^{(t)}})^{a_r}.$$

Since $f \in \ker(\phi_{n,r})$, the term $\phi_{r,n}(M)$ must cancel out. Any other monomial M' appearing in f that could potentially cancel this term must evaluate to the same set of column indices but permuted. By looking at the rightmost index that is changed by the non-identity permutation, the definition of `degrevlex` order ensures that M' will be strictly greater than the term M, yielding a contradiction. This contradicts the assumption that M is the leading monomial of f and concludes the proof.

□

The proof of this theorem describes a Gröbner basis for the ideal $I_{n,r}$ w.r.t. the term order $\prec$, that is

$$\{ \sum_{\substack{\pi \text{ is an} \\ (i,r+1-i)\text{-shuffle}}} \operatorname{sign}(\pi) p_{\pi(\sigma)} p_{\pi(\tau)} \mid \sigma, \tau \text{ are incomparable} \}.$$

A.8 Example

To see how this description of $I_{4,2}$ recovers the relation obtained at the start of this subsection, let us generate the Plücker relation derived by the incomparable pair $\sigma = (1,4)$ and $\tau = (2,3)$ for $r = 2$.

The comparability fails at the second index, since $\sigma_1 \leq \tau_1$ $(1 \leq 2)$ holds, but $\sigma_2 \leq \tau_2$ $(4 \leq 3)$ fails. The chain of $r+1 = 3$ elements is formed as follows:

$$\tau_1 < \tau_2 < \sigma_2 \implies 2 < 3 < 4.$$

To generate the Plücker relation, we compute the sum over all $(2,1)$-shuffles of the chain $\{2,3,4\}$. There are $\binom{3}{2} = 3$ such shuffles, which correspond to selecting two elements for τ and the remaining one for σ:

- *Assign $\{2,3\} \to \tau$ and $4 \to \sigma$ $\implies$ Sign: $(+)$ $\implies$ $+p_{14}p_{23}$.*
- *Assign $\{2,4\} \to \tau$ and $3 \to \sigma$ $\implies$ Sign: $(-)$ $\implies$ $-p_{13}p_{24}$.*
- *Assign $\{3,4\} \to \tau$ and $2 \to \sigma$ $\implies$ Sign: $(+)$ $\implies$ $+p_{12}p_{34}$.*

The sum of these terms yields the relation:

$$p_{14}p_{23} - p_{13}p_{24} + p_{12}p_{34} = 0.$$

Note that this specific relation is exactly the one obtained at the start of this chapter (where coordinates were denoted as $X_{u,v}$). Moreover, since σ and τ are strictly increasingly ordered subsets of $\{1,2,3,4\}$ of cardinality 2, the only possible incomparable pair is $\sigma = (1,4)$ and $\tau = (2,3)$ or $\sigma = (2,3)$ and $\tau = (1,4)$. The second case yields exactly the same relation and, consequently,

$$I_{4,2} = (p_{14}p_{23} - p_{13}p_{24} + p_{12}p_{34}).$$

References

[Ara+23] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. *RYDE specifications*. Tech. rep. Submission to the NIST Post-Quantum Cryptography Standardization Process (Call for Additional Digital Signatures). National Institute of Standards and Technology, 2023.

[Bar+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. "An Algebraic Attack on Rank Metric Code-Based Cryptosystems". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 64–93. ISBN: 978-3-030-45727-3.

[Ber+11a] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. *Cryptographic sponge functions*. `http://sponge.noekeon.org/CSF-0.1.pdf`. Jan. 2011.

[Ber+11b] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. *The KECCAK reference*. `http://keccak.noekeon.org/Keccak-reference-3.0.pdf`. Version 3.0. Jan. 2011.

[Bid+25] Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. "Dual Support Decomposition in the Head: Shorter Signatures from Rank SD and MinRank". In: *Advances in Cryptology – ASIACRYPT 2024*. Ed. by Kai-Min Chung and Yu Sasaki. Singapore: Springer Nature Singapore, 2025, pp. 38–69.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer International Publishing, 2015, pp. 567–576. DOI: 10.1007/978-3-319-16721-3.

[Dwo15] Morris J. Dworkin. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Tech. rep. Federal Information Processing Standards Publication (FIPS) 202. National Institute of Standards and Technology, Aug. 2015. DOI: 10.6028/NIST.FIPS.202.

[Fau+93] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. "Efficient computation of zero-dimensional Gröbner bases by change of ordering". In: *Journal of Symbolic Computation* 16.4 (1993), pp. 329–344.

[Fau99] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1-3 (1999), pp. 61–88. DOI: 10.1016/S0022-4049(99)00005-5.

[Fen24] Thibauld Feneuil. *The Polynomial-IOP Vision of the Latest MPCitH Frameworks for Signature Schemes*. Presentation at the Post-Quantum Algebraic Cryptography - Workshop 2. Institut Henri Poincaré, Paris, France, 2024.

[Fin03] Steven R. Finch. *Mathematical constants*. Cambridge University Press, 2003.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO '86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[Gol01] Oded Goldreich. *Foundations of Cryptography*. Cambridge: Cambridge University Press, Aug. 2001.

[Gor19] Elisa Gorla. *Rank-metric codes*. 2019. arXiv: 1902.02650 [cs.LG]. URL: https://arxiv.org/abs/1902.02650.

[KR00] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*. Springer, 2000.

[Laz83] Daniel Lazard. "Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations". In: *Computer Algebra*. Vol. 162. Lecture Notes in Computer Science. Berlin: Springer, 1983, pp. 146–156. DOI: 10.1007/3-540-12868-9_99.

[MS09] Ezra Miller and Bernd Sturmfels. *Combinatorial commutative algebra*. Springer, 2009, pp. 273–288.

[Ste+26] W. A. Stein et al. *Sage Mathematics Software (Version 10.8.4)*. https://www.sagemath.org. The Sage Development Team. 2026.

[SW89] Bernd Sturmfels and Neil White. "Gröbner bases and invariant theory". In: *Advances in Mathematics* 76.2 (1989), pp. 245–259. ISSN: 0001-8708. DOI: https://doi.org/10.1016/0001-8708(89)90053-4. URL: https://www.sciencedirect.com/science/article/pii/0001870889900534.