Algebraic Approach for Code Equivalence

Mohamed Ahmed Saeed

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein d'Université de Rouen Normandie et Université de Khartoum

Approche Algébrique sur L'Équivalence des Codes

Présentée et soutenue par
Mohamed A. M. SAEED TAHA

Thèse soutenue publiquement le 18/12/2017 devant le jury composé de		
M. Daniel AUGOT	DR / INRIA Saclay, Laboratoire d'Informatique de l'Ecole Polytechnique (LIX)	Rapporteur
Mme Magali BARDET	MCF / LITIS Université de Rouen Normandie	Encadrante de thèse
Mme Delphine BOUCHER	MCF/ IRMAR Université de Rennes I	Examinatrice
M. Philippe GABORIT	PR / XLIM MATHIS Université de Limoges	Examinateur
M. Ayoub OTMANI	PR / LITIS Université de Rouen Normandie	Directeur de thèse
M. Nicolas SENDRIER	DR /INRIA	Rapporteur
M. Jean-Pierre TILLICH	DR / INRIA	Examinateur

Thèse dirigée par Ayoub OTMANI, laboratoire LITIS, et Mohsin HASHIM, Faculté des sciences mathématiques et encadrée par Magali BARDET, laboratoire LITIS

Dedication

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيم

الحَمْدُ لِلَّهِ الَّذِي بِنِعْمَتِهِ تَتِمُّ الصَّالِحَاتْ

إلى أمِّي الغَالِية العَظِيمَة ...
إلى زَوْجَتِي الحَبِيبَة الوَفِيَّة ...
إلى بِنْتَيَّ الجَمِيلَتَيْن شَهْد وجَنَّات ...
إلى رُوحِ وَالِدِي الحَبِيب ...
إلى إخْوَتِي وأخَوَاتِي ...
إلى كُلِّ مَنْ عَلَّمَنِي ... سَانَدَنِي ... وشَجَّعَنِي

أهْدِي عَمَلِي

Acknowledgement

I would like to express my sincere gratitude and appreciation to my advisor Pr. Ayoub Otmani for his valuable time, patience and immense knowledge as well as for his continuous help and support. His guidance and advices were very helpful to me in all the time of research and writing of this thesis. He was always there facilitating the difficulties along the way either in research, financial or administrative.

I would like also to express my gratefulness to my co-advisor Dr. Magali Bardet for her valuable time, guidance, advices and continuous support, either scientific or administrative, during the whole project.

I would like also to thank Dr. Mohsin Hashim my second co-advisor for facilitating and directing my study at the University of Khartoum.

I thank my fellow lab-mates for the stimulating discussions, for working together, and for all the fun we have had during our study.

All love and care to my mother, my wife and all my family for their love and support. You provided me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

A special thanks to the University of Khartoum for sponsoring the major part of my study. I would like also to thank the Embassy of France in Khartoum, Campus France and LITIS Lab especially the team of Combinatorics and Algorithms for facilitating my travel as well my stay in France and partially sponsoring my study.

Contents

Introduction

The mathematical theory behind communication was initiated by Claude Shannon. In 1948 he published the article "A Mathematical Theory of Communication". In this article he discussed how to optimize the communication and storage of information. Shannon's work was considered the beginning of both Information and Coding Theory. In 1950, Richard W. Hamming wrote "Error Detecting and Error Correcting Codes", which was the first paper explicitly introducing error-correcting codes.

Coding theory continued to evolve with the evolution of communication. Nowadays coding theory is used in data compression, cryptography, error-correction and networking. It is also used in computer storage media such as CDs, DVDs, Hard disks and Flash drives, as well as communication such as satellite communication, deep space probes, Internet communications, mobile phones and networks.

The importance of coding theory and its involvement in our daily life gives the importance to study different type of codes and to explore their properties in order to select the suitable codes for suitable application. One class of codes that has gained much attention is the class of linear codes. That is due to allowance for more efficient encoding and decoding algorithms than other codes. In this class there are many types of codes with different properties and capabilities. This in turn raises the question of classification of linear codes. This will be one of our main motivations to address the code equivalence problem.

Code equivalence problem plays an important role in coding theory and code-based cryptography. That is due to its significance in classification of codes and also construction and cryptanalysis of code-based cryptosystems. It is also related to the long standing problem of graph isomorphism, a well-known problem in the world of complexity theory.

Two linear codes are said to be linearly equivalent if there exists a linear isomorphism between them that preserves the Hamming weight. Equivalent codes have the same properties such as length, dimension, minimum distance, weight distribution and correction capabilities. In coding theory the goal is to have an efficient and reliable data transmission methods. Thus classification of codes by equivalence enables to identify the codes that have the same capabilities.

In cryptography, more specifically in code-based cryptosystems, code equivalence problem is extensively used in hiding the structure of secret codes such as in the McEliece-like cryptosystems. Thus proving the difficulty of this problem enables to design secure cryptosystems. On the other hand introducing an efficient algorithm to solve the problem enables to do efficient cryptanalysis.

The link between code equivalence problem and graph isomorphism problem was established by E. Petrank and R. Roth in [70]. Graph isomorphism problem is a long standing problem in graph and complexity theory. They provide a polynomial-time reduction of graph isomorphism to permutation equivalence. This shows that graph isomorphism is easier than code equivalence problem.

Graph isomorphism problem is not known if it is in the class P nor NP−complete but it is known to be in the low hierarchy of the class NP. Thus graph isomorphism is not NP−complete unless P = NP. Petrank and Roth in [70] proved that permutation equivalence problem has the same situation. That is, it is not NP−complete unless P = NP.

The problem of finding the automorphism group of linear codes is strongly connected to the code equivalence problem. Finding automorphism group of codes is the special case of code equivalence when the two codes are equal. In fact one problem can be used to solve the other. The solution set of the equivalence of codes is a coset of the automorphism group of the codes. On the other hand the automorphism groups of equivalent codes are isomorphic. This isomorphism gives the solutions of equivalence.

There are some few algorithms that were introduced to solve code equivalence problem. Up to now there is no algorithm that solves the problem efficiently for all instances. In 1982 J. Leon introduced a method to find automorphism groups of linear codes. This method is also used to find equivalence between codes [53]. His algorithm works by finding the minimum weight codewords and retrieving the automorphisms between them. Support splitting algorithm (SSA) was introduced in 1999 by N. Senderier to find permutation equivalence between linear codes. The algorithm works by trying to find a discriminant signature for the equivalent codes using the weight enumerators of their hull. The heuristic complexity of this algorithm is exponential in the dimension of the hull which is usually small for random instances of codes [75]. Later in 2013, SSA was extended to handle diagonal equivalence after reducing the problem to permutation equivalence [78]. In this also the general complexity of code equivalence was discussed. Another attempt to solve code equivalence problem was made by I. Bouyukliev. He introduced an algorithm that reduces the test of equivalence between linear codes to a test of isometry between binary matrices [17]. In [37] Feulner introduced an algorithm to find automorphism group of linear codes that uses the idea of partition and refinement. His algorithm finds a canonical representative of linear code such that any two equivalent codes have the same representative. More details about previous work is given in Chapter 1.

Code equivalence problem is solved by some computer algebra systems such as Magma and GAP. They implement Leon's algorithm. In Magma it is used to solve permutation and diagonal equivalence for codes of small length over small prime fields and $\mathbb{F}_4$ due to the high complexity. In GAP it is used to solve permutation equivalence over binary field.

In this thesis we adopt algebraic approach to solve code equivalence problem in its permutation and diagonal versions. Up to our knowledge, this is the first attempt to solve code equivalence problem algebraically. Our contributions are contained in the chapters 3, 4 and 5.

We start by the problem of permutation equivalence. We develop algebraic model that describes the problem where we prove that the problem is well-described by this model. In a first try to solve the algebraic system we use the F_4 algorithm implemented in Magma [16] for Groebner basis computation. By using Groebner basis we can always solve the problem but as the codes get larger the computation becomes complex. One important factor that improves the solving is the rank of the linear system in our algebraic model. We use different techniques to improve the linear part of the system such as block linearization and Frobenius action. We identify instances of the problem that can be solved efficiently in polynomial-time. We find that the hull is a source of difficulty in solving the problem since it decreases the rank of the linear part. Also it seems that during Groebner basis computation other linear equations are found only in a higher degrees as the hull increases. We introduce two techniques to overcome the difficulty associated to the existence of the hull. The first is based on the

supplementary of the hull and the second is based on the closest vector problem.

We introduce a polynomial-time reduction from permutation equivalence when the hull is trivial to graph isomorphism. This reduction enables to solve graph isomorphism using our algebraic modeling for permutation equivalence. Moreover, the algebraic system of graph isomorphism has interesting properties. These properties consist in obtaining the maximum rank of the linear part of the system in addition to reducing the number of variables. For random graphs that are not isomorphic or with few isomorphisms we can find the solutions only by solving the linear system. For regular graphs or graphs with large number of isomorphisms we use additional techniques such as block linearization and guessing strategy.

We study the diagonal equivalence where we develop algebraic models for the problem and solve it using Groebner basis F_4 algorithm. Considering codes with the same parameters, the complexity of Groebner basis computation for diagonal equivalence is higher than the case of permutation equivalence. That is due to the fact that we have less linear equations and/or more variables in the model. Reduction to permutation equivalence was also considered. A new hybrid algebraic model that considers the code and its closure is developed. In this case, the solutions are binary matrices similar to permutations but with additional rows of zero we call them puncturing permutations. This model permits to reduce the number of variables and to increase the number of linear equations. Solving this system with our approaches for permutation equivalence such as block linearization and guessing strategy solve the problem over small fields, namely $\mathbb{F}_3$ and $\mathbb{F}_4$.

The rest of this thesis is organized as follows: The first chapter provides a review of the literature of the code equivalence problem in its permutation, diagonal and semilinear versions. In this we also discuss its relationship with graph isomorphism problem. Chapter 2 is a quick glance on Groebner basis where the main algorithms that compute Groebner basis are introduced and the general complexity is discussed.

Chapter 3 is about our contributions to solving permutation equivalence problem. In this we develop new algebraic model and we prove that the problem is well-described by this model. Then we use different techniques to improve the model and to identify instances that can be solved efficiently.

Chapter 4 is dedicated to our contributions to the diagonal equivalence. We distinguish between two different approaches. The first approach is to develop algebraic model that describes the problem and to use Groebner basis techniques to solve the problem. The second approach is to use the notion of the closure to reduce the diagonal equivalence to permutation equivalence. Then we use our techniques from Chapter 3 to solve the reduced problem.

Chapter 5 is an application to our techniques of solving permutation equivalence in solving graph isomorphism problem. We prove that permutation equivalence is reducible to graph isomorphism in polynomial-time when the hull of the codes is trivial. This reduction in addition to our techniques of solving permutation equivalence are used to solve graph isomorphism problem.

Chapter 1

Code Equivalence Problem

In this chapter we review the main results and findings related to code equivalence problem. We start by defining code equivalence problem, the decision and computational version. We see that there are many types of code equivalence: permutation, diagonal and semi-linear equivalence. We review the existing methods for solving code equivalence problem. We describe the support splitting algorithm (SSA) which solves permutation code equivalence problem for random linear codes with heuristic complexity that is exponential in the dimension of the hull [75, 76]. A polynomial time reduction from diagonal equivalence to permutation equivalence is introduced. The complexity of solving diagonal equivalence using SSA is low (almost polynomial) over $\mathbb{F}_3$ and $\mathbb{F}_4$ but its complexity becomes high for $q \geqslant 5$ [78].

The relation between graph isomorphism problem and code equivalence problem is discussed where a reduction from graph isomorphism problem to permutation code equivalence over $\mathbb{F}_2$ is introduced. This reduction shows that code equivalence problem is not easier than graph isomorphism problem [70]. On the other hand a reduction from special type of permutation equivalence which is called basic equivalence to bipartite graph isomorphism was introduced in [3].

Other work in finding automorphism group and classification of linear codes by finding canonical representative is considered. The general problem of code equivalence, semilinear equivalence, is considered and a polynomial time reduction from semilinear equivalence to diagonal equivalence is introduced.

Basics of graph theory are required for this chapter. The necessary definitions and required background regarding graph theory will be provided when needed.

1.1 Notions about Linear Codes

In this section we give basic notions and definitions related to linear codes. Further notions on coding theory are given in Appendix.

Definition 1.1.1. Let $\mathbb{F}_q$ be a finite field and $\mathbb{F}_q^n$ is an n-dimensional vector space over $\mathbb{F}_q$. We define a $q\text{-}ary$ $[n, k]$ *linear code* to be a k-dimensional linear subspace of $\mathbb{F}_q^n$. The parameter n is called the length of the code and k the dimension of the code. The elements of the code are called *codewords*.

Definition 1.1.2. Let $\mathscr{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. We define *a generator matrix* $\boldsymbol{G}$ of $\mathscr{C}$ to be a $k \times n$ matrix whose rows form a basis of $\mathscr{C}$. *A parity check matrix* $\boldsymbol{H}$ of $\mathscr{C}$ is an $(n-k) \times n$ matrix over $\mathbb{F}_q$ with rank $n-k$ which satisfies: For every $\boldsymbol{c} \in \mathbb{F}_q^n$, $\boldsymbol{c} \in \mathscr{C} \Leftrightarrow \boldsymbol{H}\boldsymbol{c}^T = \boldsymbol{0}$.

Definition 1.1.3. The generator matrix of an $[n,k]$ linear code is said to be in the standard (systematic) form if it is of the form $\begin{pmatrix}\boldsymbol{I}_k & \boldsymbol{A}\end{pmatrix}$, where $\boldsymbol{A}$ is a $k \times (n-k)$ matrix. The corresponding parity check matrix in the standard form is of the shape $\begin{pmatrix}-\boldsymbol{A}^T & \boldsymbol{I}_{n-k}\end{pmatrix}$.

Any generator matrix can be transformed to standard form by elementary row operations and column permutation.

Definition 1.1.4. Let $\mathscr{C}$ be an $[n,k]$ linear code with parity check matrix $\boldsymbol{H}$. The $[n,n-k]$ linear code generated by $\boldsymbol{H}$ is called the *dual code* of $\mathscr{C}$ and denoted $\mathscr{C}^\perp$. The *hull* of the code is defined to be $\mathscr{C} \cap \mathscr{C}^\perp$ and denoted by $\mathcal{H}(\mathscr{C})$.

Definition 1.1.5. The code $\mathscr{C}$ is called *weakly self-dual* if $\mathscr{C} \subseteq \mathscr{C}^\perp$ and is called *self-dual* if $\mathscr{C} = \mathscr{C}^\perp$.

Suppose $\mathscr{C}$ is an $[n,k]$ self-dual code, then n must be even and it must satisfy $n = 2k$. In case of weakly self-dual $n > 2k$. Note that, in both cases, self-dual and weakly self-dual, the code is equal to its hull.

Definition 1.1.6. Let $\boldsymbol{u} = (u_1, ..., u_n)$ and $\boldsymbol{v} = (v_1, ..., v_n)$ be vectors in $\mathbb{F}_q^n$ then we define the *Hamming distance* d_H between $\boldsymbol{u}$ and $\boldsymbol{v}$ as follows: $d_H(\boldsymbol{u}, \boldsymbol{v}) = \#\{i : u_i \neq v_i\}$. *Hamming weight* of a vector $\boldsymbol{u}$ is $wt(\boldsymbol{u}) = d_H(\boldsymbol{u}, \boldsymbol{0})$.

Thus the Hamming distance represents the number of coordinates that they differ between the two vectors or codeword where Hamming weight is the number of coordinates that differ from zero in the vector. Thus $d_H(\boldsymbol{u}, \boldsymbol{v}) = wt(\boldsymbol{u} - \boldsymbol{v})$.

Thus we define the minimum distance d of a code $\mathscr{C}$ as follows

Definition 1.1.7. The *minimum distance* d of a code $\mathscr{C}$ is defined as

$$d = \min\{d_H(\boldsymbol{u}, \boldsymbol{v}) : \boldsymbol{u}, \boldsymbol{v} \in \mathscr{C}, \boldsymbol{u} \neq \boldsymbol{v}\} = \min\{wt(\boldsymbol{c}) : \boldsymbol{c} \in \mathscr{C}, \boldsymbol{c} \neq \boldsymbol{0}\}$$

Definition 1.1.8. Let $\boldsymbol{u} = (u_1, ..., u_n)$ and $\boldsymbol{v} = (v_1, ..., v_n)$ be vectors in $\mathbb{F}_q^n$, we define the *inner product* as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^n \boldsymbol{u}_i \boldsymbol{v}_i$.

1.2 Code Equivalence Problem (CEP)

In this section we define the three types of code equivalence: permutational, diagonal and semilinear equivalence. We introduce the decision and computational versions of code equivalence. We see that CEP can also be defined in terms of matrices.

1.2.1 Permutation Equivalence Problem (PEP)

Let $\mathscr{C}$ and $\mathscr{C}'$ be two $[n,k]$ linear codes over $\mathbb{F}_q$. We say $\mathscr{C}$ and $\mathscr{C}'$ are *permutationally equivalent* if there exists a permutation $\pi \in S_n$ such that

$$\mathscr{C}' = \{(c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(n)}) : (c_1, c_2, \ldots, c_n) \in \mathscr{C}\}.$$

In other words, the codewords of $\mathscr{C}'$ results from permutation of coordinate entries of the codewords of $\mathscr{C}$. We write $\mathscr{C}' = \mathscr{C}^{\pi}$.

In terms of generator matrices, let $\boldsymbol{G}$ and $\boldsymbol{G}'$ be the generator matrices of $\mathscr{C}$ and $\mathscr{C}'$, respectively. We say $\mathscr{C}$ and $\mathscr{C}'$ are *permutationally equivalent* if there exist a $k \times k$ invertible matrix $\boldsymbol{S}$ and an $n \times n$ permutation matrix $\boldsymbol{P}$ such that

$$\boldsymbol{G}' = \boldsymbol{SGP}.$$

Note that, the representation of the code by its generator matrix is not unique. This is why we need the matrix $\boldsymbol{S}$, for example the two matrices $\boldsymbol{G}$ and $\boldsymbol{SG}$ generate the same code for any $k \times k$ invertible matrix $\boldsymbol{S}$.

Proposition 1.2.1. *Any linear code is permutationally equivalent to a linear code in the standard form.*

Proof. This is clear since the generator matrix of the standard form is obtained from the generator matrix of the code by elementary row operation in addition to column permutation. ■

The *decision* version of PEP is to decide if two given $[n, k]$ linear codes are permutationally equivalent or not where the *computational* version is to find one permutation by which the two codes are equivalent.

1.2.2 Diagonal Equivalence Problem (DEP)

Let $\mathscr{C}$ and $\mathscr{C}'$ be two $[n, k]$ linear codes over $\mathbb{F}_q$. We say $\mathscr{C}$ and $\mathscr{C}'$ are *diagonally equivalent* if there exist $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^*$ and a permutation $\pi \in S_n$ such that

$$\mathscr{C}' = \{(\alpha_1 c_{\pi(1)}, \alpha_2 c_{\pi(2)}, \ldots, \alpha_n c_{\pi(n)}) : (c_1, c_2, \ldots, c_n) \in \mathscr{C}\}.$$

In other words, the codewords of $\mathscr{C}'$ results from the codewords of $\mathscr{C}$ by permutation of coordinate entries and scaling each coordinate by a nonzero element from $\mathbb{F}_q$. Note that, when $\alpha_i = 1$ for $1 \leqslant i \leqslant n$ this is simply permutation equivalence.

In terms of generator matrices, let $\boldsymbol{G}$ and $\boldsymbol{G}'$ be generator matrices of $\mathscr{C}$ and $\mathscr{C}'$, respectively. We say $\mathscr{C}$ and $\mathscr{C}'$ are *diagonally equivalent* if there exist a $k \times k$ invertible matrix $\boldsymbol{S}$, an $n \times n$ permutation matrix $\boldsymbol{P}$ and an invertible diagonal matrix $\boldsymbol{D}$ such that

$$\boldsymbol{G}' = \boldsymbol{SGPD}.$$

Note that, one might prefer to write $\boldsymbol{G}' = \boldsymbol{SGDP}$, this depends on the order of applying the operations, scaling then permuting or the opposite, however they are both valid notations.

Again as in permutation case, the *decision* version of DEP is to decide if two given $[n, k]$ linear codes are diagonally equivalent or not where the *computational* version is to find a permutation and elements $\alpha_i, 1 \leqslant i \leqslant n$ (a diagonal matrix) by which the two codes are equivalent.

Diagonal equivalence is also termed by some authors linear equivalence as in [78] or monomial equivalence as in [43].

1.2.3 Semilinear Equivalence

Let $\mathscr{C}$ and $\mathscr{C}'$ be two $[n, k]$ linear codes over $\mathbb{F}_q$. We say $\mathscr{C}$ and $\mathscr{C}'$ are *semilinear equivalent* if there exist $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^*$, a permutation $\pi \in S_n$ and a field automorphism $\Psi : \mathbb{F}_q \to \mathbb{F}_q$ such that

$$\mathscr{C}' = \{(\alpha_1 \Psi(c_{\pi(1)}), \alpha_2 \Psi(c_{\pi(2)}), \ldots, \alpha_n \Psi(c_{\pi(n)})) : (c_1, c_2, \ldots, c_n) \in \mathscr{C}\}.$$

In other words, the codewords of $\mathscr{C}'$ results from the codewords of $\mathscr{C}$ by permutation of coordinate entries, applying the field automorphism in each coordinate entry and scaling each coordinate by a nonzero element from $\mathbb{F}_q$. Note that, when $\Psi = id$ the equivalence is diagonal. If in addition $\alpha_i = 1$ for $1 \leqslant i \leqslant n$ this is simply permutation equivalence.

It is easy to see that permutation and diagonal equivalence are linear isomorphisms between linear codes, i.e isomorphism that preserve the linearity between the two spaces. On the other hand, semilinear equivalence is a semilinear isomorphism since for $\boldsymbol{c} \in \mathscr{C}$ and $a \in \mathbb{F}_q$, $\Psi(a\boldsymbol{c}) = \Psi(a)\Psi(\boldsymbol{c}) \neq a\Psi(\boldsymbol{c})$. Note that, for a word $\boldsymbol{c} \in \mathscr{C}$, $\Psi(\boldsymbol{c}) = (\Psi(c_1), \ldots, \Psi(c_n))$.

Two equivalent codes they must have the same length, dimension, minimum distance and weight distribution. In the other direction, due to MacWilliams [60, 15], any linear isometry between linear codes that preserves the Hamming distance induces linear equivalence.

1.2.4 Automorphism Group of Linear Code

The two problems of code equivalence and finding automorphism group of linear codes are tightly related, see for example Section 3.2.2. Thus it is suitable to define the automorphism group of linear code in this section.

The set of isomorphisms that preserve the Hamming weights from $\mathscr{C}$ to itself is called *automorphism group* of $\mathscr{C}$. When we restrict ourselves to permutation automorphisms it is called *permutation group* of $\mathscr{C}$.

1.3 Support Splitting Algorithm (SSA)

In this section we review the support splitting algorithm introduced by Sendrier in 1999 in [75, 76]. SSA algorithm solves the code equivalence problem in its permutation version for linear codes that have small non-trivial hull and trivial permutation group. The SSA algorithm finds the equivalence between two linear codes by computing the weight enumerators of their hull. The algorithm complexity is exponential in the dimension of the hull. The hull of random linear code is of small dimension thus the run time for the algorithm is expected to be small. One of the drawbacks of the algorithm is that its complexity becomes intractable for codes with large hull dimension specially self-dual and weakly self-dual codes. That is because the algorithm computes the weight enumerators which is known to be `NP-hard` problem [11].

The weight enumerator is invariant by permutation equivalence, i.e if two codes are permutationally equivalent they must have the same weight distribution. Unfortunately the opposite is not true. If the weight enumerators for two linear codes are equal, it does not imply that the two codes are equivalent.

This will raise another issue about the use of weight enumerator to decide the equivalence. Practically for random codes this is true with high probability that is why SSA works successfully in this case.

Next we show that the hull of the permutation equivalent codes are also equivalent by the same permutation.

Proposition 1.3.1. *Two linear codes are permutation equivalent if and only if their dual codes are permutation equivalent by the same permutation.*

Proof. Let $\mathscr{C}$ be a linear code, $\boldsymbol{P}$ an $n \times n$ permutation matrix, and $\mathscr{C}\boldsymbol{P}$ a permutation equivalent linear code of $\mathscr{C}$, we show that $(\mathscr{C}\boldsymbol{P})^{\perp} = \mathscr{C}^{\perp}\boldsymbol{P}$.

$\boldsymbol{x} \in (\mathscr{C}\boldsymbol{P})^{\perp}$ if and only if for all $\boldsymbol{c} \in \mathscr{C}$, $\langle \boldsymbol{x}, \boldsymbol{cP} \rangle = \boldsymbol{0}$. But $\langle \boldsymbol{x}, \boldsymbol{cP} \rangle = \boldsymbol{x}(\boldsymbol{cP})^T = \boldsymbol{x}\boldsymbol{P}^T\boldsymbol{c}^T = (\boldsymbol{x}\boldsymbol{P}^T)\boldsymbol{c}^T = \langle \boldsymbol{x}\boldsymbol{P}^T, \boldsymbol{c} \rangle$. Hence $\boldsymbol{x} \in (\mathscr{C}\boldsymbol{P})^{\perp}$ if and only if $\boldsymbol{x}\boldsymbol{P}^T \in \mathscr{C}^{\perp} \Leftrightarrow \boldsymbol{x} \in (\mathscr{C}^{\perp})(\boldsymbol{P}^T)^{-1} = (\mathscr{C}^{\perp})\boldsymbol{P}$, since $\boldsymbol{P}$ is a permutation.

The other direction is obvious since the dual of the dual is the code itself. ■

The following proposition shows that the hull of the code, the subfield subcodes, and the trace codes are stable by permutation equivalence, see Appendix A for the definitions.

Proposition 1.3.2. *Let $\mathscr{C}$ be linear code $\mathcal{H}(\mathscr{C}), \mathscr{C}|_{\mathbb{F}_p}$, and $Tr(\mathscr{C})$ are the hull, the subfield subcode and the trace code, respectively. Then we have*

1. $\mathcal{H}(\mathscr{C})^{\sigma} = \mathcal{H}(\mathscr{C}^{\sigma})$
2. $(\mathscr{C}|_{\mathbb{F}_p})^{\sigma} = \mathscr{C}^{\sigma}|_{\mathbb{F}_p}$
3. $(Tr(\mathscr{C}))^{\sigma} = Tr(\mathscr{C}^{\sigma})$.

Proof.

1. $\mathcal{H}(\mathscr{C})^{\sigma} = (\mathscr{C} \cap \mathscr{C}^{\perp})^{\sigma} = \mathscr{C}^{\sigma} \cap (\mathscr{C}^{\perp})^{\sigma} = \mathscr{C}^{\sigma} \cap (\mathscr{C}^{\sigma})^{\perp} = \mathcal{H}(\mathscr{C}^{\sigma})$.
2. $(\mathscr{C}|_{\mathbb{F}_p})^{\sigma} = (\mathscr{C} \cap \mathbb{F}_q^n)^{\sigma} = \mathscr{C}^{\sigma} \cap \mathbb{F}_q^n = \mathscr{C}^{\sigma}|_{\mathbb{F}_p}$.
3. $(Tr(\mathscr{C}))^{\sigma} = \{(Tr(c))^{\sigma} : c \in \mathscr{C}\} = \{Tr(c^{\sigma}) : c \in \mathscr{C}\} = Tr(\mathscr{C}^{\sigma})$.

■

Proposition 1.3.2 shows that any solution for the equivalence problem between two linear codes will be also a valid solution for the equivalence problem between their hulls, subfield subcodes, and their trace codes. Unfortunately, the opposite is not always true as we show by the next example.

Example 1.3.3. Consider the two $[5, 3]$ linear codes $\mathscr{A}$ and $\mathscr{B}$ defined over $\mathbb{F}_{2^2}$ with primitive element t and generated by

$$\boldsymbol{G}_{\mathscr{A}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & t \\ 0 & 0 & 1 & 1 & t^2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{G}_{\mathscr{B}} = \begin{pmatrix} 1 & 0 & 0 & 1 & t^2 \\ 0 & 1 & 0 & 1 & t \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

These two codes are equivalent and the solution set of equivalence has order 2. Looking at their hulls, they are $[5, 1]$ linear codes and equivalent by 4 permutations. Their subfield subcodes are $[5, 2]$ linear codes and equivalent by 6 permutations. Finally their trace codes are $[5, 4]$ linear codes and they are equivalent by 12 permutations.

In the example above the solution set of the equivalence for $\mathscr{A}$ and $\mathscr{B}$ is strictly included in the solution set of the equivalence of their hull, subfield subcodes and trace codes.

Before we proceed to the algorithm we introduce some definitions.

Definition 1.3.4. Let $\mathcal{L}$ denotes the set of all codes. An invariant over a set E is defined to be a mapping $\mathcal{L} \to E$ such that any two permutation equivalent codes take the same value.

Definition 1.3.5. Let $I_n = \{1, 2, \ldots, n\}$ be the set used to index the coordinates of the words of $\mathbb{F}_q^n$. A signature S over a set F maps a code $\mathscr{C}$ of length n and $i \in I_n$ into an element of F such that for all permutations σ on I_n, $S(\mathscr{C}, i) = S(\sigma(\mathscr{C}), \sigma(i))$.

Assume that we relate the indexes i and j if $S(\mathscr{C}, i) = S(\mathscr{C}, j)$ this induces a partition in the index set I_n.

Definition 1.3.6. Let $\mathscr{C}$ be a code of length n. A signature S is said to be discriminant for $\mathscr{C}$ if there exist i and j in I_n such that $S(\mathscr{C}, i) \neq S(\mathscr{C}, j)$ and is said to be fully discriminant if for all i and j distinct in I_n, $S(\mathscr{C}, i) \neq S(\mathscr{C}, j)$.

Note that the fully discriminant signature exists if and only if the permutation group of $\mathscr{C}$ is trivial. Thus a fully discriminant signature does not exist for codes that have non-trivial permutation group.

SSA Algorithm

1: **Input**: Two permutation equivalent codes $\mathscr{A}$ and $\mathscr{B}$; fully discriminant signature S for $\mathscr{A}$.
2: **Output**: Permutation σ.
3: **for** $i, j \in I_n$ **do**
4: **if** $S(\mathscr{A}, i) = S(\mathscr{B}, j)$ **then**
5: $\sigma(i) = j$.
6: **end if**
7: **end for**

Next we give simple examples to illustrate the support splitting algorithm. These examples are taken from [75].

Example 1.3.7. We consider this simple example of two equivalent codes $\mathscr{C} = \{1110, 0111, 1010\}$ and $\mathscr{C}' = \{0011, 1011, 1101\}$. To obtain fully discriminant signature we puncture the code each time at one position and use the weight enumerator denoted by $\mathcal{W}(.)$.

$$\begin{array}{ll}
\mathscr{C}_1 = \{\underline{0}110, \underline{0}111, \underline{0}010\} & \mathcal{W}(\mathscr{C}_1) = X + X^2 + X^3 \\
\mathscr{C}_2 = \{1\underline{0}10, 0\underline{0}11\} & \mathcal{W}(\mathscr{C}_2) = 2X^2 \\
\mathscr{C}_3 = \{11\underline{0}0, 01\underline{0}1, 10\underline{0}0\} & \mathcal{W}(\mathscr{C}_3) = X + 2X^2 \\
\mathscr{C}_4 = \{111\underline{0}, 011\underline{0}, 101\underline{0}\} & \mathcal{W}(\mathscr{C}_4) = 2X^2 + X^3
\end{array}$$

and for $\mathscr{C}'$

$$\begin{array}{ll}
\mathscr{C}'_1 = \{\underline{0}011, \underline{0}101\} & \mathcal{W}(\mathscr{C}'_1) = 2X^2 \\
\mathscr{C}'_2 = \{0\underline{0}11, 1\underline{0}11, 1\underline{0}01\} & \mathcal{W}(\mathscr{C}'_2) = 2X^2 + X^3 \\
\mathscr{C}'_3 = \{00\underline{0}1, 10\underline{0}1, 11\underline{0}1\} & \mathcal{W}(\mathscr{C}'_3) = X + X^2 + X^3 \\
\mathscr{C}'_4 = \{001\underline{0}, 101\underline{0}, 110\underline{0}\} & \mathcal{W}(\mathscr{C}'_4) = X + 2X^2
\end{array}$$

Thus we can easily find the permutation between the two codes which is: $\sigma(1) = 3, \sigma(2) = 1, \sigma(3) = 4, \sigma(4) = 2$

It is not always possible to find a fully discriminant signature for the equivalent codes or it is too complex to compute, for that reason the notion of refinement is introduced. We illustrate bellow how to construct full discriminant signature from a discriminant signature by the notion of refinement.

Example 1.3.8. Consider the two codes $\mathscr{C}$ and $\mathscr{C}'$ as

$$\mathscr{C} = \{01101, 01011, 01110, 10101, 11110\} \text{ and } \mathscr{C}' = \{10101, 00111, 10011, 11100, 11011\}.$$

We get the following:

$$\begin{array}{ll}
\mathscr{C}_1 = \{\underline{0}1101, \underline{0}1011, \underline{0}1110, \underline{0}0101\} & \mathcal{W}(\mathscr{C}_1) = X^2 + 3X^3 \\
\mathscr{C}_2 = \{0\underline{0}101, 0\underline{0}011, 0\underline{0}110, 1\underline{0}101, 1\underline{0}110\} & \mathcal{W}(\mathscr{C}_2) = 3X^2 + 2X^3 \\
\mathscr{C}_3 = \{01\underline{0}01, 01\underline{0}11, 01\underline{0}10, 10\underline{0}01, 11\underline{0}10\} & \mathcal{W}(\mathscr{C}_3) = 3X^2 + 2X^3 \\
\mathscr{C}_4 = \{011\underline{0}1, 010\underline{0}1, 011\underline{0}0, 101\underline{0}1, 111\underline{0}0\} & \mathcal{W}(\mathscr{C}_4) = 2X^2 + 3X^3 \\
\mathscr{C}_5 = \{0110\underline{0}, 0101\underline{0}, 0111\underline{0}, 1010\underline{0}, 1111\underline{0}\} & \mathcal{W}(\mathscr{C}_5) = 2X^2 + X^3 + X^4
\end{array}$$

and for $\mathscr{C}'$

$$\begin{array}{ll}
\mathscr{C}'_1 = \{\underline{0}0101, \underline{0}0111, \underline{0}0011, \underline{0}1100, \underline{0}1011\} & \mathcal{W}(\mathscr{C}'_1) = 3X^2 + 2X^3 \\
\mathscr{C}'_2 = \{1\underline{0}101, 0\underline{0}111, 1\underline{0}011, 1\underline{0}100\} & \mathcal{W}(\mathscr{C}'_2) = X^2 + 3X^3 \\
\mathscr{C}'_3 = \{10\underline{0}01, 00\underline{0}11, 10\underline{0}11, 11\underline{0}00, 11\underline{0}11\} & \mathcal{W}(\mathscr{C}'_3) = 3X^2 + X^3 + X^4 \\
\mathscr{C}'_4 = \{101\underline{0}1, 001\underline{0}1, 100\underline{0}1, 111\underline{0}0, 110\underline{0}1\} & \mathcal{W}(\mathscr{C}'_4) = 2X^2 + 3X^3 \\
\mathscr{C}'_5 = \{1010\underline{0}, 0011\underline{0}, 1001\underline{0}, 1110\underline{0}, 1101\underline{0}\} & \mathcal{W}(\mathscr{C}'_5) = 3X^2 + 2X^3
\end{array}$$

Thus we have $\sigma(1) = 2, \sigma(4) = 4, \sigma(5) = 3$ and $\sigma(\{2, 3\}) = \{1, 5\}$. We found that $\sigma(\{2, 3\}) = \{1, 5\}$ thus we can not distinguish if position 2 is mapped to 1 or 5 by σ. The same is said about position 3, thus the signature is not fully discriminant. We refine the signature to obtain a full discriminant signature by doing the following. In this case one refinement is sufficient:

$$\begin{array}{ll}
\mathscr{C}_{\{1,2\}} = \{\underline{00}101, \underline{00}011, \underline{00}110\} & \mathcal{W}(\mathscr{C}_{\{1,2\}}) = 3X^2 \\
\mathscr{C}_{\{1,3\}} = \{\underline{0}1\underline{0}01, \underline{0}1\underline{0}11, \underline{0}1\underline{0}10, \underline{0}0\underline{0}01\} & \mathcal{W}(\mathscr{C}_{\{1,3\}}) = X + 2X^2 + X^3 \\
\mathscr{C}'_{\{2,1\}} = \{\underline{00}101, \underline{00}111, \underline{00}011, \underline{00}100\} & \mathcal{W}(\mathscr{C}'_{\{2,1\}}) = X + 2X^2 + X^3 \\
\mathscr{C}'_{\{2,5\}} = \{1\underline{0}10\underline{0}, 0\underline{0}11\underline{0}, 1\underline{0}01\underline{0}\} & \mathcal{W}(\mathscr{C}'_{\{2,5\}}) = 3X^2
\end{array}$$

Thus we have $\sigma(2) = 5$ and $\sigma(3) = 1$

Of course computing the weight enumerators has high complexity especially when the code is large, in fact it is `NP-hard` problem [11], for this reason SSA algorithm uses the hull of the code instead. It has been proven that the dimension of the hull for random linear code is very small with high probability [74]. The number of refinements to obtain fully discriminant signature is not known, for that there is nothing precise about the complexity of the algorithm. The heuristic complexity is $\mathcal{O}(n^3 + q^h n^2 \log n)$ where h is the dimension of the hull.

One factor that speeds up the computation when working with the hull to find a discriminant signature is that the hull of punctured code can be deduced from the hull of the code without need for the whole computation [75].

One disadvantage is that the signature built from the hull is less discriminant than the signature built from the code since the hull is usually of small size, this can be handled by extra cycles of refinements

of the signature. Another disadvantage of using the hull to decide equivalence is that the permutations set that makes the hull of codes equivalent may be strictly larger than the permutations set that makes the codes equivalent, see Example 1.3.3.

1.3.1 Issues about the SSA Algorithm

The algorithm uses the weight enumerator of the hull for signature. The weight enumerator computation in general is `NP-hard` but computing weight enumerator of the hull for random codes is achievable with high probability. For codes with large hull such as self-dual and weakly self-dual codes still the algorithm is not practical.

Using the hull has some disadvantages since not any solution for the equivalence of the hulls is a solution for the equivalence of the codes. Another disadvantage of using the hull is that it increases the number of needed refinements to reach the fully discriminant signature since the signature that is built from the hull is less discriminant [75].

Obtaining a fully discriminant signature is not always achievable, for example codes with non-trivial permutation group cannot have fully discriminant signature.

Although the algorithm achieves good complexity for linear codes with small hull there is no clear bound in the number of needed refinements for the algorithm to terminate. Another important issue about SSA is that the algorithm is expected to have high complexity for structured codes such as Reed-Muller codes and cyclic codes. That is because Reed-Muller codes have large hull (see Section A.5.4) and also almost all cyclic codes have large hull [80].

1.4 Diagonal Equivalence

In this section we discuss the general case of code equivalence which we call diagonal equivalence. In this case not only a permutation that acts on the code is considered but also an invertible diagonal matrix. We distinguish the work of Sendrier and Simos in [78] where they give a reduction from diagonal equivalence to permutation equivalence by using the notion of closure of the code then they use the SSA to solve the reduced problem. The monomial matrix can be easily recovered from the permutation between closures.

Definition 1.4.1. Let $\boldsymbol{A} = (a_{i,j})$ and $\boldsymbol{B} = (b_{i,j})$ be two matrices of size $k_{\boldsymbol{A}} \times n_{\boldsymbol{A}}$ and $k_{\boldsymbol{B}} \times n_{\boldsymbol{B}}$ respectively. The Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $k_{\boldsymbol{A}} k_{\boldsymbol{B}} \times n_{\boldsymbol{A}} n_{\boldsymbol{B}}$ matrix defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = (a_{i,j}\boldsymbol{B}) .$$

Definition 1.4.2. Let $\mathscr{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$ with primitive element α. Let $\boldsymbol{G}$ be a generator matrix of $\mathscr{C}$ and $\boldsymbol{a} = (\alpha, \ldots, \alpha^{q-1})$. We define the closure $\widetilde{\mathscr{C}}$ of $\mathscr{C}$ to be the $[n(q-1), k]$ linear code

$$\widetilde{\mathscr{C}} = \{\boldsymbol{c} \otimes \boldsymbol{a} : \boldsymbol{c} \in \mathscr{C}\}.$$

The generator matrix of $\widetilde{\mathscr{C}}$ is given by $\widetilde{\boldsymbol{G}} = \boldsymbol{G} \otimes \boldsymbol{a}$.

The closure duplicates the coordinates $q-1$ times with different scalars from $\mathbb{F}_q^*$.

Example 1.4.3. Consider the $[3, 2]$ linear code $\mathscr{C}$ over $\mathbb{F}_4$ (with primitive element α) that is generated by

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & \alpha^2 \\ 0 & \alpha^2 & 1 \end{pmatrix}.$$

The closure $\widetilde{\mathscr{C}}$ is the $[9, 2]$ linear code that is generated by

$$\widetilde{\boldsymbol{G}} = \boldsymbol{G} \otimes (\alpha, \alpha^2, 1) = \begin{pmatrix} \alpha & \alpha^2 & 1 & 0 & 0 & 0 & 1 & \alpha & \alpha^2 \\ 0 & 0 & 0 & 1 & \alpha & \alpha^2 & \alpha & \alpha^2 & 1 \end{pmatrix}.$$

1.4.1 Properties of the Closure

The closure has some interesting properties that enable to find the solutions for diagonal equivalence between linear codes. These properties are discussed next.

Proposition 1.4.4. *[78] If two linear codes are diagonally equivalent then their closures are permutationally equivalent.*

Proof. Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonally equivalent $[n, k]$ linear codes over $\mathbb{F}_q$ with generator matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Then we have $\boldsymbol{B} = \boldsymbol{SAPD}$. Let $\widetilde{\mathscr{A}}$ and $\widetilde{\mathscr{B}}$ be their closure with generator matrices $\widetilde{\boldsymbol{A}}$ and $\widetilde{\boldsymbol{B}}$, respectively. We can look at $\boldsymbol{A}$ as an n blocks of columns each of size $q-1$, that is for $1 \leqslant d \leqslant n$, $\mathcal{A}_d = \{\alpha^j \boldsymbol{A}_d : 1 \leqslant j \leqslant q-1\}$, recall that $\boldsymbol{A}_d$ is column d of the matrix $\boldsymbol{A}$. Similarly we can define $\mathcal{B}_d$. Assume that $\boldsymbol{A}_i$, column i in $\boldsymbol{A}$, is sent to $\boldsymbol{B}_j$ by $\boldsymbol{PD}$ with a scalar α^l. This is true if and only if block $\mathcal{A}_i$ is mapped to block $\mathcal{B}_j$ with the columns inside the blocks are mapped using the cyclic shift $\alpha^{l+a}\boldsymbol{A}_i \mapsto \alpha^a \boldsymbol{B}_j, 1 \leqslant a \leqslant q-1$. ■

Example 1.4.5. Consider the code $\mathscr{A}$ over $\mathbb{F}_4$ with generator matrix $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 \end{pmatrix}$. Let α be the primitive element of $\mathbb{F}_4$,

$$\boldsymbol{P} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

and $\boldsymbol{D} = diag(\alpha^3, \alpha, \alpha^2)$. Let $\mathscr{B}$ be the diagonally equivalent code to $\mathscr{A}$ that is generated by $\boldsymbol{B} = \boldsymbol{APD} = \begin{pmatrix} \alpha^3\boldsymbol{A}_2 & \alpha\boldsymbol{A}_3 & \alpha^2\boldsymbol{A}_1 \end{pmatrix}$. Then we have $\widetilde{\mathscr{A}}$ and $\widetilde{\mathscr{B}}$ are respectively generated by:

$$\widetilde{\boldsymbol{A}} = \begin{pmatrix} \alpha\boldsymbol{A}_1 & \alpha^2\boldsymbol{A}_1 & \boldsymbol{A}_1 & \alpha\boldsymbol{A}_2 & \alpha^2\boldsymbol{A}_2 & \boldsymbol{A}_2 & \alpha\boldsymbol{A}_3 & \alpha^2\boldsymbol{A}_3 & \boldsymbol{A}_3 \end{pmatrix}$$

$$\widetilde{\boldsymbol{B}} = \begin{pmatrix} \alpha\boldsymbol{A}_2 & \alpha^2\boldsymbol{A}_2 & \boldsymbol{A}_2 & \alpha^2\boldsymbol{A}_3 & \boldsymbol{A}_3 & \alpha\boldsymbol{A}_3 & \boldsymbol{A}_1 & \alpha\boldsymbol{A}_1 & \alpha^2\boldsymbol{A}_1 \end{pmatrix}.$$

The permutation between the closures is given by $\pi = (1\ 8\ 4)(2\ 9\ 5)(3\ 7\ 6)$ and satisfies $\widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{A}}^{\pi}$.

The following proposition shows that the closure is weakly self-dual for all codes defined over $\mathbb{F}_q$ with $q \geqslant 4$, see also [78].

Proposition 1.4.6. *Consider $\mathbb{F}_q$ with primitive element α. Let $\widetilde{\mathscr{A}}$ be a closure of an $[n, k]$ linear code $\mathscr{A}$ over $\mathbb{F}_q$ with hull of dimension h and let $\widetilde{h}$ be the dimension of the hull of $\widetilde{\mathscr{A}}$ then*

1. *$\widetilde{h} = h$ when $q = 3$.*
2. *For $q \geqslant 4$, $\widetilde{\mathscr{A}}$ is weakly self-dual code*

Proof. Let $\mathcal{H}(\mathscr{A})$ be the hull of $\mathscr{A}$ and let $\boldsymbol{x} \in \mathcal{H}(\mathscr{A})$ thus for all $\boldsymbol{y} \in \mathscr{A}$ we have $\boldsymbol{x}.\boldsymbol{y} = \sum_{i=1}^{n} x_i y_i = 0$. On the other hand, we denote by $\widetilde{\boldsymbol{v}} \in \widetilde{\mathscr{A}}$ the element that corresponds to $\boldsymbol{v} \in \mathscr{A}$.

$$\widetilde{\boldsymbol{x}}.\widetilde{\boldsymbol{y}} \;=\; \sum_{j=1}^{q-1}(\alpha^j)^2 \sum_{i=1}^{n} x_i y_i$$

In $\mathbb{F}_3$ we have $\lambda^2 = 1$ for all $\lambda \in \mathbb{F}_3^*$, thus we have $\widetilde{\boldsymbol{x}}.\widetilde{\boldsymbol{y}} = (q-1)\sum_{i=1}^{n} x_i y_i$ and $\widetilde{\boldsymbol{x}}.\widetilde{\boldsymbol{y}} = 0$ implies $\boldsymbol{x}.\boldsymbol{y} = 0$. Hence in $\mathbb{F}_3$, $\widetilde{\boldsymbol{x}} \in \mathcal{H}(\widetilde{\mathscr{A}})$ if and only if $\boldsymbol{x} \in \mathcal{H}(\mathscr{A})$ and thus $\widetilde{h} = h$.

To prove the second part

$$\widetilde{\boldsymbol{x}}.\widetilde{\boldsymbol{y}} \;=\; \sum_{j=1}^{q-1}(\alpha^j)^2 \sum_{i=1}^{n} x_i y_i$$

where $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}}$ are any two elements from $\widetilde{\mathscr{A}}$. Since we have $\sum_{j=1}^{q-1}(\alpha^j)^2 = 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \widetilde{\mathscr{A}}$ we have $\widetilde{\boldsymbol{x}}.\widetilde{\boldsymbol{y}} = 0$. Thus $\widetilde{\mathscr{A}}$ is weakly self-dual. ■

1.4.2 Issues and Complexity of Computation

In [78] the problem of diagonal equivalence is reduced to permutation equivalence and SSA algorithm is used to solve the reduced problem where the weight distribution of the hull is used to build the signature. Same issues mentioned in Section 1.3.1 about SSA for permutation equivalence hold here. For random linear codes over $\mathbb{F}_q$ with $q = 3$ with small hull the complexity of solving the equivalence is low. Over $\mathbb{F}_4$ they used different inner product which is called Hermitian inner product in order to avoid weakly self-dual codes. This trick was not successful for larger q. For $q \geqslant 5$ the complexity is high since all closures are weakly self-dual even if the original code has trivial hull. In [78] they conjecture that for a given $q \geqslant 5$, the decision and computational problems of the diagonal equivalence over $\mathbb{F}_q$ are hard for almost all instances. This conjecture is still questionable since it is related to a specific way of building the signature from the weight enumerator using the hull.

1.4.3 Reduction of Semilinear Equivalence

The semilinear code equivalence problem is reducible to diagonal equivalence in polynomial time. That is because if $q = p^m$ we have exactly m distinct field automorphisms thus if we have an algorithm for solving diagonal equivalence, repeating this algorithm at most m times can solve the related semilinear equivalence.

1.5 Code Equivalence and Graph Isomorphism

In this section we study the relationship between code equivalence and graph isomorphism problems. Graph isomorphism problem is a well-known problem in graph and complexity theory. It is known to be in the low hierarchy of class `NP` but it is not known whether it is in class `P` or `NP-complete`. In practice, very often graph isomorphism problem can be solved efficiently but still there are many hard instances. The two problems of graph isomorphism and code equivalence are strongly related where a reduction from graph isomorphism problem to permutation code equivalence problem is introduced in [70]. This shows that code equivalence problem cannot be easier than graph isomorphism problem. On the other hand a reduction of special type of permutation code equivalence which is called basic equivalence to isomorphism of hypergraphs is introduced in [3].

1.5.1 Reduction of Graph Isomorphism to Code Equivalence

In this section we summarize the results of Petrank and Roth [70]. They show that under the assumption of polynomial-time hierarchy does not collapse code equivalence problem is not `NP-complete`. They also relate the code equivalence problem to graph isomorphism problem where they present a polynomial-time reduction from graph isomorphism to code equivalence. This shows that code equivalence problem is at least as hard as graph isomorphism. Thus if one is able to find an efficient algorithm for deciding code equivalence then he can settle the long-standing open problem of graph isomorphism.

In their work they proved that code equivalence is not `NP-complete` (unless `P = NP`) using techniques of interactive proof that will not be presented here but rather we will go through the detail of the reduction from graph isomorphism to code equivalence.

Definition 1.5.1. A graph $\mathcal{G} = (V, E)$ is called *simple graph* if the set of vertices $V \neq \emptyset$ and E is a set of unordered pairs of elements of V representing edges, and E has the following properties:

- Edges are undirected.
- No edge from a vertex to itself.
- No parallel edges.

If the edges have directions the graph is called *directed graph* or *digraph*; if there is an edge from a vertex to itself it is called *graph with self-loop*; and if the graph contains parallel edges it is called *multigraph*. Unless it is mentioned explicitly, "graph" refers to a simple graph.

Definition 1.5.2. Let $\mathcal{G}_1 = (V, E_1)$ and $\mathcal{G}_2 = (V, E_2)$ be two graphs with the same set of vertices V and sets of edges E_1 and E_2, respectively. We say $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if there exists a permutation (isomorphism) $\pi : V \to V$ such that $\{u, v\} \in E_1$ if and only if $\{\pi(u), \pi(v)\} \in E_2$.

Definition 1.5.3. Consider the graph $\mathcal{G} = (V, E)$, the incident matrix of $\mathcal{G}$ (denoted by $\boldsymbol{D}$) is the $|E| \times |V|$ matrix satisfying: for every $e \in E$ and $v \in V$ we have

$$\boldsymbol{D}_{e,v} = \begin{cases} 1 & \text{if } e = \{v, u\} \text{ for some } u \in V \\ 0 & \text{otherwise} \end{cases}$$

From the definition of graph isomorphism and incidence matrix one can see the following proposition.

Proposition 1.5.4. *Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if and only if their incidence matrices differ only by permutation of rows and columns.*

Next we present a mapping ϕ from the set of all graphs to the set of generator matrices over $\mathbb{F}_2$ such that two graphs are isomorphic if and only if their respective images under ϕ are code equivalent. For a graph $\mathcal{G} = (V, E)$, $\phi(\mathcal{G})$ is a generator matrix $\boldsymbol{M}$ with the following properties:

1. $\boldsymbol{M}$ is over $\mathbb{F}_2$ with size $|E| \times (3|E| + |V|)$.
2. Each row in $\boldsymbol{M}$ has hamming weight $\leqslant 5$.
3. The sum of each two rows or more is a vector of hamming weight $\geqslant 6$.

Note that the second and third conditions guarantee that the code generated by $\boldsymbol{M}$ has exactly $|E|$ codewords of weight $\leqslant 5$ which are the rows of $\boldsymbol{M}$ and $\boldsymbol{M}$ is unique generator matrix for this code with properties $1-3$ (up to row swap).

We define $\phi(\mathcal{G}) = [\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{D}]$ where $\boldsymbol{I}$ is $|E| \times |E|$ identity matrix and $\boldsymbol{D}$ is the incident matrix for $\mathcal{G}$. It can be easily seen that the reduction can be done in polynomial time. Now assume we have two isomorphic graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with $\phi(\mathcal{G}_1) = [\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{D}_1]$ and $\phi(\mathcal{G}_2) = [\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{D}_2]$. It is clear that the two matrices have the same size and one can be obtained from the other by doing rows and columns swap thus the two codes generated by them are equivalent.

Now assume $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ with $\phi(\mathcal{G}_1) = \boldsymbol{M}_1$ and $\phi(\mathcal{G}_2) = \boldsymbol{M}_2$ and assume $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ generate equivalent codes. This means the two matrices have the same size i.e $|E_1| = |E_2|$ and $|V_1| = |V_2|$ (We can assume $V_1 = V_2 = V$). Since the two codes are equivalent we can write $\boldsymbol{M}_2 = \boldsymbol{S}\boldsymbol{M}_1\boldsymbol{P}$ where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{S}$ is nonsingular matrix. Since $\boldsymbol{M}_2$ and $\boldsymbol{M}_1\boldsymbol{P}$ generate the same code and satisfy conditions $1-3$ they are the same matrix up to row permutation. Thus $\boldsymbol{S}$ is a permutation matrix. Now we can write $[\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{I}|\boldsymbol{D}_2] = \boldsymbol{M}_2 = \boldsymbol{S}\boldsymbol{M}_1\boldsymbol{P} = [\boldsymbol{S}|\boldsymbol{S}|\boldsymbol{S}|\boldsymbol{S}\boldsymbol{D}_1]\boldsymbol{P}$. Thus $\boldsymbol{P}$ can be written:

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{S}^{-1} & & & \\ & \boldsymbol{S}^{-1} & & \\ & & \boldsymbol{S}^{-1} & \\ & & & \boldsymbol{T} \end{pmatrix}$$

where $\boldsymbol{T}$ is $|V| \times |V|$ permutation matrix. Note that the two graphs are isomorphic since $\boldsymbol{D}_2 = \boldsymbol{S}\boldsymbol{D}_1\boldsymbol{T}$ and the isomorphism is the permutation associated with $\boldsymbol{T}$.

In conclusion, any instance of graph isomorphism problem can be reduced to permutation code equivalence problem over $\mathbb{F}_2$. Thus code equivalence problem is at least as hard as graph isomorphism problem.

1.5.2 Permutation Equivalence and Bipartite Graphs

In this section we consider the work of Babai et al. in [3] which discusses the relationship between a special case of permutation code equivalence which is called basic equivalence and the isomorphism of bipartite graphs. In this work a reduction from basic equivalence to the isomorphism of bipartite graphs is introduced and complexity bound is found.

Definition 1.5.5. Let $\mathscr{A}$ and $\mathscr{B}$ be two $[n, k]$ linear codes such that $\mathscr{B} = \mathscr{A}^{\sigma}$. We call this basic equivalence if $\sigma \in S_n$ can be decomposed into two disjoint permutations $\sigma_1 \in S_k$ that acts in the first k coordinates and $\sigma_2 \in S_{n-k}$ that acts on the last $n-k$ coordinates.

Definition 1.5.6. A bipartite graph, also called a bigraph, is graph that its vertices can be decomposed into two disjoint sets such that no two vertices within the same set are adjacent.

Definition 1.5.7. A hypergraph $\mathcal{H}$ is a pair $\mathcal{H} = (X, E)$ where X is a set of vertices, and E is a set of non-empty subsets of X called hyperedges or edges. Therefore, E is a subset of $\mathcal{P}(X) \setminus \{\emptyset\}$, where $\mathcal{P}(X)$ is the power set of X.

In other words, a hypergraph is a generalization of a graph in which an edge can join any number of vertices. A hypergraph $\mathcal{H}$ may be represented by a bipartite graph B as follows: the sets X and E

are the partitions of B vertices, and (x_1, e_1) are connected with an edge if and only if vertex x_1 is contained in edge e_1 in $\mathcal{H}$.

Definition 1.5.8. The biadjacency matrix of a bipartite graph (U, V, E) is a $(0, 1)$-matrix of size $|U| \times |V|$ that has a one for each pair of adjacent vertices and a zero for nonadjacent vertices.

Biadjacency matrices may be used to describe equivalences between bipartite graphs, hypergraphs, and directed graphs. The biadjacency matrices of bipartite graphs are exactly the incidence matrices of the corresponding hypergraphs.

Theorem 1.5.9. *[3] Any basic equivalence problem can be reduced to the bipartite graph isomorphism problem.*

Proof. Consider the two $[n, k]$ linear codes $\mathscr{A}$ and $\mathscr{B}$ over $\mathbb{F}_q$ which are basically equivalent. Let $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ their generator matrices then the equivalence between $\mathscr{A}$ and $\mathscr{B}$ can be written as

$$\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$$

where $\boldsymbol{S}$ is an invertible matrix and $\boldsymbol{P}$ is a permutation matrix. Note that $\boldsymbol{P}$ can be written

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_1 & 0 \\ 0 & \boldsymbol{P}_2 \end{pmatrix}$$

where $\boldsymbol{P}_1$ is $k \times k$ permutation matrix and $\boldsymbol{P}_2$ is $n - k \times n - k$ permutation matrix.

Without loss of generality, let $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ be in the standard form thus

$$\boldsymbol{G}_{\mathscr{A}} = \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{A} \end{pmatrix} \text{ and } \boldsymbol{G}_{\mathscr{B}} = \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{B} \end{pmatrix}.$$

Thus

$$\boldsymbol{G}_{\mathscr{B}} = \begin{pmatrix} \boldsymbol{I}_k & \boldsymbol{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{S}\boldsymbol{P}_1 & \boldsymbol{S}\boldsymbol{A}\boldsymbol{P}_2 \end{pmatrix},$$

hence

$$\boldsymbol{I}_k = \boldsymbol{S}\boldsymbol{P}_1 \text{ and } \boldsymbol{S} = \boldsymbol{P}_1^{-1}.$$

Considering this we get

$$\boldsymbol{B} = \boldsymbol{P}_1^{-1}\boldsymbol{A}\boldsymbol{P}_2.$$

Referring to Proposition 1.5.4 and if we consider $\boldsymbol{A}$ and $\boldsymbol{B}$ as incidence matrices of labeled bipartite graphs then these two graphs are isomorphic. The labels here represent the field elements. ■

Note that in the basic equivalence we assume that we know the set of coordinates where the information set of code $\mathscr{A}$ is permuted. This is not true for the general problem and in practice for cryptosystems using code equivalence problem. Thus intuitively we need to find the information set of k coordinates among n, thus $\binom{n}{k}$.

Combining this reduction with the method introduced in [4] to solve graph isomorphism problem by finding canonical form with time complexity $\exp(\widetilde{O}(\sqrt{n}))$ we obtain a bound of exponential time complexity in n to solve permutation code equivalence problem.

1.6 Other Related Work

The problem of finding automorphism group of linear codes is closely related to solving equivalence problem between linear codes. Thus the algorithms that are introduced to find automorphism group of linear codes can be also used to solve the equivalence.

Leon in [53] used the idea of refinement of ordered partitions to compute the automorphism group of linear codes. The algorithm works by finding the set of low weight codewords, since it is of small size with respect to the code and it is invariant by the automorphism. This set is used to find the automorphism instead of looking at the whole code and it should contain sufficient structure to describe the automorphism. Often it is sufficient to use the minimum weight codewords, if this set is very small and does not contain sufficient structure to describe the automorphism we extend the set by adding the codewords of low weight next to the minimum. The algorithm partitions this set into small subsets each invariant by the automorphism then it utilizes the concepts of base and strong generating set to find the automorphism group. The algorithm of Leon is implemented in some computer algebra systems to find automorphism groups of linear codes and to solve the equivalence. In GAP it is used to solve permutation equivalence over binary field while in Magma it is used to solve permutational and diagonal equivalence over small prime fields and $\mathbb{F}_4$ for codes of small length [78]. The complexity of the algorithm is exponential in the dimension of the code since it finds all the minimum weight codewords.

In [37] Feulner introduced an algorithm to find automorphism group of linear codes that uses the idea of partition and refinement. His algorithm finds also a canonical representative (canonical generator matrix) of linear code such that any two equivalent codes have the same representative. There is nothing concrete about the complexity of the algorithm beyond the practical experiments which show that it is comparable to Leon's algorithm and thus exponential in the dimension of the code.

Chapter 2

Groebner Basis

Groebner basis is an important algebraic tool to solve systems of non-linear algebraic equations in multivariables. It was introduced by B. Buchberger in 1965 in his PhD thesis [18]. Groebner basis has various applications in cryptography, coding theory, algebra and many other fields. In this section we provide a quick review of Groebner basis theory and the famous algorithms that compute Groebner basis and its computation complexity. We start by some basic definitions and notations, we will follow [25, 30] in their notation.

2.1 Basic Definitions and Notation

Definition 2.1.1. Let $\mathbb{F}$ be a field. A *monomial* in $x_1, \ldots, x_n$ is a product of the form $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ where all the exponents are non-negative integers. We write $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$ with $\alpha = (\alpha_1, \ldots, \alpha_n)$. A *term* is of the form $a_\alpha x^\alpha$ with $a_\alpha \in \mathbb{F}^*$. A *polynomial* f over $\mathbb{F}$ is a sum of terms $f = \sum_\alpha a_\alpha x^\alpha$ where $a_\alpha \in \mathbb{F}^*$. The *total degree of a monomial* x^α is $|\alpha| = \alpha_1 + \ldots + \alpha_n$. The *total degree of a polynomial* $f = \sum_\alpha a_\alpha x^\alpha$ is $deg(f) = \max_{\alpha : a_\alpha \neq 0}(|\alpha|)$. The set of all polynomials over $\mathbb{F}$ form a commutative algebra under polynomial addition and multiplication denoted by $\mathbb{F}[x_1, \ldots, x_n]$ and called *polynomial ring* in n variables over $\mathbb{F}$.

Definition 2.1.2. A subset $I \subset \mathbb{F}[x_1, \ldots, x_n]$ is an *ideal* if it satisfies:

1. $0 \in I$ ($I \neq \emptyset$).
2. If $f, g \in I$, then $f + g \in I$.
3. If $f \in I$ and $h \in \mathbb{F}[x_1, \ldots, x_n]$ then $hf \in I$.

The ideal generated by the polynomials $f_1, \ldots, f_s$ is

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

Definition 2.1.3. A polynomial is said to be *homogeneous* if all its monomials have the same degree. An ideal generated by homogeneous polynomials is called *homogeneous ideal*. The *homogenization* of a non-homogeneous polynomial $f(x_1, \ldots, x_n)$ is $f^h = h^{deg(f)} f(\frac{x_1}{h}, \ldots, \frac{x_n}{h})$ where h is a new variable.

Remark 2.1.4. Homogeneous ideals contain non-homogeneous polynomials.

Definition 2.1.5. Let $I \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal, the *radical* of I, denoted $\sqrt{I}$ is

$$\sqrt{I} = \{f : f^m \in I \; for \; some \; integer \; m \geqslant 1\}$$

The ideal I is called *radical ideal* if $\sqrt{I} = I$.

Note that it is always true that $I \subseteq \sqrt{I}$.

Lemma 2.1.6. *[25] Let $I \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal then $\sqrt{I}$ is an ideal in $\mathbb{F}[x_1, \ldots, x_n]$ with $I \subseteq \sqrt{I}$ and $\sqrt{I}$ is a radical ideal.*

Theorem 2.1.7 (Hilbert Basis Theorem)**.** *[25] Every ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$ has a finite generating set. That is, $I = \langle g_1, \ldots, g_t \rangle$ for some $g_1, \ldots, g_t \in I$.*

Theorem 2.1.8 (The Ascending Chain Condition)**.** *[25] Let $I_1 \subset I_2 \subset I_3 \subset \ldots$ be an ascending chain of ideals in $\mathbb{F}[x_1, \ldots, x_n]$ then there exists an $N \in \mathbb{N}$ such that $I_N = I_{N+1} = I_{N+2} = \ldots$.*

Definition 2.1.9. Let $\mathbb{F}$ be a field, $\overline{\mathbb{F}}$ its algebraic closure and $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$ are polynomials. The *affine variety* defined by $f_1, \ldots, f_s$ in $\overline{\mathbb{F}}$ is

$$V_{\overline{\mathbb{F}}}(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in \overline{\mathbb{F}}^n : f_i(a_1, \ldots, a_n) = 0, \; \forall 1 \leqslant i \leqslant s\}$$

Let $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal we define

$$V_{\overline{\mathbb{F}}}(I) = \{(a_1, \ldots, a_n) \in \overline{\mathbb{F}}^n : f(a_1, \ldots, a_n) = 0, \forall f \in I\}$$

Let $V \subset \mathbb{F}^n$ be an affine variety then we define $I(V) \subset \mathbb{F}[x_1, \ldots, x_n]$ by

$$I(V) = \{f \in \mathbb{F}[x_1, \ldots, x_n] : f(a_1, \ldots, a_n) = 0, \forall (a_1, \ldots, a_n) \in V\}$$

The ideal I is said to be *zero-dimensional* if the polynomial system $f_i = 0, 1 \leqslant i \leqslant s$ has a finite number of solutions in an algebraically closed extension $\overline{\mathbb{F}}$ of $\mathbb{F}$ and *positive-dimensional* if the polynomial system has infinitely many solutions. This naming corresponds to the dimension of $V_{\overline{\mathbb{F}}}(I)$ as a variety.

Note that in a finite field of q elements, $\mathbb{F}_q$, the equations $x_i^q = x_i, 1 \leqslant i \leqslant n$ always hold, these equations are called *field equations*. In this case $V_{\mathbb{F}_q}(f_1, \ldots, f_s) = V_{\overline{\mathbb{F}}_q}(f_1, \ldots, f_s, x_1^q - x_1, \ldots, x_n^q - x_n)$.

Theorem 2.1.10. *[25] Let $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal then $V_{\overline{\mathbb{F}}}(I) = V_{\overline{\mathbb{F}}}(f_1, \ldots, f_s)$ and $V_{\mathbb{F}}(I) = V_{\mathbb{F}}(f_1, \ldots, f_s)$.*

Theorem 2.1.11. *[25] Let $\mathbb{F}$ be an algebraically closed field and $J \subset \mathbb{F}[x_1, \ldots, x_n]$ is an ideal then*

$$I(V_{\mathbb{F}}(J)) = \sqrt{J}$$

Theorem 2.1.12. *[8] Let $I \subset \mathbb{F}[x_1, \ldots, x_n]$ be an ideal such that, for all $1 \leqslant i \leqslant n$, there exists $f_i \in I \cap \mathbb{F}[x_i] \neq 0$ with $gcd(f_i, f_i') = 1$, where f_i' is the derivative of f_i. Then I is radical.*

Corollary 2.1.13. *Let $I \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal then the ideal $\langle I, x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ is radical.*

2.2 Solving Polynomial Systems

The problem of solving polynomial systems over a given field is an old problem in algebraic geometry and symbolic computation. It also appears in many different disciplines such as computer algebra, physics, geometry and cryptography. Optimizing the algorithms that solve this problem is a major research problem since this will lead to a breakthrough in many different fields of science. In cryptography, there are many cryptosystems relying on the hardness of solving polynomial systems such as multivariate public key cryptosystems (MPKC) [27].

The questions arising when we want to solve a polynomial system are, for instance, to decide if a given polynomial system is inconsistent (has no solution), zero-dimensional (has finitely many solutions), or positive-dimensional (has infinitely many solutions). If it is zero-dimensional then we would like to find the solution set (or at least a description of it) and if it is positive-dimensional we want to describe it.

The elimination theory is the oldest method to solve polynomial systems where it solves the systems by elimination of variable. The initial system of polynomials is reduced to an equivalent but easier system (e.g triangular system) then solutions can be found by solving univariate equations (which is not always possible) and successive evaluations of polynomials. This can be done by the use of linear algebra to represent polynomial systems (linearization) and the use of the theory of resultant. Informally speaking, the resultant of two polynomials is a polynomial expression of their coefficients, which is equal to zero if and only if the polynomials have a common root [2].

The resultant and Groebner basis are the main algebraic tools that are used to solve polynomial systems. There are also numerical methods to find approximate solutions for polynomial systems. Here we focus on algebraic methods and in this chapter we discuss Groebner basis and its algorithms.

2.3 Monomials Ordering

Definition 2.3.1. A *monomial ordering* on $\mathbb{F} >$ is any relation on the set of monomials $x^\alpha, \alpha \in \mathbb{N}^n$ satisfying:

1. The relation $>$ is a total ordering on the monomials of $\mathbb{F}[x_1, \ldots, x_n]$.
2. If $t_1 > t_2$ and s are monomials, then $st_1 > st_2$ for all t_1, t_2 and s.
3. for any monomial t we have $t > 1$.

Monomial ordering is called *graded* if monomials of different total degrees are ordered according to their total degree.

Below are some important monomial ordering:

1. *Lexicographic Order* (lex): We say $x^\alpha >_{lex} x^\beta$ if the leftmost nonzero entry in $(\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n)$ is positive.
2. *Graded Lex Order* $(grlex)$: We say $x^\alpha >_{grlex} x^\beta$ if the total degree of x^α is greater than the total degree of x^β. If the total degrees are equal we use the lex order.

3. *Graded Reverse Lex Order* ($grevlex$): We say $x^\alpha >_{grevlex} x^\beta$ if the total degree of x^α is greater than the total degree of x^β. If the total degrees are equal then if the rightmost nonzero entry in $(\alpha_1 - \beta_1, \ldots, \alpha_n - \beta_n)$ is negative.

4. *Elimination Order* (elm): We divide the variables in blocks that are ordered lexicographically. A monomial ordering is chosen inside the block, usually $grevlex$. That is, an order to eliminate the variables $x_1, \ldots, x_n$ is a monomial ordering on $\mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_m]$ such that for all monomials $x^{\alpha_1}y^{\beta_1}, x^{\alpha_2}y^{\beta_2}$ we have $x^{\alpha_1} > x^{\alpha_2} \Rightarrow x^{\alpha_1}y^{\beta_1} > x^{\alpha_2}y^{\beta_2}$. Elimination order is also called block order.

Example 2.3.2. Let $x > y > z$ and consider $M_1 = x^2y^3z^4$ and $M_2 = xy^5z^3$ then

$$M_1 >_{lex} M_2, \qquad M_1 >_{grlex} M_2, \qquad M_2 >_{grevlex} M_1.$$

Let $x_1 > x_2, y_1 > y_2, B_1 = \{x_1, x_2\} > B_2 = \{y_1, y_2\}$. An elimination order with $grevlex$ to eliminate B_1 orders the following monomials as follows

- $x_1^2y_1^4 > x_2^2y_2^6$.
- $x_1^2y_1^6 > x_1^2y_2^6$.

Definition 2.3.3. Let $f = \sum_\alpha a_\alpha x^\alpha$ be a nonzero polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ and $>$ is a monomial order.

1. The *leading monomial* of f, $LM(f) = \max_{\alpha: a_\alpha \neq 0} x^\alpha$ in monomials of f.
2. The *leading coefficient* of f is the coefficient associated to $LM(f)$.
3. The *leading term* of f is $LT(f) = LC(f)LM(f)$.
4. For an ideal I, we denote by $LT(I)$ the set of all leading terms of elements of I and $\langle LT(I)\rangle$ is the ideal generated by the elements of $LT(I)$.
5. The *terms* of f is the set $T(f) = \{a_\alpha x^\alpha : a_\alpha \neq 0\}$.
6. The *monomials* of f is the set $M(f) = \{x^\alpha : a_\alpha \neq 0\}$.

2.4 Polynomials Division

In this section we generalize the division of polynomials of one variable to division of polynomials of $\mathbb{F}[x_1, \ldots, x_n]$.

Theorem 2.4.1. *[25] Fix a monomial order on $\mathbb{F}[x_1, \ldots, x_n]$ and let $F = (f_1, \ldots, f_s)$ be an ordered s-tuple of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ then every $f \in \mathbb{F}[x_1, \ldots, x_n]$ can be written as*

$$f = a_1f_1 + \ldots + a_sf_s + r$$

where $a_i, r \in \mathbb{F}[x_1, \ldots, x_n]$ and either $r = 0$ or r is a linear combination, with coefficients in $\mathbb{F}$, of monomials, none of which is divisible by any of $LT(f_1), \ldots, LT(f_s)$. We will call r a remainder of f on division by F. Furthermore, $deg(f) \geqslant deg(a_if_i)$.

The division algorithm processes the following steps:

Input: $f_1, \ldots, f_s, f$
Output: $a_1, \ldots, a_s, r$
$a_1 := 0, \ldots, a_s := 0, r := 0$
$p := f$
while $p \neq 0$ **do**
  $i := 1$
  $divisionoccurred := false$
  while $i \leqslant s$ **and** $divisionoccurred = false$ **do**
    if $LT(f_i)$ divides $LT(p)$ **then**
      $a_i := a_i + LT(p)/LT(f_i)$
      $p := p - (LT(p)/LT(f_i))f_i$
      $divisionoccurred := true$
    else
      $i := i + 1$
    end if
  end while
  if $divisionoccurred = false$ **then**
    $r := r + LT(p)$
    $p := p - LT(p)$
  end if
end while

Unfortunately, the remainder r of the division is not unique if we change the order of the division. This will not help to determine if a polynomial lies in certain ideal since it can give a non-zero remainder by some order of the division while another order gives zero. Thus we need to look for a generating set for the ideal that gives a unique remainder regardless of the order of the division. Fortunately Groebner basis will have this property.

Definition 2.4.2. Let $f, g, p \in \mathbb{F}[x_1, \ldots, x_n]$ with $p \neq 0$ and P is a finite set of $\mathbb{F}[x_1, \ldots, x_n]$. Then we say

- The polynomial f *reduces to* g *modulo* p and we write $f \underset{p}{\rightarrow} g$ if there exist a monomial s and $m = x^\alpha \in M(f)$ such that $sLM(p) = m$ and $g = f - \frac{a_\alpha}{LC(p)} sp$ where a_α is the coefficient of m in f.
- The polynomial f *reduces to* g *modulo* P and we write $f \underset{P}{\rightarrow} g$ if $f \underset{p}{\rightarrow} g$ for some $p \in P$.
- The polynomial f is *reducible modulo* p if there exists $g \neq f \in \mathbb{F}[x_1, \ldots, x_n]$ such that $f \underset{p}{\rightarrow} g$.
- The polynomial f is *reducible modulo* P if there exists $g \neq f \in \mathbb{F}[x_1, \ldots, x_n]$ such that $f \underset{P}{\rightarrow} g$.
- The polynomial f is *top reducible modulo* P if there exists $g \in \mathbb{F}[x_1, \ldots, x_n]$ such that $f \underset{P}{\rightarrow} g$ and $LM(g) < LM(f)$.
- A *reduction of* f *modulo* $P = \{p_1, \ldots, p_s\}$ is a polynomial g such that there exist polynomials $g_1, \ldots, g_s \in \mathbb{F}[x_1, \ldots, x_n]$ with $f = \sum_{i=1}^{s} g_i p_i + g$ and no leading monomial of p_i divides a monomial of g. We write $g = \overline{f}^P$.

Definition 2.4.3. Let $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ different from zero with $LM(f) = x^\alpha$ with $\alpha = (\alpha_1, \ldots, \alpha_n)$

and $LM(g) = x^\beta$ with $\beta = (\beta_1, \ldots, \beta_n)$ and consider $\gamma = (\gamma_1, \ldots, \gamma_n)$ with $\gamma_i = \max(\alpha_i, \beta_i)$ for each i. We define the *least common multiple* of $LM(f)$ and $LM(g)$ and we write $LCM(LM(f), LM(g))$ to be the monomial x^γ.

We define the S-polynomial of f and g to be

$$S(f,g) = \frac{LCM(LM(f), LM(g))}{LT(f)} f - \frac{LCM(LM(f), LM(g))}{LT(g)} g$$

The S-polynomials play an important role in Groebner basis computation because of the Buchberger criterion of Groebner basis with S-polynomials stated in Theorem 2.5.8.

2.5 Groebner Basis

Definition 2.5.1. (Groebner Basis) Fix a monomial order. A finite subset $G = \{g_1, \ldots, g_m\}$ of an ideal I is said to be a Groebner basis if $\langle LT(G)\rangle = \langle LT(I)\rangle$. A Groebner basis G is called reduced if $LC(g) = 1$ for all $g \in G$ and g is not reducible modulo $G \setminus \{g\}$ for all $g \in G$.

Theorem 2.5.2. *[25] Any polynomial ideal different from zero has a Groebner basis and has a unique reduced Groebner basis.*

For a fixed monomial ordering, Groebner basis is not necessarily unique, that is if $\{g_1, g_2\}$ is a Groebner basis for some ideal also $\{g_1, g_2, g_1 + g_2\}$ is a Groebner basis. The reduced Groebner basis is unique.

Definition 2.5.3. Let $I = \langle f_1, \ldots, f_s\rangle$ be a homogeneous ideal. The set $G \subset \mathbb{F}[x_1, \ldots, x_n]$ is a *degree d Groebner basis* of the ideal I if G generates I and one of the following holds:

- $\overline{S(g_1, g_2)}^G = 0$ for all $g_1, g_2 \in G$ where $deg(S(g_1, g_2)) \leqslant d$.
- All $f \in I$, $deg(f) = d'$ with $d' \leqslant d$ is top reducible by G.

One can notice that, as a result from Hilbert Basis Theorem and ascending chain condition, for a large enough degree D, degree D Groebner basis is a Groebner basis for the homogeneous ideal and thus $G_1 \subset \ldots \subset G_{D-1} \subset G_D = G_{D+1} = \ldots$ where G_i is the reduced i-basis.

Proposition 2.5.4. *Let $G = \{g_1, \ldots, g_t\}$ be a Groebner basis for an ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$ and let $f \in \mathbb{F}[x_1, \ldots, x_n]$. Then there is a unique $r \in \mathbb{F}[x_1, \ldots, x_n]$ with the following two properties:*

- *No term of r is divisible by any of $LT(g_1), \ldots, LT(g_t)$.*
- *There is $g \in I$ such that $f = g + r$.*

In particular, r is the remainder on division of f by G no matter how the elements of G are listed when using the division algorithm.

As a direct result from the above proposition, a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ belongs to the ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$ which has a Groebner basis G if and only if $\overline{f}^G = 0$, i.e the remainder on division of f by G is zero. Note that, the reduced Groebner basis of a polynomial system is $\{1\}$ if and only if the system has no solution.

Proposition 2.5.5. *Fix an admissible term ordering and let G be a basis of the ideal $I \subset \mathbb{F}[x_1, \ldots, x_n]$, then the following statements are equivalent:*

1. *The set G is a Groebner basis.*
2. *For all $f \in I, \overline{f}^G = 0$.*
3. *Every $0 \neq f \in I$ is reducible modulo G.*
4. *Every $0 \neq f \in I$ is top reducible modulo G.*
5. *For all $f \in I$, there exists $g \in G$ such that $LT(g)|LT(f)$.*

In this case, the reduction of any f modulo G is unique, and is called the Normal form of f modulo G.

Theorem 2.5.6 (Elimination Theorem)**.** *Let $I \subset \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_m]$ be an ideal and G a Groebner basis of I for an elimination order with $x_1, \ldots, x_n > y_1, \ldots, y_m$ then*

$$G_n = G \cap \mathbb{F}[y_1, \ldots, y_m]$$

is a Groebner basis for the elimination ideal $I_n = I \cap \mathbb{F}[y_1, \ldots, y_m]$.

Theorem 2.5.7 (Homogenization Theorem)**.** *Let $F = \{f_1, \ldots, f_s\} \subset \mathbb{F}[x_1, \ldots, x_n]$ and F^h is the homogenized version of F. If G^h is a Groebner basis for $\langle F^h \rangle$, a Groebner basis of $\langle F \rangle$ can be obtained by dehomogenizing G^h (by setting the homogenization variable to 1). This Groebner basis is not necessarily reduced.*

Groebner basis is very powerful algebraic tool and it is used to solve the following problems:

- *The ideal membership problem* of determining if a given polynomial belongs to a given ideal.
- *Solving polynomial equations problem* where we need to find the solutions of a given polynomial system.
- *The implicitization problem* of finding system of polynomial equations that defines a given variety.

We conclude this section by Buchberger criterion which will be used in Buchberger algorithm to find Groebner basis.

Theorem 2.5.8 (Buchberger Criterion)**.** *Let I be a polynomial ideal. Then a basis $G = \{g_1, ..., g_t\}$ for I is a Groebner basis for I if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G (listed in some order) is zero.*

2.6 Buchberger Algorithm

Buchberger algorithm was first introduced in 1965 together with Groebner basis in Bruno Buchberger PhD thesis under the supervision of Wolfgang Groebner [18]. The algorithm starts the computation by a set of polynomials G and extends it to a Groebner basis of the ideal I. It uses the fact that if $f \in I$ and G is a Groebner basis then the reduction of f by G gives zero otherwise G is not a Groebner basis and f will be added to G. We continue adding all such f using Buchberger criterion until Groebner basis definition is satisfied. A basic version of Buchberger algorithm is presented next.

Theorem 2.6.1 (Buchberger Algorithm)**.** *Let $I = \langle f_1, \ldots, f_s \rangle \neq 0$ be a polynomial ideal. Then a Groebner basis for I can be constructed in a finite number of steps by the following algorithm:*

***Input**: $F = \{f_1, \ldots, f_s\}$*
***Output**: a Groebner basis $G = \{g_1, \ldots, g_t\}$ for I , with $F \subset G$*
$G := F$
repeat
  $G' := G$
  for** each pair $\{p, q\}, p \neq q$ in G' **do
    $S := \overline{S(p,q)}^{G'}$
    if** $S \neq 0$ **then
      $G := G \cup \{S\}$
    end if
  end for
***until** $G = G'$*

The Buchberger criterion decides that G is a Groebner basis for an ideal I if and only if for all pairs g_i and g_j in G the remainder of the division of $S(g_i, g_j)$ by G (listed in some order) is zero. Thus Buchberger algorithm starts from a set of polynomials and then computes the S-polynomials for all pairs in G. If the computed S-polynomial is not reduced to zero by G it will be added to G until all the S-polynomials are reduced to zero by the elements of G.

It is clear that the basic version of Buchberger algorithm is not optimized in terms of computation. The most intensive operation in the algorithm is the reduction of the S-polynomials by the basis elements. The algorithm considers all pairs of S-polynomials regardless of that there are some of them are known to be zero in a previous step. To improve the performance of the algorithm the minimal number of S-polynomials need to be considered. One way to do that, there are some S-polynomials that are known to be reduced to zero beforehand and some of them are combination of other S-polynomials. An improved version of Buchberger algorithm which cares of the these improvements is presented next.

Input: $F = \{f_1, \ldots, f_s\}$
Output: G, a Groebner basis for $I = \langle f_1, \ldots, f_s \rangle$
$B := \{(i, j) : 1 \leqslant i < j \leqslant s\}$
$G := F$
$t := s$
while $B \neq \emptyset$ **do**
  Select $(i, j) \in B$
  if $LCM(LT(f_i), LT(f_j)) \neq LT(f_i)LT(f_j)$ **and** $Criterion(f_i, f_j, B)$ is $false$ **then**
    $S := \overline{S(f_i, f_j)}^{G}$
    if $S \neq 0$ **then**
      $t := t + 1$
      $f_t := S$
      $G := G \cup \{f_t\}$
      $B := B \cup \{(i, t) : 1 \leqslant i \leqslant t - 1\}$
    end if
  end if
  $B := B \setminus \{(i, j)\}$

18: **end while**

Note that, when $LCM(LT(f_i), LT(f_j)) = LT(f_i)LT(f_j)$ we say the leading terms $LT(f_i)$ and $LT(f_j)$ are *relatively prime* and it is easy to see that $S(f_i, f_j) \underset{G}{\rightarrow} 0$. The $Criterion(f_i, f_j, B)$ is true if there is some $k \notin \{i, j\}$ for which the pairs (i, k) and (k, j) are not in B and $LT(f_k)$ divides $LCM(LT(f_i), LT(f_j))$. It is not difficult to see that $S(f_i, f_j)$ is a combination of $S(f_i, f_k)$ and $S(f_k, f_j)$, thus it suffices to add $S(f_i, f_k)$ and $S(f_k, f_j)$.

2.6.1 Other Improvements of Buchberger algorithm

In this section we discuss some other improvements that may be very useful to avoid a lot of useless computation when implementing Buchberger algorithm.

- As we have noticed, the order of the division by the polynomials $f_1, \ldots, f_s$ affects the computation of Groebner basis. Ordering the leading terms of the polynomials in an increasing degree for a selected monomial ordering has been shown to perform better practically. That is because for a graded ordering, leading terms with smaller degree will be selected first and thus less comparison operations will be needed.
- Another way to improve Buchberger algorithm is to use what is so called *normal selection strategy* which selects pairs from B such that their LCM has the smallest degree. The resulting S-polynomial from this pair will have a small degree and thus can be used to avoid a lot of other S-polynomials that are reduced to zero.
- Selecting the suitable monomial ordering according to the problem. In practice *grevlex* ordering performs better most of the time. There are some algorithms to move from an order to another in polynomial time, when the number of solutions is finite (without need to recompute Groebner basis), such as FGLM [33] and Groebner walk [21].
- Avoiding reduction to zero provides big enhancement to the performance since a lot of S-polynomials are reduced to zero though the improved algorithm avoids many of them but still big number of reduction to zero still takes place. This has been noted by Faugère F_5 algorithm [31] and a strong criterion to avoid zero reduction was introduced.
- Another way to improve the algorithm is to systematically inter-reduce the set of polynomials of G each time it is enlarged. This will lessen the number of S-polynomials to be considered in the next steps.

2.7 Groebner Basis and Linear Algebra

In this section we discuss the linear algebra techniques that are used to compute Groebner basis. These techniques has been used in F_4 and F_5 algorithms and have shown considerable improvement in Groebner basis computation. Before we proceed to the F_4 and F_5 algorithms we need to introduce matrix representation of polynomial system and Macaulay matrix.

2.7.1 Matrix Representation of Polynomials

This technique is an old method for solving non-linear systems of multivariables where we use linear algebra techniques to represent these systems. We represent the system by a matrix where each row corresponds to one of the polynomials and each column corresponds to a monomial that appears in one of the polynomials. Then we can use linear algebra and Gauss elimination to simplify and solve the system.

Consider $G = \{g_1, \ldots, g_m\}$ a set of polynomials in a polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ and let $M = \{m_1, \ldots, m_r\}$ be the decreasing ordered set of all monomials appearing in the polynomials in G. For $g_i, 1 \leqslant i \leqslant m$ we can write $g_i = \sum_{j=1}^{r} a_{ij} m_j$. We represent this system by an $m \times r$ matrix $\mathcal{M}$ where the element $\mathcal{M}_{i,j}$ represents the coefficient of the monomial m_j appearing in the polynomial g_i. Thus the matrix representation of this system of polynomials will be as follows:

$$\mathcal{M} = \begin{array}{c} \\ g_1 \\ g_2 \\ \vdots \\ g_m \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & \ldots & m_r \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1r} \\ a_{21} & a_{22} & \ldots & a_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mr} \end{pmatrix} \end{array}$$

After expressing the system of polynomials G by a matrix $\mathcal{M}$ we can perform row operations to compute an Echelon form. The computation of an echelon form will result in an equivalent matrix $\mathcal{M}'$. By reverting back $\mathcal{M}'$ to a system of polynomials will produce a system G' which is equivalent to G. In other words, if $I = \langle G \rangle$ and $I' = \langle G' \rangle$ then $I = I'$.

2.7.2 Macaulay Matrix and Groebner Basis

Definition 2.7.1 (Macaulay Matrix)**.** Let $f_1, \ldots, f_s$ be polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ each of degree $d_i, 1 \leqslant i \leqslant s$. The Macaulay matrix of degree d, $\mathcal{M}_d$, is the matrix that contains for each polynomial f_i and each monomial m of degree $d - d_i$ a row that corresponds to mf_i. The columns of $\mathcal{M}_d$ are indexed by the monomials of degree $\leqslant d$. In the row that corresponds to mf_i, the element in the column indexed by m_j corresponds to the coefficient of the monomial m_j in the polynomial mf_i.

$$\mathcal{M}_d = \begin{array}{c} \\ \vdots \\ mf_i \\ \vdots \end{array} \begin{array}{c} \textit{Monomials } m_j \textit{ of degree} \leqslant d \\ \begin{pmatrix} \\ \textit{coefficients of } m_j \textit{ in } mf_i \\ \\ \end{pmatrix} \end{array}$$

We can define Macaulay matrix to be the matrix representation of the polynomial system

$$\{mf_i : 1 \leqslant i \leqslant s, m \text{ is a monomial with } deg(m) = d - d_i\}$$

We call the representation of polynomial system by Macaulay matrix, linearization of the system.

For the homogeneous case, if $I = \langle f_1, \ldots, f_s \rangle$ is a homogeneous ideal in $\mathbb{F}[x_1, \ldots, x_n]$ with $deg(f_i) = d_i, 1 \leqslant i \leqslant s$. We define $I_d = \{f \in I : deg(f) = d\}$, I_d has a structure of vector space. Macaulay matrix of degree d that describes I_d, $\mathcal{M}_d^h$, will have for each f_i and each monomial m of degree $d - d_i$ one row whose entry in the column indexed by a monomial m_j is the coefficient of m_j in mf_i. Note that, the number of monomials of degree d in n variables is $\binom{n+d-1}{d}$, this represents the number of columns of $\mathcal{M}_d^h$.

Performing Gaussian elimination in Macaulay matrix $\mathcal{M}_d^h$ gives a basis for I_d. It has been proved in [51] that if D is the maximal degree that appears in the reduced Groebner basis of the ideal I, Groebner basis of I can be constructed by considering the reduced Macaulay matrices $\widetilde{\mathcal{M}_d^h}$ with $\min(d_1, \ldots, d_s) \leqslant d \leqslant D$.

Theorem 2.7.2. *[51] Let $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ be a homogeneous ideal. The set of polynomials corresponding to the rows of the reduced Macaulay matrices of degree $d \leqslant D$ of the polynomials $f_1, \ldots, f_s$ forms a degree D Groebner basis of I. There exists an integer D such that a degree D Groebner basis of I is a Groebner Basis of I.*

Groebner basis obtained from Macaulay matrices is not reduced.

Considering the affine case, of course any polynomial can be homogenized by adding new variable, thus the previous approach is valid for affine case. The drawback of this method (homogenization) is that wrong solutions can be found that correspond to $h = 0$ where h is the homogenization variable. These solutions are called *solutions at infinity* and they are not solutions to the initial system.

Computation of Groebner basis for affine polynomials using Macaulay matrices (without homogenization) is performed in a similar way where we consider the general construction of Macaulay matrices. In $\mathcal{M}_d$ the columns represent all monomials of degree $\leqslant d$ and each row is a tf_i with t runs over all monomials that make the degree of the product $\leqslant d$. Degree D Groebner basis is constructed as before from the reduced Macaulay matrices $\widetilde{\mathcal{M}_d}, d \leqslant D$. For large enough D Groebner basis for the ideal is computed.

2.7.3 The F_4 Algorithm

The F_4 algorithm was introduced in 1999 by Faugère in [30]. The algorithm uses the techniques of linear algebra and S-polynomials to compute Groebner basis. The biggest advantage of the algorithm is that it reduces large number of S-polynomials simultaneously in one step using linear algebra. This makes the algorithm much more efficient than Buchberger algorithm where only one S-polynomial is selected and reduced at a time. In this section we describe the F_4 algorithm. We follow Faugère original description and notation.

Let $F = \{f_1, \ldots, f_s\}$ be a set of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$, $>$ be a monomial ordering, and $M(F) = (m_1, \ldots, m_l)^T$ is the column vector of monomials appearing in F ordered decreasingly according to $>$. Let $\mathcal{M}_F$ be the matrix representation of F with columns ordered according to $M(F)$. Then $\mathcal{M}_{F_{i,j}}$ is the coefficient of m_j in f_i and $\overline{F} = \mathcal{M}_F.M(F)$ where $\overline{F}$ is the column vector $(f_1, \ldots, f_s)^T$. We can move back and forth between polynomial representation and matrix representation.

Let $\mathcal{M}_F$ be the matrix representation of the set of polynomials F then we denote by $\widetilde{F}$ the polynomial representation of the reduced matrix $\widetilde{\mathcal{M}_F}$ of $\mathcal{M}_F$ (the reduction is done without pivoting the columns).

Definition 2.7.3. Let f_i and f_j be two polynomials from $\mathbb{F}[x_1, \ldots, x_n]$, we define the critical pair of f_i and f_j denoted by $Pair(f_i, f_j)$ to be the 5-tuple $(LCM_{ij}, m_i, f_i, m_j, f_j)$ where m_i, m_j are monomials and LCM_{ij} is the least common multiple of f_i and f_j such that $LCM_{ij} = LM(m_i f_i) = LM(m_j f_j)$. The degree of $P_{ij} = Pair(f_i, f_j)$ is defined to be $deg(LCM_{ij})$ and $Left(P_{ij}) = m_i f_i$ and $Right(P_{ij}) = m_j f_j$.

The F_4 Algorithm

Input: A set of polynomials $F = (f_1, \ldots, f_s)$,
Sel is a function that takes the list of pairs and returns a list of pairs of minimum degree with $Sel(I) \neq \emptyset$ if $I \neq \emptyset$
Output: G, a Groebner basis for $I = \langle f_1, \ldots, f_s \rangle$
$G := F, d := 0, P := \{Pair(f, g) : f, g \in G, f \neq g\}$
while $P \neq \emptyset$ **do**
  $d := d + 1$
  $P_d := Sel(P)$
  $P := P \setminus P_d$
  $L_d := Left(P_d) \cup Right(P_d)$
  $\tilde{F}_d^+ := Reduction(L_d, G)$
  for $h \in \tilde{F}_d^+$ **do**
    $P := P \cup \{Pair(h, g) : g \in G\}$
    $G := G \cup \{h\}$
  end for
end while
return G

The Reduction Process

Input: L, G finite subsets of $\mathbb{F}[x_1, \ldots, x_n]$
Output: A finite subset of $\mathbb{F}[x_1, \ldots, x_n]$ or $\emptyset$
$F := SymbolicPreProcessing(L, G)$
$\tilde{F} :=$ Gaussian reduction of F with respect to $>$
$\tilde{F}^+ := \{f \in \tilde{F} : LT(f) \notin LT(F)$ and $f \neq 0\}$
return $\tilde{F}^+$

The Symbolic Preprocessing

Input: L, G finite subsets of $\mathbb{F}[x_1, \ldots, x_n]$
Output: A finite subset of $\mathbb{F}[x_1, \ldots, x_n]$
$F := L$
$Done := LM(F)$
while $Monomials(F) \neq Done$ **do**
  Choose an element $m \in Monomials(F) \setminus Done$
  $Done := Done \cup \{m\}$
  if m is top reducible modulo G **then**
    There exists $g \in G$ and a monomial m' such that $m := m' LM(g)$

10: $F := F \cup \{m'.g\}$
11: **end if**
12: **end while**
13: **return** F

The Selection Function

1: **Input**: A list of critical pairs P
2: **Output**: A list of critical pairs P_d
3: $d := \min\{deg(LCM(p)) : p \in P\}$
4: $P_d := \{p \in P : deg(LCM(p)) = d\}$
5: **return** P_d

When the F_4 algorithm selects the pairs of the minimum degree from the list of pairs, this is called the normal strategy for F_4. Then a set of $Left$ and $Right$ of the selected pairs is constructed and reduced by the elements of G (the set that will be extended to a Groebner basis). This corresponds to the selection of S-polynomials and their reduction by the basis elements in Buchberger algorithm. The difference is that we select and reduce many polynomials at a time and not only one. This is done using the matrix representation of the polynomials and Gaussian elimination which significantly enhances the computation. We update the list of pairs and the set G with the elements that are not reduced to zero in the reduction process until all the pairs are reduced to zero by G.

Theorem 2.7.4. *[30] The F_4 algorithm computes a Groebner basis G for the ideal $I = \langle F \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ such that $F \subseteq G$ and $\langle G \rangle = \langle F \rangle$.*

The symbolic preprocessing helps to improve the reduction process. If we want to reduce a set L by G then the symbolic preprocessing adds to a set F (initially $F = L$) all the multiples $m'.g$ with m' a monomial and $g \in G$ such that $LM(m'g)$ reduces an element from $Monomials(L)$. Thus the maximum number of reductors of L from G are considered. It also includes the polynomials whose leading terms are a monomial in one of the $LM(m'g)$, and so on by induction.

The F_4 algorithm can be improved by applying Buchberger criteria and improvements that were discussed in Section 2.6. In the basic version of F_4 presented above, the rows of the matrices generated in previous steps of reduction are ignored in the next steps. Another way to improve F_4 algorithm is to reuse all rows of matrices that are generated during the computation in the reduction process. This helps to speed up the reduction.

In spite of the considerable improvements introduced by the F_4 algorithm, still big amount of computation is wasted in reduction to zero. The algorithm can be improved by avoiding this unnecessary computation by knowing in advance the pairs that will be reduced to zero. This is introduced by F_5 algorithm that will be discussed next.

Improvements of linear algebra step results in a considerable improvement in the whole algorithm, since Gaussian elimination takes the biggest part of the algorithm. Using sparse matrices and compressed representation for them will improve the space complexity.

2.7.4 The F_5 Algorithm

In both Buchberger and F_4 algorithms about 90% of the computation time is spent in reduction to zero. The main idea of the F_5 algorithm is to optimize Buchberger criterion by avoiding as many as possible unnecessary computation that is reduced to zero. Assuming that the input system of polynomials is regular (see Definition 2.7.5) there is no reduction to zero. If the system is semi-regular (see Definition 2.7.7) there will be reduction to zero in the last step.

The F_5 algorithm avoids zero reduction by taking into account the relationships between polynomials $f_i f_j - f_j f_i = 0$ which are called the *trivial syzygies*. Groebner basis is computed incrementally, i.e. if G_i is a Groebner basis for $\langle f_1, \ldots, f_i \rangle$ the next step is to compute Groebner basis G_{i+1} for $\langle G_i, f_{i+1} \rangle$. In the matrix representation changing the order of the rows is not allowed and we eliminate rows that will be reduced to zero by previous rows, thus we avoid zero reduction. It has been proved by Faugère in [31] that if the system is regular, no zero reduction occurs. If the system of polynomials is not regular some zero reductions still may happen.

We present the matrix version of the F_5 algorithm (matrix-F_5) as presented by Bardet et al. in [7].

The matrix $\mathcal{M}$ appearing during Groebner basis computation of the ideal generated by $F = \{f_1, \ldots, f_s\}$ contains either elements of F or elements of F multiplied by some monomial or combination of a row with smaller rows (order of rows is defined next). It is convenient to locate each element in $\mathcal{M}$ by the triple (i, m, μ) where $1 \leqslant i \leqslant s$, m is a scaling monomial, and μ is a monomial representing the column index. The pair (i, m) is the index of a row in $\mathcal{M}$ that results from mf_i with combination with smaller rows, $S = (i, m)$ is used as a signature of the polynomial mf_i. We say $S = (i, m) < S' = (i', m')$ if $i < i'$ or $i = i'$ and $m < m'$. The allowed row operation is to replace the row by a linear combination of this row with rows that have smaller signatures scaled by some field elements. We denote by Rows($\mathcal{M}$) the polynomials corresponding to the rows of $\mathcal{M}$ and by $LM(\mathcal{M})$ the leading monomials of the polynomial representation of $\mathcal{M}$.

As we mentioned previously, F_5 computes Groebner basis incrementally thus we denote by $\mathcal{M}_{d,i}$ the Macaulay matrix that is used to compute degree d Groebner basis for $\langle f_1, \ldots, f_i \rangle$ with some rows are removed.

Definition 2.7.5. Let $(f_1, \ldots, f_s)$ be sequence of polynomials from $\mathbb{F}[x_1, \ldots, x_n]$. We say this sequence is *regular* if for all $1 \leqslant i \leqslant s$ and g such that $gf_i \in \langle f_1, \ldots, f_{i-1} \rangle$ then $g \in \langle f_1, \ldots, f_{i-1} \rangle$.

An affine sequence of polynomials $(f_1, \ldots, f_s)$ is regular if the sequence $(\tilde{f_1}^h, \ldots, \tilde{f_s}^h)$ is regular, where $\tilde{f_i}^h$ is the homogeneous part of f_i of highest degree.

Definition 2.7.6. Let $I = \langle f_1, \ldots, f_s \rangle$ in $\mathbb{F}[x_1, \ldots, x_n]$ be a homogeneous ideal of dimension 0 over an algebraically closed field, we define the *degree of regularity* of I by

$$deg_{reg}(I) = \min \left\{ d \geqslant 0 : dim_{\mathbb{F}} \left(\{ f \in I, deg(f) = d \} \right) = \binom{n+d-1}{d} \right\}$$

Note that, $\binom{n+d-1}{d}$ is the number of monomials of degree d.

Definition 2.7.7. Let $(f_1, \ldots, f_s)$ be sequence of homogeneous polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ and $I = \langle f_1, \ldots, f_s \rangle$ is of dimension 0. We say this sequence is *semi-regular* if for all $1 \leqslant i \leqslant s$ and g such that $gf_i \in \langle f_1, \ldots, f_{i-1} \rangle$ and $deg(gf_i) < deg_{reg}(I)$ then $g \in \langle f_1, \ldots, f_{i-1} \rangle$.

An affine sequence of polynomials $(f_1, \ldots, f_s)$ is semi-regular if the sequence $(f_1^h, \ldots, f_s^h)$ is semi-regular, where f_i^h is the homogeneous part of f_i of highest degree.

Proposition 2.7.8 (F_5 Criterion)**.** *[7] Assume we have homogeneous polynomials. Let $S = (j, m)$ be a signature of a row in $\widetilde{\mathcal{M}}_{d-d_i,i-1}$ with leading monomial t where $S < (i, 1)$ then the row with signature (i, t) belongs to the vector space generated by the rows of $\mathcal{M}_{d,i}$ having smaller index.*

Remark 2.7.9. For affine case, we let $S = (j, m)$ be the signature of a row in $\widetilde{\mathcal{M}}_{d',i-1}$ (for some d') where $S < (i, 1)$, with leading monomial t of degree $d - d_i$ (d' may be $> d - d_i$ in this case).

The F_5 criterion states that if a monomial t is reducible by $LM(G_{i-1})$, leading monomials of Groebner basis of $\langle f_1, \ldots, f_{i-1}\rangle$, then we do not need to consider the polynomial tf_i in the next step.

Matrix-F_5 Algorithm

The following matrix-F_5 algorithm computes degree D Groebner basis for homogeneous system of polynomials.

Input: Homogeneous polynomials $f_1, \ldots, f_s$ of degrees $d_1 \leqslant \ldots \leqslant d_s$; a maximal degree D
Output: The elements of degree at most D of the reduced Groebner basis of $f_1, \ldots, f_i$ with $1 \leqslant i \leqslant s$
Set $G_i := \emptyset, 1 \leqslant i \leqslant s$
for $d := d_1$ **to** D **do**
  $\mathcal{M}_{d,0} := \emptyset, \widetilde{\mathcal{M}}_{d,0} := \emptyset$
  for $i := 1$ **to** s **do**
    if $d < d_i$ **then**
      $\mathcal{M}_{d,i} := \mathcal{M}_{d,i-1}$
    else if $d = d_i$ **then**
      $\mathcal{M}_{d,i} :=$ add the new row f_i to $\widetilde{\mathcal{M}}_{d,i-1}$ with index $(i, 1)$
    else
      $\mathcal{M}_{d,i} := \widetilde{\mathcal{M}}_{d,i-1}$
      $Crit := LM(\widetilde{\mathcal{M}}_{d-d_i,i-1})$
      for f *in* Rows($\mathcal{M}_{d-1,i}$) $\setminus$ Rows($\mathcal{M}_{d-1,i-1}$) **do**
        $(i, u) := index(f)$ with $u = x_{j_1} \ldots x_{j_{d-d_i-1}}$ and $1 \leqslant j_1 \leqslant \ldots \leqslant j_{d-d_i-1} \leqslant n$
        for $j := j_{d-d_i-1}$ **to** n **do**
          if $ux_j \notin Crit$ **then**
            add the new row $x_j f$ with index (i, ux_j) in $\mathcal{M}_{d,i}$
          end if
        end for
      end for
    end if
    Compute $\widetilde{\mathcal{M}}_{d,i}$ by Gaussian elimination from $\mathcal{M}_{d,i}$
    Add to G_i all rows of $\widetilde{\mathcal{M}}_{d,i}$ not reducible by $LT(G_i)$
  end for
end for
return $[G_i | i = 1, \ldots, m]$

Proposition 2.7.10. *Any row entered by the algorithm matrix-F_5 into the matrix $\mathcal{M}_{d,i}$ represents a*

polynomial $g_i f_i + \ldots + g_1 f_1$, where g_i is reduced with respect to $\langle f_1, \ldots, f_{i-1} \rangle$. If $g \in G_i \setminus G_{i-1}$ then its index has the form (i, t).

Theorem 2.7.11. *The algorithm matrix-F_5 computes the elements of degree at most D of the reduced Groebner basis of $\langle f_1, \ldots, f_i \rangle$ with $1 \leqslant i \leqslant s$.*

Theorem 2.7.12. *Let $(f_1, \ldots, f_s)$ be a regular sequence then all the matrices generated by the matrix-F_5 algorithm have full rank.*

As we have mentioned above, one problem of F_5 algorithm if the sequence of polynomials is not regular still zero reduction may happen. That is because in regular sequences all syzygies are generated by the trivial syzygies. Thus in non-regular sequence different syzygies may exist. The F_5 algorithm enhances the computation by avoiding zero reduction by using signatures to track polynomials. On one hand this improves the reduction process on the other hand this adds some complexity since we need to keep track of signatures of the polynomials to be used later to avoid adding polynomials that will be reduced to zero. If the maximal degree D is large enough the reduced Groebner basis of the ideal is computed.

2.7.5 Complexity of Groebner Basis

The complexity of computation of Groebner basis is doubly exponential in the number of variables in its worst case 2.7.13. That is because the degrees of the polynomials in the resulting basis can be quite large. For example in [63], the authors gave the example of a system of n polynomials in n variables where polynomials of degree doubly exponential in n appear in the Groebner basis.

A large amount of work has been done to bound the degree of polynomials appearing during Groebner basis computation based on the degrees of the given generators of the ideal, see for example [67, 28]. In [67, 28] the following upper bound was introduced for Groebner basis computation

Theorem 2.7.13. *Fix a monomial order and let $I = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ and let $D = \max\{deg(f_i) : 1 \leqslant i \leqslant s\}$ then there exists a Groebner basis of I such that all its elements of degree $\leqslant D^{2^{O(n)}}$.*

Another bound on the maximal degree reached during the computation and on the complexity of Groebner basis computation is introduced by the following theorem

Theorem 2.7.14. *[51, 41] Let $f_1, \ldots, f_s$ be a system of s polynomials in n variables with $s \leqslant n$ and coefficients in any field $\mathbb{F}$. If the homogenized system has a finite number of solutions, then for the grevlex order, and for almost any linear change of variable, the maximum degree of any polynomial during the computation of the Groebner basis is bounded by Macaulay bound $\sum_{i=1}^{s}(d_i - 1) + 1$ with $d_i = deg(f_i)$ and $d_1 \geqslant \ldots \geqslant d_s$. The cost of the computation of Groebner basis is polynomial in D^n with $D = \max\{d_i\}$.*

In spite of existence of examples where the degree of polynomials grows double exponentially, in practice the computation of a Groebner basis seems to be much more efficient. For example if the ideal is zero dimensional, a Groebner basis may be computed in $O(D^n)$ where D is the maximum degree of the generators of the ideal [6]. The selection of monomial ordering plays an important role in the complexity of computation. The *grevlex* order has been shown to produce polynomials of smallest degree in all cases [9].

Introducing F_4 algorithm is considered a major milestone in Groebner basis computation. That is because it enhances a lot the computation by using the techniques of linear algebra accompanied with Buchberger criteria and the normal strategy for F_4 beside the symbolic preprocessing of polynomials before reduction and reuse of polynomials. The use of linear algebra enables doing reduction of large number of S-polynomials at a time. When introduced, the F_4 algorithm, it was able to solve some previously intractable problems such as *Cyclic9* [30].

In spite of the considerable improvements introduced by the F_4 algorithm still the worst case complexity remains the same as Buchberger algorithm but it is several times faster than previous algorithms for integers and modular computation [30].

As in F_4, the F_5 algorithm can use linear algebra to compute Groebner basis. The F_5 algorithm was introduced to optimize Buchberger criterion by avoiding any unnecessary computation of S-polynomials that are reduced to zero by means of signature of polynomials. When introduced, F_5 algorithm was very promising since it was able to solve *Cyclic10* that was previously intractable [31]. It was also used to break the HFE cryptosystem successfully [35].

The F_5 algorithm is about ten times faster than F_4 algorithm in most cases but this is not the case in general where one can find examples where F_4 performs better. The best performance of F_5 is achieved for non over-constrained systems. In the original version of F_5 algorithm when the system is over-constrained, bad performance is expected [31].

The work of Bardet in her PhD thesis [8] was considered the first to analyze the complexity of F_5 algorithm, the matrix version, for homogeneous regular and semi-regular sequences and to give a bound of the number of operations required to compute Groebner basis.

In Section 2.7.2 we described how to obtain degree D Groebner basis using linear algebra in Macaulay matrices. This gives a rise to estimate a simple bound on the number of operations required to compute Groebner basis for the F_5 algorithm which is given by the following proposition [7].

Proposition 2.7.15. *Fix a graded monomial ordering and let $(f_1, \ldots, f_s)$ be a system of homogeneous polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ and $I = \langle f_1, \ldots, f_s \rangle$. The number of operations in the filed $\mathbb{F}$ required to compute Groebner basis of I up to degree D is bounded by*

$$O\left(sD\binom{n+D-1}{D}^{\omega}\right), \quad \text{as } D \to \infty$$

where ω is the exponent of matrix multiplication over $\mathbb{F}$.

A good estimate of the maximum degree reached by computation of Groebner basis leads to a good estimate of the complexity. Considering *grevlex* and regular sequences, it has been shown in [51] that with a generic linear change of coordinates one can bound the degree of regularity d_{reg} of the ideal by Macaulay bound

$$d_{reg} \leqslant \sum_{i=1}^{s} (deg(f_i) - 1) + 1$$

Proposition 2.7.16. *[7] Let $(f_1, \ldots, f_n)$ be a regular system of homogeneous polynomials in $\mathbb{F}[x_1, \ldots, x_n]$, then the highest degree in the elements of a Groebner basis for a graded ordering is bounded by Macaulay bound.*

Our target in this section was to give a quick glance on the complexity of computation of Groebner basis rather than going to the whole detail. We refer the interested reader to [8, 6, 5, 7] to get the

detailed complexity analysis of Groebner basis computation with matrix-F_5 for regular and semi-regular sequences.

Chapter 3

Permutation Equivalence Problem (PEP)

Permutation equivalence is an important problem that plays a major role in code-based cryptography and complexity theory since we can reduce graph isomorphism problem to code equivalence problem in polynomial time. In this chapter we introduce many techniques, mostly algebraic, for solving PEP. We recall matrix definition of PEP and introduce algebraic modeling to solve the problem. We show that a permutation is a solution for the equivalence if and only if it is a solution for our algebraic system. The new modeling contains linear and polynomial equations in multivariables. We study the properties of the linear system and also the properties of the solution set.

Groebner basis is used successfully to solve algebraic system. We used the F_4 algorithm implemented in computation algebra system Magma. We use the maximum number of linear equations since it improves Groebner basis computation. There are many factors that affect the computation of Groebner basis such as: the number of variables, the rank of the linear system, the dimension of the hull, the size of the solution set, and the size of the field. We implement experiments in random linear codes with different parameters over different fields to estimate time and memory complexity.

When we solve PEP practically by Groebner basis, if the hull is trivial and the system has a unique solution, then the maximum degree reached by the F_4 algorithm is 2. Thus we can solve the system using linear algebra and Macaulay matrices of degree 2 polynomials obtained from the initial linear system.

Based on the experimental observations from the structure of the system and Groebner basis we found that we can reduce the complexity of solving in some cases. We introduce a new and efficient algebraic technique which we call block linearization. Using this technique we can improve the linear system by adding many equations without much computation. Thus in many cases we can solve PEP with the cost of linear algebra in our matrices.

Frobenius action is used to improve the linear part of the system. If the extension of the field is large enough and the linear codes under equivalence test are random with trivial hull one can solve the problem efficiently using only the linear part and Frobenius action. combining Frobenius action with block linearization enables to extend the range of solving to codes with non-trivial hull. We introduce two techniques to overcome the difficulty associated to the existence of the hull. The first is based on the supplementary of the hull and the second is based on the closest vector problem.

We also develop other techniques such as distinguishing correct positions of the permutation by punc-

turing the codes. Finally, we introduce an approach based on the ISD technique to solve the equivalence.

3.1 Notation

Throughout this chapter and the rest of the thesis we use the following notation. The symbol $\mathbb{1}_n$ represents the vector $(1,\dots,1)$ of length n. When the context is clear we will simply write $\mathbb{1}$. The concatenation of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is denoted by $(\boldsymbol{u},\boldsymbol{v})$. The set of matrices with entries in $\mathbb{F}$ having m rows and n columns is denoted by $\mathcal{M}_{m,n}(\mathbb{F})$. The set of $n\times n$ invertible matrices is the general linear group $\mathsf{GL}_n(\mathbb{F})$. The rows of a matrix are denoted by bold letters and the columns are denoted by capital bold letters. For instance the i-th row of a matrix $\boldsymbol{A}=(a_{i,j})$ is $\boldsymbol{a}_i$ and its j-th column is $\boldsymbol{A}_j$. It will be convenient to view matrices as column vector. For that purpose the column matrix associated to a $k\times n$ matrix $\boldsymbol{A}$ is denoted by $\overline{\boldsymbol{A}}$ and is defined by $\overline{\boldsymbol{A}}^T \stackrel{\text{def}}{=} \left(\boldsymbol{A}_1^T,\dots,\boldsymbol{A}_n^T\right)$. The identity matrix of size n is written as $\boldsymbol{I}_n$. The Kronecker product $\boldsymbol{A}\otimes\boldsymbol{B}$ of two matrices $\boldsymbol{A}=(a_{i,j})$ and $\boldsymbol{B}=(b_{i,j})$ is the matrix $(a_{i,j}\boldsymbol{B})$.

We denote by S_n the Symmetric group of all permutations on n letters. The image of an integer i by a permutation σ is represented by i^σ. However, in some occasions we will use the classical notation $\sigma(j)$. Notice that $\sigma\gamma$ where both σ and γ are permutations from S_n is the permutation of S_n defined for any i in $[1,n]\stackrel{\text{def}}{=}\{1,\dots,n\}$ by $i^{\sigma\gamma}\stackrel{\text{def}}{=}(i^\sigma)^\gamma$.

To each σ in S_n we associate a matrix $\boldsymbol{P}=(p_{i,j})$ from $\mathsf{GL}_n(\mathbb{R})$ with $p_{i,j}=1$ if $i=j^\sigma$ and $p_{i,j}=0$ otherwise. We will identify σ and $\boldsymbol{P}$ so that we will systematically write $\boldsymbol{P}\in S_n$, and the image of i by $\boldsymbol{P}$ will be $i^{\boldsymbol{P}}$ and sometimes $\boldsymbol{P}(i)$. The action of S_n over $\mathbb{F}^n$ is defined for any σ in S_n and $\boldsymbol{v}$ in $\mathbb{F}^n$ by $\boldsymbol{v}^\sigma\stackrel{\text{def}}{=}\boldsymbol{v}\boldsymbol{P}=\left(v_{\sigma(1)},\dots,v_{\sigma(n)}\right)$. We extend this convention for any subset $\mathscr{C}\subset\mathbb{F}^n$ by defining $\mathscr{C}^\sigma\stackrel{\text{def}}{=}\mathscr{C}\boldsymbol{P}\stackrel{\text{def}}{=}\left\{\boldsymbol{u}^\sigma:\boldsymbol{u}\in\mathscr{C}\right\}$. Notice that for any σ and γ from S_n we have by definition $\boldsymbol{v}^{\sigma\gamma}\stackrel{\text{def}}{=}(\boldsymbol{v}^\sigma)^\gamma$.

3.2 Our Modeling

In this section we introduce our algebraic modeling of PEP. The modeling includes linear and polynomial equations. We show that this modeling is sufficient to describe PEP. We also study the properties of the linear part of the system as well as the properties of the solution set. Before going into details, we first make the following assumptions in order to focus only on interesting instances.

Assumption 3.2.1. *We assume that $\mathscr{A}$ and $\mathscr{B}$ are two linear codes of length n and dimension k over a field $\mathbb{F}$ such that*

1. $h\stackrel{def}{=}\dim\mathcal{H}(\mathscr{A})=\dim\mathcal{H}(\mathscr{B})$.
2. $\mathbb{1}_n\in\mathscr{A}$ *(resp.* $\mathscr{A}^\perp$*) if and only if* $\mathbb{1}_n\in\mathscr{B}$ *(resp.* $\mathscr{B}^\perp$*).*
3. $\mathbb{1}_n\in\mathcal{H}(\mathscr{A})^\perp$ *if and only if* $\mathbb{1}_n\in\mathcal{H}(\mathscr{B})^\perp$.

Furthermore, without loss of generality, we also assume that $\mathscr{A}$ and $\mathscr{B}$ (resp. $\mathscr{A}^\perp$ and $\mathscr{B}^\perp$) have no zero coordinates (zero coordinates can be removed).

These assumptions are not restrictive. If one of the conditions is not satisfied, it is easy to decide the non-equivalence of the two codes. Conditions 2–3 come from the fact that $\mathbb{1}_n$ is invariant by any permutation. Actually, the vector space generated by $\mathbb{1}_n$ contains exactly all the vectors invariant by any permutation of S_n.

Proposition 3.2.2. *If the characteristic of the field $\mathbb{F}$ is 2 then $\mathbb{1}_n$ belongs to $\mathcal{H}(\mathscr{A})^{\perp}$.*

Proof. For any $\boldsymbol{a}$ in $\mathcal{H}(\mathscr{A})$ we have the following relations

$$0 = \langle \boldsymbol{a}, \boldsymbol{a} \rangle = \sum_{i=1}^{n} a_i^2 = \left(\sum_{i=1}^{n} a_i \right)^2 .$$

This equality implies that, for any $\boldsymbol{a}$ in $\mathcal{H}(\mathscr{A})$, $0 = \sum_{i=1}^{n} a_i = \langle \boldsymbol{a}, \mathbb{1}_n \rangle$ which means that $\mathbb{1}_n$ belongs to $\mathcal{H}(\mathscr{A})^{\perp}$. ■

3.2.1 Defining PEP

In this part we recall our definition of PEP in terms of generator matrices. The definition establishes a relationship between the generator matrices of the equivalent codes. In the light of this definition we also provide a relationship between the parity check matrices of the equivalent codes. Moreover we deduce a relationship between the generator matrix of the code and the parity check matrix of its equivalent code. This will give rise to build our algebraic system.

Definition 3.2.3. (**Permutation Equivalence Problem (PEP)**) Let $\mathscr{A}$ and $\mathscr{B}$ be two $[n, k]$ linear codes with $k \times n$ generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$, respectively. Are there a $k \times k$ matrix $\boldsymbol{S}$ and an $n \times n$ permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$?

This modeling has the disadvantage of seeking two unknown matrices $\boldsymbol{S}$ and $\boldsymbol{P}$ which leads to k^2+n^2 variables. We will show later how to improve this modeling and reduce the number of variables.

3.2.2 Properties of the Solution Set

In this section we show the relationship between the solution set of the equivalence of two linear codes and their permutation groups. Precisely we show that if the two codes are equivalent then the solution set is a coset to their permutation groups. Moreover the permutation groups of equivalent codes are isomorphic. The solutions set of the equivalence can be found from this isomorphism.

Proposition 3.2.4. *Let $\mathscr{A}$ be a linear code of length n, $\Pi(\mathscr{A})$ the permutation group of $\mathscr{A}$ and σ a permutation from the symmetric group S_n. The set of permutations γ in S_n such that $\mathscr{A} \xrightarrow{\gamma} \mathscr{A}^{\sigma}$ is exactly the set $\sigma\,\Pi(\mathscr{A}) \stackrel{def}{=} \{\sigma\tau \,:\, \tau \in \Pi(\mathscr{A})\}$.*

Proof. Clearly any $\sigma\Pi(\mathscr{A})$ is a solution to the equivalence problem between $\mathscr{A}$ and $\mathscr{A}^{\sigma}$. Now let us assume that γ is such that $\mathscr{A} \xrightarrow{\gamma} \mathscr{A}^{\sigma}$. Since by assumption we also have $\mathscr{A} \xrightarrow{\sigma} \mathscr{A}^{\sigma}$, we deduce that $\sigma^{-1}\gamma$ belongs to $\Pi(\mathscr{A})$, which terminates the proof. ■

The next proposition shows that the permutation groups of the equivalent codes are isomorphic.

Proposition 3.2.5. *The permutation groups of the permutationally equivalent codes are isomorphic.*

Proof. Let $\Pi(\mathscr{A})$ and $\Pi(\mathscr{B})$ be the permutation groups of the two equivalent codes $\mathscr{B} = \mathscr{A}^{\sigma}$. Define the mapping $\phi : \Pi(\mathscr{A}) \to \Pi(\mathscr{B})$ by $\phi(\pi) = \sigma\pi\sigma^{-1}$. It is easy to verify that ϕ is an isomorphism. ■

Remark 3.2.6. Proposition 3.2.5 shows that if we can find the permutation groups of equivalent codes we can find the solution of the equivalence by finding the isometry between the two permutation groups. This problem is called group isomorphism problem which is known to be not harder than graph isomorphism problem [65]. Thus it is not harder than code equivalence problem.

Remark 3.2.7. Note that we can look at the problem of finding the permutation group of a code as a special case of the PEP problem where $\mathscr{A} = \mathscr{B}$.

3.2.3 Permutation Equations

In this section we introduce algebraic equations that describe the permutation. Let $\boldsymbol{P} = (x_{i,j})$ be an $n \times n$ permutation matrix, then it satisfies $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}_n$ and the set of permutation matrices is exactly the set of solutions of the following system of polynomial equations:

$$\mathcal{P}erm = \begin{cases} \boldsymbol{P}\boldsymbol{P}^T - \boldsymbol{I}_n, & & (n^2 \text{ degree 2 equations}) \\ x_{i,j}^2 - x_{i,j}, & 1 \leqslant i,j \leqslant n & (\boldsymbol{P} \text{ has binary entries}) \\ x_{i,j}x_{i,j'}, & 1 \leqslant i,j,j' \leqslant n, j \neq j' & (\text{at most one non-zero element on each row}) \\ x_{i,j}x_{i',j}, & 1 \leqslant i,i',j \leqslant n, i \neq i' & (\text{at most one non-zero element on each column}) \\ \sum_{j'=1}^{n} x_{i,j'} - 1, & 1 \leqslant i \leqslant n & (\text{the non-zero element in each row is 1}) \\ \sum_{i'=1}^{n} x_{i',j} - 1, & 1 \leqslant j \leqslant n & (\text{the non-zero element in each column is 1}) \end{cases} \tag{3.1}$$

The following proposition shows that, as a system of generators, the previous equations contain redundancy, but we will see later that this redundancy may be useful for the computational point of view.

Proposition 3.2.8. *The set $V(\mathcal{P}erm)$ of solutions of the polynomial system $\mathcal{P}erm$ is exactly the set of permutation matrices. Moreover, the ideal $\langle \mathcal{P}erm \rangle$ admits a smaller set of generators :*

$$\langle \mathcal{P}erm \rangle = \left\langle \{x_{i,j}x_{i',j}\}_{1 \leqslant i,i',j \leqslant n, i \neq i'}, \left\{ \sum_{j'=1}^{n} x_{i,j'} - 1 \right\}_{1 \leqslant i \leqslant n} \right\rangle. \tag{3.2}$$

Proof. We prove only ($\subseteq$) since the other inclusion is clear. Indeed, we will show that the two associated varieties are equal, i.e. that any matrix $\boldsymbol{P}$ whose entries satisfy (1) $x_{i,j}x_{i',j} = 0 (1 \leqslant i,i',j \leqslant n, i \neq i')$ and (2) $\sum_{j'=1}^{n} x_{i,j'} = 1$ $(1 \leqslant i \leqslant n)$ is a permutation matrix. The equations in (1) show that each column is either zero or contains one non-zero element. Thus $\boldsymbol{P}$ contains at most n non-zero elements. The equations in (2) with (1) show that there must be exactly one non-zero element and this element equals to 1. There cannot be more than one 1 in a row otherwise there is a row sum up to zero. Thus $\boldsymbol{P}$ is a permutation matrix. ■

Remark 3.2.9. Notice that, for any j, if we multiply the polynomial $\sum_{j'=1}^{n} x_{i,j'} - 1$ by $x_{i,j}$ and reduce the result by the polynomials $x_{i,j}x_{i,j'}$ for $j' \neq j$, then we get the polynomial $x_{i,j}^2 - x_{i,j}$. Also

for row $\boldsymbol{p}_i$ in $\boldsymbol{P}$ we have $\boldsymbol{p}_i \cdot \boldsymbol{p}_{i'} = \mathbf{0}$ if $i \neq i'$ and $\boldsymbol{p}_i \cdot \boldsymbol{p}_i = 1$ thus $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}_n$. It is also possible to express the polynomials $x_{i,j}x_{i,j'}$ as a polynomial combination of (1) and (2), but the degree of the polynomials appearing in this expression are larger than 2. From the computational point of view, it is more interesting to start with a generating system containing many polynomials of small degree rather than with a minimal set of generators.

Remark 3.2.10. We also have

$$\langle \mathcal{P}erm \rangle = \left\langle \{x_{i,j}x_{i,j'}\}_{1\leqslant i,j,j'\leqslant n, j\neq j'}, \left\{ \sum_{i'=1}^{n} x_{i',j} - 1 \right\}_{1\leqslant j\leqslant n} \right\rangle. \tag{3.3}$$

3.2.4 Our Modeling for PEP

Consider the following algebraic system

$$\mathfrak{S}_{\min} \ : \begin{cases} \boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0} \\ \sum_{j'=1}^{n} x_{i,j'} = 1, 1 \leqslant i \leqslant n \\ x_{i,j}x_{i',j} = 0, i \neq i' \end{cases}$$

Note that, by Theorems 3.2.8 and 2.1.12, $\mathfrak{S}_{\min}$ generates a radical ideal. The following theorem proves that the system $\mathfrak{S}_{\min}$ fully describes PEP.

Theorem 3.2.11. *Let $\mathscr{A}$ and $\mathscr{B}$ be two $[n,k]$ linear codes with generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ and parity check matrices $\boldsymbol{H}_{\mathscr{A}}$ and $\boldsymbol{H}_{\mathscr{B}}$. Then $\mathscr{A}$ and $\mathscr{B}$ are permutationally equivalent by a permutation $\boldsymbol{P}$ if and only if $\boldsymbol{P}$ satisfies the system $\mathfrak{S}_{\min}$.*

Proof. ($\Rightarrow$). Since $\mathscr{A}$ and $\mathscr{B}$ are permutationally equivalent then there exists a permutation matrix $\boldsymbol{P}$ and an invertible matrix $\boldsymbol{S}$ such that $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$ according to Definition 3.2.1. We know that $\boldsymbol{G}_{\mathscr{B}}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$ thus $\boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$ and hence $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$. Thus if $\mathscr{A}$ and $\mathscr{B}$ are permutationally equivalent then the matrix $\boldsymbol{P}$ satisfies the system $\mathfrak{S}_{\min}$.

($\Leftarrow$). Now assume that we have a matrix $\boldsymbol{P}$ that satisfies the system $\mathfrak{S}_{\min}$. Then $\boldsymbol{P}$ is a permutation matrix by Proposition 3.2.8. $\boldsymbol{P}$ also satisfies $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$, and $\mathtt{rank}(\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}) = \mathtt{rank}(\mathscr{B})$ thus $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$ is a generator matrix for $\mathscr{B}$. Any generator matrix for $\mathscr{B}$, $\boldsymbol{G}_{\mathscr{B}}$, can be found through $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$ where $\boldsymbol{S}$ is $k \times k$ invertible matrix and this is our definition of the equivalence. Thus $\mathscr{A}$ and $\mathscr{B}$ are permutation equivalent by $\boldsymbol{P}$. ■

Remark 3.2.12. Note one can use the linear system $\boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{G}_{\mathscr{B}}^T = \mathbf{0}$ instead of $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$ using the same argument as in Theorem 3.2.11.

Corollary 3.2.13. *Let $\mathscr{A}$ and $\mathscr{B}$ be two $[n,k]$ linear codes with generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ and parity check matrices $\boldsymbol{H}_{\mathscr{A}}$ and $\boldsymbol{H}_{\mathscr{B}}$ respectively, then the following statements are equivalent*

- *$\mathscr{A}$ and $\mathscr{B}$ are permutation equivalent by a permutation $\boldsymbol{P}$.*
- *There exists an $n \times n$ permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T = \mathbf{0}$*
- *There exists an $n \times n$ permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{G}_{\mathscr{B}}^T = \mathbf{0}$*

Proof. This corollary follows form Theorem 3.2.11 and Proposition 1.3.1 which states, $\mathscr{B} = \mathscr{A}^{\pi} \Leftrightarrow \mathscr{B}^{\perp} = (\mathscr{A}^{\perp})^{\pi}$. ■

3.2.5 Properties of the Linear System

In this section we focus only on the linear part of the system and we show some of its properties. Although the system $\mathfrak{S}_{\min}$ is sufficient to describe PEP, but for the sake of enhancement of computation one might need to consider the maximum number of linear equations in the system. That is because if we start Groebner basis computation from the minimized system the other set of linear equations will be generated during the computation. Thus it will be useful to add these linear equations from the beginning. Thus $\mathfrak{S}_{\min}$ can be written:

$$\mathfrak{S} : \begin{cases} \boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T \\ \boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{G}_{\mathscr{B}}^T \\ \sum_{j'=1}^{n} x_{i,j'} - 1, & 1 \leqslant i \leqslant n \\ \sum_{i'=1}^{n} x_{i',j} - 1, & 1 \leqslant j \leqslant n \\ x_{i,j}x_{i',j}, & 1 \leqslant i, i', j \leqslant n, i \neq i' \\ x_{i,j}x_{i,j'}, & 1 \leqslant i, j, j' \leqslant n, j \neq j' \\ x_{i,j}^2 - x_{i,j}, & 1 \leqslant i, j \leqslant n. \end{cases}$$

We separate the linear system into two parts one from the equivalence and the other from the permutation.

$$\mathfrak{L} : \begin{cases} \boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{H}_{\mathscr{B}}^T &= \mathbf{0} \\ \boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{G}_{\mathscr{B}}^T &= \mathbf{0} \end{cases}$$

$$\mathfrak{P} : \begin{cases} \sum_{j'=1}^{n} x_{i,j'} = 1, & 1 \leqslant i \leqslant n \\ \sum_{i'=1}^{n} x_{i',j} = 1, & 1 \leqslant j \leqslant n \end{cases}$$

Definition 3.2.14. The linear system associated to PEP is $\mathfrak{E} = \mathfrak{L} \cup \mathfrak{P}$.

Next we define our system in terms of Kronecker product since it is more convenient for mathematical writing. Firstly, we recall well-known facts about Kronecker product of matrices.

Lemma 3.2.15. *[45] Let $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ be matrices over a field $\mathbb{F}$. Then the following properties hold:*

1. $\mathtt{rank}(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathtt{rank}(\boldsymbol{A})\ \mathtt{rank}(\boldsymbol{B})$.
2. $(\boldsymbol{A} \otimes \boldsymbol{B})\,(\boldsymbol{C} \otimes \boldsymbol{D}) = (\boldsymbol{A}\boldsymbol{C}) \otimes (\boldsymbol{B}\boldsymbol{D})$
3. *If $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible then $\boldsymbol{A} \otimes \boldsymbol{B}$ is invertible with $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$*
4. $(\boldsymbol{A} \otimes \boldsymbol{B})^T = \boldsymbol{A}^T \otimes \boldsymbol{B}^T$
5. *If $\boldsymbol{a}$ (resp. $\boldsymbol{b}$) is an eigenvector of $\boldsymbol{A}$ (resp. $\boldsymbol{B}$) with α (resp. β) in $\mathbb{F}$ is the associated eigenvalue then $\boldsymbol{a} \otimes \boldsymbol{b}$ is an eigenvector of $\boldsymbol{A} \otimes \boldsymbol{B}$ and $\alpha\beta$ is its associated eigenvalue.*

Lemma 3.2.16. *For any linear codes $\mathscr{A} \subset \mathbb{F}^{n_1}$ and $\mathscr{B} \subset \mathbb{F}^{n_2}$ we have*

$$(\mathscr{A} \otimes \mathbb{F}^{n_1})^{\perp} = \mathscr{A}^{\perp} \otimes \mathbb{F}^{n_1}. \tag{3.4}$$

$$(\mathbb{F}^{n_2} \otimes \mathscr{B})^{\perp} = \mathbb{F}^{n_2} \otimes \mathscr{B}^{\perp}. \tag{3.5}$$

Proposition 3.2.17. *Let $\mathscr{A} \subset \mathbb{F}^{n_1}$ and $\mathscr{B} \subset \mathbb{F}^{n_2}$ be two linear codes where n_1 and n_2 are integers $\geqslant 1$. We then have the following relations*

1. $(\mathscr{A} \otimes \mathbb{F}^{n_2}) \cap (\mathbb{F}^{n_1} \otimes \mathscr{B}) = \mathscr{A} \otimes \mathscr{B}$.
2. $(\mathscr{A} \otimes \mathscr{B})^{\perp} = \mathscr{A}^{\perp} \otimes \mathbb{F}^{n_2} + \mathbb{F}^{n_1} \otimes \mathscr{B}^{\perp}$.

Proof. Let us prove the first equality. Firstly, we clearly have the inclusion $\mathscr{A} \otimes \mathscr{B} \subset (\mathscr{A} \otimes \mathbb{F}^{n_2}) \cap (\mathbb{F}^{n_1} \otimes \mathscr{B})$. The other inclusion can be deduced by using (3.4) and observing that the only way for a codeword $\boldsymbol{b}$ from $\mathbb{F}^{n_1} \otimes \mathscr{B}$ to be in $\mathscr{A} \otimes \mathbb{F}^{n_2}$ is that $\boldsymbol{b}$ belongs to $\mathscr{A} \otimes \mathscr{B}$.

The second part is the dual of the first one using (3.4) and (3.5). ■

Corollary 3.2.18. *For any linear codes $\mathscr{A}_1 \subset \mathbb{F}^{n_1}$, $\mathscr{B}_1 \subset \mathbb{F}^{n_1}$ and $\mathscr{A}_2 \subset \mathbb{F}^{n_2}$, $\mathscr{B}_2 \subset \mathbb{F}^{n_2}$ we have*

$$(\mathscr{A}_1 \otimes \mathscr{A}_2) \cap (\mathscr{B}_1 \otimes \mathscr{B}_2) = (\mathscr{A}_1 \cap \mathscr{B}_1) \otimes (\mathscr{A}_2 \cap \mathscr{B}_2) \tag{3.6}$$

Proof. Let us set $\mathscr{C}^{\perp} = (\mathscr{A}_1 \otimes \mathscr{A}_2) \cap (\mathscr{B}_1 \otimes \mathscr{B}_2)$ then by using Equation (3.4) and (3.5) and Proposition 3.2.17 we have the following equalities

$$\begin{aligned} \mathscr{C} &= (\mathscr{A}_1 \otimes \mathscr{A}_2)^{\perp} + (\mathscr{B}_1 \otimes \mathscr{B}_2)^{\perp} \\ &= \mathscr{A}_1{}^{\perp} \otimes \mathbb{F}^{n_2} + \mathbb{F}^{n_1} \otimes \mathscr{A}_2{}^{\perp} + \mathscr{B}_1{}^{\perp} \otimes \mathbb{F}^{n_2} + \mathbb{F}^{n_1} \otimes \mathscr{B}_2{}^{\perp} \\ &= \left(\mathscr{A}_1{}^{\perp} + \mathscr{B}_1{}^{\perp}\right) \otimes \mathbb{F}^{n_2} + \mathbb{F}^{n_1} \otimes \left(\mathscr{B}_2{}^{\perp} + \mathscr{A}_2{}^{\perp}\right) \end{aligned}$$

which terminates the proof. ■

Lemma 3.2.19. *For any matrices $\boldsymbol{A} = (a_{i,j})$, $\boldsymbol{B} = (b_{i,j})$ and $\boldsymbol{C} = (c_{i,j})$, we have $\overline{\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}^T} = (\boldsymbol{C} \otimes \boldsymbol{A})\,\overline{\boldsymbol{B}}$.*

Proof. This can be proved by observing that for any matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D} = \boldsymbol{A}\boldsymbol{B}\boldsymbol{C}^T$ with the notation $\boldsymbol{C}^T = \left(c^T_{i,j}\right)$ we have $d_{i,j} = \sum_k (\boldsymbol{a}_i \boldsymbol{B}_k)\, c^T_{k,j} = \sum_k c_{j,k} \boldsymbol{a}_i\, \boldsymbol{B}_k = (\boldsymbol{c}_j \otimes \boldsymbol{a}_i)\,\overline{\boldsymbol{B}}$. In particular $\boldsymbol{D}_j = (\boldsymbol{c}_j \otimes \boldsymbol{A})\ \overline{\boldsymbol{B}}$ which implies $\overline{\boldsymbol{D}} = (\boldsymbol{C} \otimes \boldsymbol{A})\,\overline{\boldsymbol{B}}$ which concludes the proof. ■

Corollary 3.2.20. *The linear system $\mathfrak{L}$ can be rewritten as follows in terms of Kronecker product:*

$$\mathfrak{L} : \begin{cases} \left(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}}\right) \overline{\boldsymbol{P}} &= \boldsymbol{0} \\ \left(\boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}}\right) \overline{\boldsymbol{P}} &= \boldsymbol{0} \end{cases}$$

Proposition 3.2.21. *Any $n \times n$ permutation matrix $\boldsymbol{P}$ is a solution of the linear system $\mathfrak{P}$:*

$$\mathfrak{P} : \begin{cases} \left(\mathbb{1}_n \otimes \boldsymbol{I}_n\right) \overline{\boldsymbol{P}} &= \mathbb{1}_n^T \\ \left(\boldsymbol{I}_n \otimes \mathbb{1}_n\right) \overline{\boldsymbol{P}} &= \mathbb{1}_n^T \end{cases}$$

The rank of $\mathfrak{P}$ is $2n - 1$.

Proof. The value of the rank comes from the fact that by adding all the rows of $\mathbb{1}_n \otimes \boldsymbol{I}_n$ we obtain $\mathbb{1}_{n^2}$. Similarly we obtain this result by adding all the rows of $\boldsymbol{I}_n \otimes \mathbb{1}_n$. It is easy to see that these two combinations provide the only non-zero row combination in $\mathfrak{P}$ that results in zero. ■

Lemma 3.2.22. *Let $\boldsymbol{G}$ and $\boldsymbol{H}$ be respectively the generator matrix and the parity check matrix of a code $\mathscr{C}$ of length n. Let $\boldsymbol{D} = \begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{H} \end{pmatrix}$. A parity check matrix of $\mathcal{H}(\mathscr{C})$ is $\boldsymbol{D}$ and in particular* $\mathtt{rank}(\boldsymbol{D}) = n - \dim \mathcal{H}(\mathscr{C})$.

Proposition 3.2.23. *Let $h_{\mathscr{A}}$ and $h_{\mathscr{B}}$ be the dimensions of $\mathcal{H}(\mathscr{A})$ and $\mathcal{H}(\mathscr{B})$, respectively. Then we have*

$$\mathtt{rank}(\mathfrak{L}) = 2k(n-k) - h_{\mathscr{A}} h_{\mathscr{B}}.$$

Proof. That the matrix defining the linear system $\mathfrak{L}$ is actually

$$\begin{bmatrix} \boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \\ \boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}} \end{bmatrix}.$$

This is a generator matrix of the linear code $\mathscr{B} \otimes \mathscr{A}^{\perp} + \mathscr{B}^{\perp} \otimes \mathscr{A}$. Hence by Corollary 3.2.18 we have

$$\begin{aligned} \dim \left(\mathscr{B} \otimes \mathscr{A}^{\perp} + \mathscr{B}^{\perp} \otimes \mathscr{A} \right) &= \dim \mathscr{B} \otimes \mathscr{A}^{\perp} + \dim \mathscr{B}^{\perp} \otimes \mathscr{A} - \dim \left(\mathscr{B} \otimes \mathscr{A}^{\perp} \right) \cap \left(\mathscr{B}^{\perp} \otimes \mathscr{A} \right) \\ &= 2k(n-k) - \dim \mathcal{H}(\mathscr{A}) \otimes \mathcal{H}(\mathscr{B}). \end{aligned}$$

This last equality gives exactly the required result and finishes the proof. ■

Remark 3.2.24. An obvious upper-bound for the rank of $\mathfrak{E}$ would be

$$\mathtt{rank}(\mathfrak{E}) \leqslant 2k(n-k) - h^2 + 2n - 1.$$

Theorem 3.2.25. *Let $\mathscr{C}$ be either $\mathscr{A}$ or $\mathscr{B}$ and*

$$\delta = 2k(n-k) - h^2 + 2n - 1 - \mathtt{rank}(\mathfrak{E}).$$

Then $0 \leqslant \delta \leqslant 2n-1$, and we have:

1. *If $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})$ then $\delta \geqslant 2(n-h) - 1$.*
2. *If $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})$ and $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})^{\perp}$ then*

$$\delta \geqslant \begin{cases} 2(n-k) & \text{if} \quad \mathbb{1}_n \in \mathscr{C}, \\ 2k & \text{if} \quad \mathbb{1}_n \in \mathscr{C}^{\perp}, \\ 2h+1 & \text{if} \quad \mathbb{1}_n \notin \mathscr{C} \text{ and } \mathbb{1}_n \notin \mathscr{C}^{\perp} \end{cases}$$

3. *If $\delta = 0$ then $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})^{\perp}$.*

Proof. Let $\boldsymbol{U}_n$ and $\boldsymbol{T}$ be the matrices corresponding to the linear systems $\mathfrak{P}$ and $\mathfrak{L}$, respectively. From Proposition 3.2.21 and Proposition 3.2.23 we know the exact rank of $\boldsymbol{U}_n$ and $\boldsymbol{T}$. Let U and T be the vector spaces generated by $\boldsymbol{U}_n$ and $\boldsymbol{T}$, respectively. Our target is to improve the bound of Remark 3.2.24 by finding elements in $U \cap T$.

Firstly assume $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})$. Without loss of generality we can assume $\mathbb{1}_n$ is a row in $\boldsymbol{G}_{\mathscr{C}}$ and $\boldsymbol{H}_{\mathscr{C}}$. Consider the vector space V which is generated by

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{H}_{\mathscr{B}} \otimes \mathbb{1}_n \\ \boldsymbol{G}_{\mathscr{B}} \otimes \mathbb{1}_n \\ \mathbb{1}_n \otimes \boldsymbol{G}_{\mathscr{A}} \\ \mathbb{1}_n \otimes \boldsymbol{H}_{\mathscr{A}}. \end{pmatrix}$$

Clearly, $V \subseteq T \cap U$. The proof terminates by observing that the dimension of V is $2(n-h)-1$. That is because $\texttt{rank}\begin{pmatrix}\boldsymbol{G}_{\mathscr{C}}\\ \boldsymbol{H}_{\mathscr{C}}\end{pmatrix} = (n-h)$ and $\mathbb{1}_n \otimes \mathbb{1}_n$ is a duplicated row in $\boldsymbol{V}$. The remaining rows in $\boldsymbol{V}$ are independent.

Next assume that $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})$, $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})^{\perp}$ and $\mathbb{1}_n \in \mathscr{C}$. Consider the vector space V which is generated by

$$\boldsymbol{V} = \begin{pmatrix}\boldsymbol{H}_{\mathscr{B}} \otimes \mathbb{1}_n\\ \mathbb{1}_n \otimes \boldsymbol{H}_{\mathscr{A}}.\end{pmatrix}$$

Clearly, $V \subseteq T \cap U$ and the dimension of V is $2(n-k)$ since $\boldsymbol{V}$ is of full rank.

The case $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})$, $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})^{\perp}$ and $\mathbb{1}_n \in \mathscr{C}^{\perp}$ can be proved similarly where we get the $dim(T \cap U) \geqslant 2k$.

Assume that $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})$, $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})^{\perp}$, $\mathbb{1}_n \notin \mathscr{C}$ and $\mathbb{1}_n \notin \mathscr{C}^{\perp}$. Thus $\mathbb{1}_n = \boldsymbol{g}_{\mathscr{C}} + \boldsymbol{h}_{\mathscr{C}}$ with $0 \neq \boldsymbol{g}_{\mathscr{C}} \in \mathscr{C}$, $0 \neq \boldsymbol{h}_{\mathscr{C}} \in \mathscr{C}^{\perp}$ and $\mathscr{C} \in \{\mathscr{A}, \mathscr{B}\}$. Then we can write

$$\boldsymbol{h}_{\mathscr{B}} \otimes \boldsymbol{g}_{\mathscr{A}} = (\mathbb{1}_n - \boldsymbol{g}_{\mathscr{B}}) \otimes \boldsymbol{g}_{\mathscr{A}}. \tag{3.7}$$

Similarly,

$$\boldsymbol{g}_{\mathscr{B}} \otimes \boldsymbol{h}_{\mathscr{A}} = \boldsymbol{g}_{\mathscr{B}} \otimes (\mathbb{1}_n - \boldsymbol{g}_{\mathscr{A}}). \tag{3.8}$$

Then we can write

$$\mathbb{1}_n \otimes \boldsymbol{g}_{\mathscr{A}} - \boldsymbol{g}_{\mathscr{B}} \otimes \mathbb{1}_n = \boldsymbol{h}_{\mathscr{B}} \otimes \boldsymbol{g}_{\mathscr{A}} - \boldsymbol{g}_{\mathscr{B}} \otimes \boldsymbol{h}_{\mathscr{A}} \in T \cap U. \tag{3.9}$$

We denote by E and E' the vector space generated by $\mathbb{1}_n \otimes \boldsymbol{g}_{\mathscr{A}} - \boldsymbol{g}_{\mathscr{B}} \otimes \mathbb{1}_n$ and $\begin{pmatrix}\boldsymbol{g}_{\mathscr{B}} \otimes \boldsymbol{h}_{\mathscr{A}}\\ \boldsymbol{h}_{\mathscr{B}} \otimes \boldsymbol{g}_{\mathscr{A}}\end{pmatrix}$, respectively. Clearly $E \subseteq E'$.

Now let $\boldsymbol{u}_{\mathscr{B}} \in \mathcal{H}(\mathscr{B})$, thus we can write

$$\begin{pmatrix}\boldsymbol{u}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}}\\ \boldsymbol{u}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}}\end{pmatrix} = \boldsymbol{u}_{\mathscr{B}} \otimes \begin{pmatrix}\boldsymbol{G}_{\mathscr{A}}\\ \boldsymbol{H}_{\mathscr{A}}\end{pmatrix}$$

Since $\mathbb{1}_n \in \mathcal{H}(\mathscr{C})^{\perp}$, thus for all $\boldsymbol{u}_{\mathscr{B}}$ in $\mathcal{H}(\mathscr{B})$ we have

$$\boldsymbol{u}_{\mathscr{B}} \otimes \mathbb{1}_n \in T \cap U.$$

Similarly for all $\boldsymbol{u}_{\mathscr{A}}$ in $\mathcal{H}(\mathscr{A})$ we have

$$\mathbb{1}_n \otimes \boldsymbol{u}_{\mathscr{A}} \in T \cap U.$$

Let V_1 and V_2 be the vector spaces generated by $\boldsymbol{G}_{\mathcal{H}(\mathscr{B})} \otimes \mathbb{1}_n$ and $\mathbb{1}_n \otimes \boldsymbol{G}_{\mathcal{H}(\mathscr{A})}$, respectively with $\boldsymbol{G}_{\mathcal{H}(\mathscr{C})}$ the generator matrix of $\mathcal{H}(\mathscr{C})$. Thus $V_i \subseteq T \cap U$ for $i \in \{1, 2\}$.

To prove that $V_1 \cap V_2 = \{\boldsymbol{0}\}$ it is sufficient to notice they are constructed from the basis of the hull and $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})$.

We terminate by proving that $dim(V_1 + V_2 + E) = 2h + 1$ where we need only to prove that $(V_1 + V_2) \cap E = \{\boldsymbol{0}\}$. Note that $(V_1 + V_2) \cap E' = \{\boldsymbol{0}\}$ since $\boldsymbol{g}_{\mathscr{B}}, \boldsymbol{h}_{\mathscr{B}} \notin \mathcal{H}(\mathscr{B})$ and $\boldsymbol{g}_{\mathscr{A}}, \boldsymbol{h}_{\mathscr{A}} \notin \mathcal{H}(\mathscr{A})$. Thus $(V_1 + V_2) \cap E = \{\boldsymbol{0}\}$ since $E \subseteq E'$.

Finally if $\delta = 0$ then we are not in any of the previous cases and thus $\mathbb{1}_n \notin \mathcal{H}(\mathscr{C})^{\perp}$. ■

3.3 PEP and Groebner Basis

Groebner basis techniques are well-known and well-studied and applied in solving system of multivariate polynomials. Here we consider these techniques to solve PEP since our modeling for the problem is a set of multivariate linear and quadratic polynomials. For brief literature about Groebner basis refer to Chapter 2 and for detailed description refer to the book [25] and for the algorithms F_4 and F_5 refer to [30, 31].

Recall our polynomial system $\mathfrak{S}$ for PEP as introduced in the Section Our Modeling for PEP

$$\mathfrak{S}\ :\ \begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\ \overline{\boldsymbol{P}}, & \\ (\boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}})\ \overline{\boldsymbol{P}}, & \\ (\mathbb{1}_n^T \otimes \boldsymbol{I}_n)\ \overline{\boldsymbol{P}}, & \\ (\boldsymbol{I}_n \otimes \mathbb{1}_n^T)\ \overline{\boldsymbol{P}}, & \\ x_{i,j}x_{i',j}, & i \neq i' \\ x_{i,j}x_{i',j}, & 1 \leqslant i, i', j \leqslant n, i \neq i' \\ x_{i,j}x_{i,j'}, & 1 \leqslant i, j, j' \leqslant n, j \neq j' \\ x_{i,j}^2 - x_{i,j}, & 1 \leqslant i, j \leqslant n. \end{cases}$$

As before we denote the linear part of $\mathfrak{S}$ by $\mathfrak{E}$. It is convenient here to consider the system with the maximum number of linear equations. That is because instead of letting Groebner basis computation generates these linear equations we input them directly in the initial system. Doing this enables to eliminate larger number of variables before starting Groebner basis computation.

As we have seen previously $2k(n-k) - h^2 \leqslant \mathtt{rank}(\mathfrak{E}) \leqslant 2k(n-k) - h^2 + 2n - 1 - \delta$ with δ as in Theorem 3.2.25. By experiments for random codes $\mathtt{rank}(\mathfrak{E})$ is almost identical to the upper bound.

We have proved in Theorem 3.2.11 that the system $\mathfrak{S}$ fully describes PEP. Thus the solutions of $\mathfrak{S}$ are exactly the permutations fulfilling the equivalence between the two codes. Thus using Groebner basis we can always solve $\mathfrak{S}$ and retrieve the set of permutations between the permutation equivalent codes. If the reduced Groebner basis is $\{1\}$ then the system has no solution and the two codes are not permutation equivalent. The complexity of Groebner basis computation for PEP depends on many factors that are discussed next.

3.3.1 Factors Affecting the Computation

There are many factors affecting the computation of Groebner basis. The first factor is the length of the code n which directly affects the number of unknowns which is n^2. Thus as the length of the code increases the number of quadratic equations increases and the size of matrices in Groebner basis computation increases.

The second factor is the rank of $\mathfrak{E}$ and this is in turn affected by the rate of the code and the size of the hull. As long as the rank of $\mathfrak{E}$ is large we can eliminate more variables from the system and this enhances Groebner basis computation significantly. The best rank is achieved when the hull is trivial and the rate of the code is 0.5 and it decreases as the rate goes far from 0.5, i.e. as the rate of the code or its dual tends to zero (note that the high rate and low rate codes behaves the same way since the system uses the code and its dual). In case of self-dual codes we have $h = k = \frac{n}{2}$ thus the rank of the

n	k	h	#Sol	Deg	Time (s)	Memory (MB)
10	2	0	288	3	$2^{-3.5}$	2^{5}
	2	0	2,880	5	$2^{0.6}$	$2^{6.5}$
		0	4	2	$2^{-4.6}$	2^{5}
	5	2	8	2	$2^{-4.3}$	2^{5}
		5	2,688	4	$2^{4.5}$	$2^{7.5}$
20	5	0	384	3	$2^{5.5}$	$2^{9.2}$
		0	1	2	$2^{4.3}$	$2^{9.7}$
	10	2	1	2	$2^{4.7}$	$2^{9.6}$
		5	1	2	$2^{5.6}$	$2^{9.0}$
		9	6,144	4	$> 2^{17.3}$	$> 2^{14.6}$
30	7	0	16	2	$2^{9.5}$	$2^{13.4}$
	15	3	1	2	$2^{10.1}$	$2^{14.8}$
		7	1	2	$2^{11.6}$	$2^{14.3}$

Table 3.1: Solving PEP for random codes over $\mathbb{F}_2$ with F_4 algorithm using the system $\mathfrak{S}$

linear system is upper bounded by $\frac{n^2}{4} + 2n - 1 - \delta$. Hence when the code is self-dual or has large hull we loose a significant number of linear equations and this in turn affects the computation.

The third factor is the size of the field since computation in small fields especially the binary field is much faster and less memory consuming. Thus as we have our codes defined in a large field the complexity of Groebner basis computation increases. The extension fields has some advantage since we can use Frobenius action to improve the linear part. The arithmetic operations complexity on the field is also affected by the implementation of the field operation in the computation algebra software.

The fourth factor is the size of the solution set where large solution sets result in a higher degrees of polynomials appearing during Groebner basis computation. As we mentioned in Groebner basis chapter (Chapter 2) the degrees of polynomials appearing during the computation is the main player affecting the total complexity.

3.3.2 Experimental Results

Table 3.1, 3.2 and 3.3 are experiments using Magma to solve PEP with our modeling using F_4 algorithm. The experiments confirm the effect of the length of the codes, the rate, the hull, the size of the solution set and the field in the computation.

One can notice from the experiments, the increase in the length of the code increases the complexity of solving. Similarly, the increase in the size of the hull also increases the complexity where the worst complexity is obtained for self-dual codes. Solving equivalence problem for codes with large solution set is more complex than the solving for codes with small solution set since large solution sets result in an increase of the degree of polynomials. The rate of the code also affects the complexity where small rates have higher complexity. Solving PEP using Groebner basis over extension field performs better in many cases specially for codes with large hull.

n	k	h	#Sol	Deg	Time (s)	Memory (MB)
10	2	0	12	2	$2^{-3.6}$	2^5
		0	1	2	$2^{-3.3}$	2^5
	5	2	1	2	$2^{-3.3}$	2^5
		5	24	3	$2^{2.6}$	2^6
20	5	0	1	2	$2^{5.3}$	$2^{11.1}$
		0	1	2	$2^{5.7}$	$2^{11.8}$
	10	2	1	2	$2^{5.5}$	$2^{11.5}$
		5	1	2	$2^{5.6}$	$2^{11.1}$
30	7	0	1	2	$2^{10.4}$	$2^{15.6}$
	15	3	1	2	$2^{11.2}$	$2^{16.6}$
		7	1	2	$2^{11.1}$	$2^{16.2}$

Table 3.2: Solving PEP for random codes over $\mathbb{F}_5$ with F_4 algorithm using the system $\mathfrak{S}$

n	k	h	#Sol	Deg	Time (s)	Memory (MB)
10	2	0	2	2	$2^{-4.6}$	2^5
	5	0	1	2	$2^{-3.8}$	2^5
		2	1	2	$2^{-3.6}$	2^5
		5	1	2	$2^{-1.9}$	2^5
20	5	0	1	2	$2^{4.9}$	$2^{11.1}$
	10	0	1	2	2^5	$2^{12.0}$
		2	1	2	$2^{4.8}$	$2^{11.7}$
		5	1	2	$2^{5.0}$	$2^{11.2}$
		10	1	2	$2^{6.8}$	$2^{10.7}$
30	7	0	1	2	$2^{10.1}$	$2^{16.0}$
	15	3	1	2	$2^{10.1}$	$2^{16.7}$
		7	1	2	$2^{10.3}$	$2^{16.4}$

Table 3.3: Solving PEP for random codes over $\mathbb{F}_8$ with F_4 algorithm using the system $\mathfrak{S}$

n	k	h	Deg	Time (s)	Memory (MB)
10	2	0	2	$2^{-4.6}$	2^{5}
		0	2	$2^{-5.6}$	2^{5}
	5	2	2	$2^{-5.6}$	2^{5}
		5	2	$2^{-3.5}$	2^{5}
20	5	0	2	$2^{3.5}$	$2^{8.5}$
		0	2	$2^{2.7}$	$2^{9.3}$
	10	2	2	$2^{4.2}$	$2^{9.5}$
		5	2	$2^{4.4}$	$2^{8.8}$
		9	3	$2^{15.0}$	$2^{14.2}$
30	7	0	2	$2^{8.8}$	$2^{13.3}$
	15	3	2	$2^{8.1}$	$2^{14.7}$
		7	2	$2^{10.0}$	$2^{14.5}$

Table 3.4: Deciding inequivalence for random codes over $\mathbb{F}_2$ with F_4 using the system $\mathfrak{S}$

Running the experiments with the same parameters for non-equivalent codes has better complexity. The maximum degree reached during the computation does not exceed two for codes with not large hull since the solution set is empty. The hull of the code seems to play an important role in the total complexity since one can notice from the experiments the large difference in time and memory complexity for codes with large hull, see Table 3.4, 3.5 and 3.6.

3.3.3 Estimating the Complexity of the Groebner basis Computation

We define the algebraic system associated to the quadratic equations as follows

$$\mathfrak{Q} \stackrel{\text{def}}{=} \bigcup_{i=1}^{n}\bigcup_{j=1}^{n}\left\{x_{i,j}^2 - x_{i,j}\right\} \bigcup_{j=1}^{n}\bigcup_{i\neq i'}\left\{x_{i,j}x_{i',j}\right\} \bigcup_{i=1}^{n}\bigcup_{j\neq j'}\left\{x_{i,j}x_{i,j'}\right\}. \tag{3.10}$$

In our case, we can remark that the polynomial ideal contains the equations from $\mathfrak{Q}$, see Section 3.2.3, which is already a Groebner basis. To simplify the computations and reduce the size of the matrices, we will work in the quotient ring

$$\mathcal{R}[\boldsymbol{X}] = \mathbb{F}[\boldsymbol{X}]/(\mathfrak{Q}) \tag{3.11}$$

where $\boldsymbol{X}$ is seen as the set of variables $\{x_{i,j}\}_{i,j\in[1,n]}$. This means that all the computations will be done modulo the polynomials in $\mathfrak{Q}$. Notice that, as $\mathfrak{Q}$ is already a Groebner basis for the chosen monomial ordering, computing a Groebner basis of $\mathfrak{E}\cup\mathfrak{Q}$ in $\mathbb{F}[\boldsymbol{X}]$ is done by computing a Groebner basis of $\mathfrak{E}$ modulo $\mathfrak{Q}$, using the Normal Form.

The first step in the Groebner basis computation consists in computing the row echelon form of the linear system $\mathfrak{E}$ modulo $\mathfrak{Q}$. This linear system has $N_r = 2k(n-k)+2n = 2R(1-R)n^2+2n$ rows and $N_c = n^2+1$ columns if we assume the rate $R=\frac{k}{n}$ of the codes is a constant. The cost of the Gaussian elimination is $O(n^{2\omega})$ where ω is the cost of matrix multiplication. For simplicity,

n	k	h	Deg	Time (s)	Memory (MB)
10	2	0	2	$2^{-3.8}$	2^5
		0	2	$2^{-3.8}$	2^5
	5	2	2	$2^{-3.8}$	2^5
		5	2	$2^{-3.3}$	2^5
20	5	0	2	$2^{4.7}$	$2^{10.7}$
		0	2	$2^{5.1}$	$2^{11.5}$
	10	2	2	$2^{5.0}$	$2^{11.5}$
		5	2	2^5	$2^{11.1}$
30	7	0	2	$2^{9.7}$	$2^{15.8}$
	15	3	2	$2^{10.5}$	$2^{16.7}$
		7	2	$2^{10.3}$	$2^{16.2}$

Table 3.5: Deciding inequivalence for random codes over $\mathbb{F}_5$ with F_4 using the system $\mathfrak{S}$

n	k	h	Deg	Time (s)	Memory (MB)
10	2	0	2	$2^{-5.1}$	2^5
	5	0	2	$2^{-4.6}$	2^5
		2	2	$2^{-5.1}$	2^5
		5	2	$2^{-2.6}$	2^5
20	5	0	2	$2^{3.6}$	$2^{11.1}$
	10	0	2	$2^{3.7}$	$2^{12.1}$
		2	2	$2^{3.3}$	$2^{11.7}$
		5	2	$2^{3.5}$	$2^{11.2}$
		10	2	$2^{5.5}$	$2^{10.4}$
30	7	0	2	$2^{8.9}$	$2^{16.0}$
	15	3	2	$2^{8.6}$	$2^{16.8}$
		7	2	$2^{8.7}$	$2^{16.4}$

Table 3.6: Deciding inequivalence for random codes over $\mathbb{F}_8$ with F_4 using the system $\mathfrak{S}$

if we take $\omega = 3$ the complexity is $O(n^6)$. After this first step, from Theorem 3.2.25 we have $2k(n-k) - h^2 + 2n - 1 - \delta = O(n^2)$ independent linear equations.

The Macaulay matrix in degree 2 consists in all products of a linear equation by a variable $x_{i,j}$ (modulo $\mathfrak{Q}$). There are n^2 different variables, the matrix we construct in degree 2 contains $O(n^4)$ rows. The columns consist of the monomials of degree 0 and 1, and the monomials $x_{i,j}x_{s,t}$ with $i,j,s,t \in [1,n]$ in $\mathcal{R}[\boldsymbol{X}]$ it excludes the cases $(i,j) = (s,t)$ or ($i = s$ and $j \neq t$) or ($j = t$ and $i \neq s$). The total number of monomials is $\binom{n^2}{2} + n^2 + 1 - 2n\binom{n}{2} = O(n^4)$.

We get a matrix with $O(n^4)$ rows and $O(n^4)$ columns. The cost of a row echelon form is $O(n^{4\omega})$, which is $O(n^{12})$ for a simple Gaussian elimination.

Theorem 3.3.1. *A Groebner basis computation for the system $\mathfrak{S}$ with Buchberger, F_4 or F_5 algorithms will have to compute an equivalent of the echelon form of the Macaulay matrices in degree 1 and 2, resulting in $O(n^{4\omega})$ operations in $\mathbb{F}$.*

As it is noticed from the experiments in Table 3.1 - 3.6, when the hull is trivial and the solution set contains one element the maximum degree reached by computation does not exceed 2. We will give a method, called block linearization, to compute in this case the Groebner basis without computing the echelon form in degree 2. The cost of this method is $O(n^{2\omega+1})$ instead of $O(n^{4\omega})$.

3.4 Block Linearization

In this section we introduce a new improvement of the way of solving the system $\mathfrak{S}$. We show how to compute linear equations in the system without computing the degree 2 row echelon form which can be hard, even if it has a polynomial complexity. In some cases we do not need to go further and we can find the solution of the equivalence directly, while other cases need more work to get the solutions. The interesting thing is that this method is quite efficient in practice.

Recall that the Hull $\mathcal{H}(\mathscr{C})$ of $\mathscr{C} \subset \mathbb{F}^n$ is the linear code $\mathscr{C} \cap \mathscr{C}^\perp$. Moreover, since $\mathcal{H}(\mathscr{C})^\perp = \mathscr{C} + \mathscr{C}^\perp$, a parity check matrix of $\mathcal{H}(\mathscr{C})$ is given by

$$\boldsymbol{H}_{\mathcal{H}(\mathscr{C})} = \begin{pmatrix} \boldsymbol{G}_{\mathscr{C}} \\ \boldsymbol{H}_{\mathscr{C}} \end{pmatrix}.$$

In particular, $\mathcal{H}(\mathscr{C}) = \{\mathbf{0}\}$ if and only if $\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}$ is invertible. We recall an elementary fact when dealing with codes over fields that are included in $\mathbb{R}$.

Proposition 3.4.1. *Let $\mathbb{F}$ be a subfield of $\mathbb{R}$. Any linear code $\mathscr{C}$ over $\mathbb{F}$ has a trivial hull.*

Proof. This comes from the fact that over $\mathbb{R}$, the inner product is definite. ■

The following proposition explains precisely how to characterize codes with trivial hulls over any field.

Proposition 3.4.2. *A linear code $\mathscr{C}$ has a trivial hull if and only if $\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T$ is invertible, and the inverse of $\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}$ is then given by*

$$\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}^{-1} = \left[\boldsymbol{G}_{\mathscr{C}}^T\left(\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T\right)^{-1} \quad \boldsymbol{H}_{\mathscr{C}}^T\left(\boldsymbol{H}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T\right)^{-1}\right]. \tag{3.12}$$

Proof. Let us assume that $\mathscr{C}$ has dimension k. The matrix $\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T$ is then not invertible if and only if there exists $\boldsymbol{x}$ in $\mathbb{F}^k$ such that both $\boldsymbol{x}\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T = \boldsymbol{0}$ and $\boldsymbol{x} \neq \boldsymbol{0}$ hold. This is equivalent to $\boldsymbol{x}\boldsymbol{G}_{\mathscr{C}} \neq \boldsymbol{0}$ and $\boldsymbol{x}\boldsymbol{G}_{\mathscr{C}}$ belongs to $\mathscr{C} \cap \mathscr{C}^\perp = \mathcal{H}(\mathscr{C})$. By symmetry, we also have that $\boldsymbol{H}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T$ is invertible. The expression of $\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}^{-1}$ follows immediately from these facts. ■

Another important property when $\mathscr{C}$ has a trivial hull is the possibility to write $\mathbb{F}^n = \mathscr{C} \oplus \mathscr{C}^\perp$. This means that for each $\boldsymbol{v}$ in $\mathbb{F}^n$ there exist a unique $\boldsymbol{v}_{\mathscr{C}}$ in $\mathscr{C}$ and a unique $\boldsymbol{v}_{\mathscr{C}^\perp}$ in $\mathscr{C}^\perp$ such that $\boldsymbol{v} = \boldsymbol{v}_{\mathscr{C}} + \boldsymbol{v}_{\mathscr{C}^\perp}$. These elements can be seen as the projections of $\boldsymbol{v}$ on $\mathscr{C}$ and $\mathscr{C}^\perp$. In order to provide their expression, we need to define the following $n \times n$ matrix $\boldsymbol{\Sigma}_{\mathscr{C}}$ where

$$\boldsymbol{\Sigma}_{\mathscr{C}} \stackrel{\text{def}}{=} \boldsymbol{G}_{\mathscr{C}}^T\Big(\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T\Big)^{-1}\boldsymbol{G}_{\mathscr{C}}. \tag{3.13}$$

Remark 3.4.3. We also define $\boldsymbol{\Sigma}_{\mathscr{C}^\perp} = \boldsymbol{H}_{\mathscr{C}}^T\Big(\boldsymbol{H}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T\Big)^{-1}\boldsymbol{H}_{\mathscr{C}}$.

Proposition 3.4.4. *Let $\mathscr{C}$ be a linear code with trivial hull. For any $\boldsymbol{v}$ in $\mathbb{F}^n$ the unique $\boldsymbol{v}_{\mathscr{C}}$ in $\mathscr{C}$ and a unique $\boldsymbol{v}_{\mathscr{C}^\perp}$ in $\mathscr{C}^\perp$ such that $\boldsymbol{v} = \boldsymbol{v}_{\mathscr{C}} + \boldsymbol{v}_{\mathscr{C}^\perp}$ are defined as*

$$\boldsymbol{v}_{\mathscr{C}} = \boldsymbol{v}\,\boldsymbol{\Sigma}_{\mathscr{C}} \quad \textit{and} \quad \boldsymbol{v}_{\mathscr{C}^\perp} = \boldsymbol{v}\,\boldsymbol{\Sigma}_{\mathscr{C}^\perp}. \tag{3.14}$$

Proof. We assume that $\mathscr{C}$ is of dimension k. It follows from the definition of $\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}$ that $\boldsymbol{v}_{\mathscr{C}}$ and $\boldsymbol{v}_{\mathscr{C}^\perp}$ can be computed by finding $\boldsymbol{x}_{\mathscr{C}}$ from $\mathbb{F}^k$ and $\boldsymbol{x}_{\mathscr{C}^\perp}$ from $\mathbb{F}^{n-k}$ such that $\boldsymbol{v} = (\boldsymbol{x}_{\mathscr{C}}, \boldsymbol{x}_{\mathscr{C}^\perp})\,\boldsymbol{H}_{\mathcal{H}(\mathscr{C})} = \boldsymbol{x}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}} + \boldsymbol{x}_{\mathscr{C}^\perp}\boldsymbol{H}_{\mathscr{C}}$, and consequently $\boldsymbol{v}_{\mathscr{C}} = \boldsymbol{x}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}$ and $\boldsymbol{v}_{\mathscr{C}^\perp} = \boldsymbol{x}_{\mathscr{C}^\perp}\boldsymbol{H}_{\mathscr{C}}$. By assumption $\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}$ is invertible, and hence, by keeping the k first coordinates of $\boldsymbol{v}\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}^{-1}$ and by using (3.12) we obtain then $\boldsymbol{x}_{\mathscr{C}} = \boldsymbol{v}\,\boldsymbol{G}_{\mathscr{C}}^T\Big(\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T\Big)^{-1}$. The last $n-k$ coordinates of $\boldsymbol{v}\boldsymbol{H}_{\mathcal{H}(\mathscr{C})}^{-1}$ give $\boldsymbol{x}_{\mathscr{C}^\perp} = \boldsymbol{v}\,\boldsymbol{H}_{\mathscr{C}}^T\Big(\boldsymbol{H}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T\Big)^{-1}$. ■

Proposition 3.4.5. *For any linear code $\mathscr{C} \subset \mathbb{F}^n$ with trivial hull, the following properties then hold*

1. $\boldsymbol{\Sigma}_{\mathscr{C}}^T = \boldsymbol{\Sigma}_{\mathscr{C}}$, $\boldsymbol{\Sigma}_{\mathscr{C}}^2 = \boldsymbol{\Sigma}_{\mathscr{C}}$ *and* $\boldsymbol{\Sigma}_{\mathscr{C}^\perp} = \boldsymbol{I}_n - \boldsymbol{\Sigma}_{\mathscr{C}}$
2. $\boldsymbol{H}_{\mathscr{C}}\boldsymbol{\Sigma}_{\mathscr{C}} = \boldsymbol{\Sigma}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T = \boldsymbol{0}$ *(the rows and columns of $\boldsymbol{\Sigma}_{\mathscr{C}}$ belong to $\mathscr{C}$)*
3. $\mathtt{rank}(\boldsymbol{\Sigma}_{\mathscr{C}}) = k$

The polynomial system $\mathfrak{S}$ contains some equations that involve only unknowns corresponding to one column (or one row) of $\boldsymbol{P}$, that is to say, equations of the form $\sum_{i=1}^n a_i x_{i,j} = b$ for some a_i and b in $\mathbb{F}$, and j in $[1, n]$. The next lemma explains that we can deduce directly the value of many variables from this kind of equations.

Lemma 3.4.6. *Assume that $\mathfrak{S}$ contains for some j in $[1, n]$ a linear equation $\sum_{i=1}^n a_i x_{i,j} = b$ where a_i and b are in $\mathbb{F}$. If we set $J \stackrel{def}{=} \{i : a_i \neq b\}$ then any solution of $\mathfrak{S}$ satisfies the relations*

$$\forall i \in J,\ x_{i,j} = 0.$$

Proof. We have for all i in $[1, n]$ the following equalities

$$b\,x_{i,j} = \left(\sum_{i'=1}^{n} a_{i'} x_{i',j}\right) x_{i,j} = a_i\,x_{i,j} \mod \mathfrak{Q}.$$

This implies that $x_{i,j} = 0$ for any $i \in J$. ■

Remark 3.4.7. If J is empty, Lemma 3.4.6 does not give new equations. This happens in particular with equation $\sum_{i=1}^{n} x_{i,j} = 1$ which is always in $\mathfrak{E}$ by construction.

Definition 3.4.8. An equation $\sum_{i=1}^{n} a_i x_{i,j} = b$ is called a block equation in column j. Similarly, we define a block equation in row i as an equation of the form $\sum_{j=1}^{n} c_j x_{i,j} = d$

Corollary 3.4.9. *Assume there exist $\boldsymbol{a}$ and $\boldsymbol{b}$ in $\mathbb{F}^n$ such that $\boldsymbol{b} = \boldsymbol{a}\boldsymbol{P}$. Then solving for all j in $[1, n]$ the linear equation $\sum_{i=1}^{n} a_i x_{i,j} = b_j$ implies $x_{i,j} = 0$ for all i and j such that $b_j \neq a_i$.*

We can obtain a similar result for row block equations by considering the equation $\boldsymbol{a} = \boldsymbol{b}\boldsymbol{P}^T$ but in reality the equality gives exactly the same set of constraints as $\boldsymbol{b} = \boldsymbol{a}\boldsymbol{P}$. This corollary permits to simplify the solving of the problem, in particular, if the coordinates in $\boldsymbol{b}$ are all distinct. In that situation, we recover the entire permutation by identifying for each i in $[1, n]$ the only integer j in $[1, n]$ such that $b_j = a_i$, which means that $j = \boldsymbol{P}(i)$.

A direct method for producing a block equation in column j consists in reducing the matrix of the linear system in row echelon form, with an ordering on the columns eliminating the variables $x_{i,l}$ with $l \neq j$, and keeping only the rows such that the columns corresponding to the variable $x_{i,l}$ are zero if $l \neq j$. This elimination method can be applied either on the matrix of the linear system $\mathfrak{E}$ or on a degree-D Macaulay matrix for any $D \geqslant 2$. But when applied on $\mathfrak{E}$ this gives a cost of $O(n^{2\omega})$ operations for one column j, and hence a total complexity of $O(n^{2\omega+1})$ operations after all the columns have been treated. Hence we have proved the following result.

Proposition 3.4.10. *Block equations can be generated from $\mathfrak{S}$ in $O(n^{2\omega+1})$ operations.*

Corollary 3.4.9 raises the question of producing efficiently such equations when they exist. An interesting candidate for producing them is $\mathbb{1}$ since it is unaffected when permuting its coordinates. Of course the linear system $\mathbb{1} = \mathbb{1}\boldsymbol{P}$ provides no information on $\boldsymbol{P}$. But if the codes have trivial hull, it is then possible to obtain easily and more efficiently block equations as explained by what follows. It consists in exploiting the property that when a code $\mathscr{C}$ has a trivial hull then $\mathbb{F}^n = \mathscr{C} \oplus \mathscr{C}^\perp$. In particular, we have $\mathbb{1} = \mathbb{1}_{\mathscr{C}} + \mathbb{1}_{\mathscr{C}^\perp}$ where $\mathbb{1}_{\mathscr{C}}$ belongs to $\mathscr{C}$ and $\mathbb{1}_{\mathscr{C}^\perp}$ lies in $\mathscr{C}^\perp$.

Proposition 3.4.11. *If there exists $\boldsymbol{P}$ in S_n such that $\mathscr{B} = \mathscr{A}\boldsymbol{P}$, and $\mathscr{A}$ and $\mathscr{B}$ have trivial hulls then $\mathbb{1}_{\mathscr{B}} = \mathbb{1}_{\mathscr{A}}\boldsymbol{P}$ and $\mathbb{1}_{\mathscr{B}^\perp} = \mathbb{1}_{\mathscr{A}^\perp}\boldsymbol{P}$.*

Proof. If $\mathcal{H}(\mathscr{A}) = \mathscr{A} \cap \mathscr{A}^\perp = \{\mathbf{0}\}$ then $\mathcal{H}(\mathscr{A})^\perp = \mathscr{A} + \mathscr{A}^\perp = \mathbb{F}^n$. Any permutation $\boldsymbol{P}$ that is solution sends a word from $\mathscr{A}$ to a word of $\mathscr{B}$. Hence, as $\mathbb{1}_{\mathscr{B}} + \mathbb{1}_{\mathscr{B}^\perp} = \mathbb{1} = \mathbb{1}\boldsymbol{P} = \mathbb{1}_{\mathscr{A}}\boldsymbol{P} + \mathbb{1}_{\mathscr{A}^\perp}\boldsymbol{P}$ and by the uniqueness of the decomposition over $\mathscr{B}^\perp$ and $\mathscr{B}$, we have therefore $\mathbb{1}_{\mathscr{B}} = \mathbb{1}_{\mathscr{A}}\boldsymbol{P}$ and $\mathbb{1}_{\mathscr{B}^\perp} = \mathbb{1}_{\mathscr{A}^\perp}\boldsymbol{P}$. ∎

We see from Proposition 3.4.4 that block equations can be obtained by essentially inverting the $n \times n$ matrix $\boldsymbol{H}_{\mathcal{H}(\mathscr{A})}$ and computing the products $\mathbb{1}\boldsymbol{\Sigma}_{\mathscr{A}}$ and $\mathbb{1}\boldsymbol{\Sigma}_{\mathscr{A}^\perp}$. So the complexity is $O(n^\omega)$ while the general method requires $O(n^{2\omega+1})$ operations as we have seen in Proposition 3.4.10. Surprisingly, for some range of parameters, we get so many new linear equations that the permutation equivalence can be solved simply by linear algebra without resorting to the computation of a Groebner basis.

Proposition 3.4.12. *When $\mathscr{A}$ and $\mathscr{B}$ are random linear codes the block equations $\mathbb{1}_{\mathscr{B}} = \mathbb{1}_{\mathscr{A}}\boldsymbol{P}$ add about $O\left(n^2\left(1 - \frac{1}{q}\right)\right)$ independent linear equations.*

Proof. From Proposition 3.4.11 and by applying Lemma 3.4.6 on $\mathbb{1}_{\mathscr{B}} = \mathbb{1}_{\mathscr{A}}\boldsymbol{P}$, we find a set $\mathcal{I} \subset$

$[1,n]^2$ such that $x_{i,j}=0$ for all $(i,j)\in\mathcal{I}$. Since the codes are random, we expect that the coefficients of $\mathbb{1}_{\mathscr{A}}$ and $\mathbb{1}_{\mathscr{B}}$ are uniformly distributed, and therefore the size of $\mathcal{I}$ is about $O\left(n^2\left(1-\frac{1}{q}\right)\right)$. ■

It is possible to improve the approach by iterating the process of finding new block equations after adding to $\mathfrak{E}$ equations $x_{i,j}=0$ when they are found coming from $\mathbb{1}_{\mathscr{B}}=\mathbb{1}_{\mathscr{A}}\boldsymbol{P}$. Experiments show that new block equations can be found by repeating twice or 3 times this process, but here again we cannot solve the system by linear algebra for all instances. Indeed, for the sake of simplicity, let us assume that $\mathbb{1}_{\mathscr{B}}=\mathbb{1}_{\mathscr{A}}\boldsymbol{P}$ adds exactly $n^2\left(1-\frac{1}{q}\right)$ equations. We also consider two codes with trivial hull and trivial permutation group satisfying Assumption 3.2.1. This means in particular that the permutation equivalence between them has at most one solution. The values of the remaining $\frac{n^2}{q}$ unknowns are then derived from the linear system $\mathfrak{E}\cup\left\{x_{i,j}\ :\ (i,j)\in\mathcal{I}\right\}$ for some set $\mathcal{I}\subset[1,n]^2$ whose rank is expected to be equal to $2k(n-k)+n^2\left(1-\frac{1}{q}\right)$ with high probability. So the linear system recovers the solution when it exists if $n^2\leqslant 2k(n-k)+n^2\left(1-\frac{1}{q}\right)$ which is equivalent to $R(1-R)\geqslant\frac{1}{2q}$ where $R=k/n$ is the rate of the considered codes and q is the size of the field $\mathbb{F}$. Finally, the computing of $\mathbb{1}_{\mathscr{A}}$ and $\mathbb{1}_{\mathscr{B}}$ requires time $O(n^\omega)$. The linear system for the remaining $\frac{n^2}{q}$ variables can be solved in $O\left(\left(\frac{n^2}{q}\right)^\omega\right)$ operations in $\mathbb{F}$. This discussion hence gives the following corollary. Our experimental results confirm the bound given here.

Corollary 3.4.13 (Informal)**.** *The permutation equivalence between two codes with trivial hull and trivial permutation group satisfying Assumption 3.2.1 can be decided and solved in $O\left(\left(\frac{n^2}{q}\right)^\omega+n^\omega\right)$ operations in $\mathbb{F}$ when the rate $R=\frac{k}{n}$ and the size q of the field $\mathbb{F}$ satisfy*

$$R(1-R)\geqslant\frac{1}{2q}. \tag{3.15}$$

We see that as the field $\mathbb{F}_q$ gets larger, the range of solvable parameters gets wider. In particular when $q=\Theta(n)$ the equivalence between two random linear codes can be decided in time $O\left(n^\omega\right)$. However, when the size q is constant, which is the most frequent encountered case, the complexity become $O\left(n^{2\omega}\right)$.

Actually the permutation equivalence between two codes with trivial hull can be greatly improved by observing that Proposition 3.4.11 can be generalized. The idea is to consider all the vectors $\left\{\boldsymbol{e}_j\ :\ j\in[1,n]\right\}$ instead of just $\mathbb{1}_n$. Indeed, if we assume again that $\mathscr{C}$ has a trivial hull then there is a unique $\boldsymbol{u}_j$ in $\mathscr{C}$ and $\boldsymbol{u}_j^\perp$ in $\mathscr{C}^\perp$ such that $\boldsymbol{e}_j=\boldsymbol{u}_j+\boldsymbol{u}_j^\perp$. We see from Proposition 3.4.4 that $\boldsymbol{u}_j$ is exactly the j-th row of the $n\times n$ matrix $\boldsymbol{\Sigma}_{\mathscr{C}}=\boldsymbol{G}_{\mathscr{C}}^T\left(\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T\right)^{-1}\boldsymbol{G}_{\mathscr{C}}$ and $\boldsymbol{u}_j^\perp$ is the j-th row of $\boldsymbol{\Sigma}_{\mathscr{C}^\perp}=\boldsymbol{H}_{\mathscr{C}}^T\left(\boldsymbol{H}_{\mathscr{C}}\boldsymbol{H}_{\mathscr{C}}^T\right)^{-1}\boldsymbol{H}_{\mathscr{C}}$.

Consequently if $\mathscr{B}=\mathscr{A}\boldsymbol{P}$ then by using the fact that $\boldsymbol{e}_j\boldsymbol{P}=\boldsymbol{e}_{\boldsymbol{P}^{-1}(j)}$ we have $\boldsymbol{\Sigma}_{\mathscr{B}}=\boldsymbol{P}^T\boldsymbol{\Sigma}_{\mathscr{A}}\boldsymbol{P}$ which is obviously equivalent to the following relation

$$\boldsymbol{P}\boldsymbol{\Sigma}_{\mathscr{B}}=\boldsymbol{\Sigma}_{\mathscr{A}}\boldsymbol{P}. \tag{3.16}$$

Proposition 3.4.14. *The linear systems $\mathfrak{L}$ and $\left\{\boldsymbol{\Sigma}_{\mathscr{A}}\boldsymbol{P}-\boldsymbol{P}\boldsymbol{\Sigma}_{\mathscr{B}}\right\}$ are equivalent.*

Proof. Firstly $\boldsymbol{\Sigma}_{\mathscr{A}}\boldsymbol{P} = \boldsymbol{P}\boldsymbol{\Sigma}_{\mathscr{B}}$ is equivalent to $\left(\boldsymbol{\Sigma}_{\mathscr{B}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{\Sigma}_{\mathscr{A}}\right)\overline{\boldsymbol{P}} = \boldsymbol{0}$, as $\boldsymbol{\Sigma}_{\mathscr{B}}$ is symmetric. From Proposition 3.4.5 the rows of $\boldsymbol{\Sigma}_{\mathscr{C}}$ belong to $\mathscr{C}$ for any code $\mathscr{C}$, and moreover $\texttt{rank}\ \boldsymbol{\Sigma}_{\mathscr{C}}$ is equal to the dimension of $\mathscr{C}$. There exist therefore two $k \times n$ matrices $\boldsymbol{T}_{\mathscr{B}}$ and $\boldsymbol{T}_{\mathscr{A}}$ such that $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{T}_{\mathscr{B}}\boldsymbol{\Sigma}_{\mathscr{B}}$ and $\boldsymbol{G}_{\mathscr{A}} = \boldsymbol{T}_{\mathscr{A}}\boldsymbol{\Sigma}_{\mathscr{A}}$. We define then $\boldsymbol{K}$ as the matrix

$$\boldsymbol{K} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{T}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \\ -\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{T}_{\mathscr{A}} \end{bmatrix}.$$

Using the properties of the Kronecker product (see Lemma 3.2.15), we then have

$$\boldsymbol{K}\left(\boldsymbol{\Sigma}_{\mathscr{B}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{\Sigma}_{\mathscr{A}}\right) = \begin{bmatrix} \boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \\ \boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}} \end{bmatrix}. \tag{3.17}$$

Consequently, $\left\{\boldsymbol{\Sigma}_{\mathscr{A}}\boldsymbol{P} - \boldsymbol{P}\boldsymbol{\Sigma}_{\mathscr{B}}\right\}$ is linearly dependent with $\mathfrak{E}$. Reciprocally, we know from (3.13) and Proposition 3.4.5 that by setting $\boldsymbol{D}_{\mathscr{C}} \stackrel{\text{def}}{=} \boldsymbol{G}_{\mathscr{C}}^T\left(\boldsymbol{G}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}^T\right)^{-1}$ for any linear code $\mathscr{C}$ with trivial hull, the following equalities hold

$$\begin{aligned} (\boldsymbol{D}_{\mathscr{B}} \otimes \boldsymbol{D}_{\mathscr{A}^\perp}, -\boldsymbol{D}_{\mathscr{B}^\perp} \otimes \boldsymbol{D}_{\mathscr{A}}) \begin{bmatrix} \boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \\ \boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}} \end{bmatrix} &= \boldsymbol{\Sigma}_{\mathscr{B}} \otimes \boldsymbol{\Sigma}_{\mathscr{A}^\perp} - \boldsymbol{\Sigma}_{\mathscr{B}^\perp} \otimes \boldsymbol{\Sigma}_{\mathscr{A}} \\ &= \boldsymbol{\Sigma}_{\mathscr{B}} \otimes (\boldsymbol{I}_n - \boldsymbol{\Sigma}_{\mathscr{A}}) \\ &\quad - (\boldsymbol{I}_n - \boldsymbol{\Sigma}_{\mathscr{B}}) \otimes \boldsymbol{\Sigma}_{\mathscr{A}} \\ &= \boldsymbol{\Sigma}_{\mathscr{B}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{\Sigma}_{\mathscr{A}}. \end{aligned}$$

This last equality with (3.17) proves that both systems are equivalent. ■

Table 3.7 shows experiments for codes with trivial hull over $\mathbb{F}_{2^2}$ where we obtain the maximal rank of the linear system by block linearization.

3.5 Codes over Non Prime Fields

We assume from now that the field $\mathbb{F}$ is equal to $\mathbb{F}_{p^m}$ where $m > 1$ and p is a prime number. We recall that the Frobenius map $\zeta_p : \mathbb{F}_q \to \mathbb{F}_q$ is defined for any $x \in \mathbb{F}_q$ as $\zeta_p(x) = x^p$. The composition of ζ_p with itself u times is denoted by ζ_{p^u}.

We have seen that the solutions are binary permutation matrices but $\mathfrak{L}$ is built upon the field $\mathbb{F}_{p^m}$. A classical way to take into consideration this property is to use the Frobenius action ζ_{p^u} since for any linear equation $\sum_{i,j} a_{i,j}x_{i,j} = b$ we also have

$$b^{p^u} = \left(\sum_{i,j} a_{i,j}x_{i,j}\right)^{p^u} = \sum_{i,j} a_{i,j}^{p^u}\, x_{i,j}.$$

This adds new linear equations which enables to define new linear systems. For any u in $[1, m-1]$ we set

$$\mathfrak{L}^+ \stackrel{\text{def}}{=} \bigcup_{u=0}^{m-1} \left\{ \zeta_{p^u}\left(\boldsymbol{H}_{\mathscr{B}}\right) \boldsymbol{P}\, \zeta_{p^u}\left(\boldsymbol{G}_{\mathscr{A}}^T\right)\ ,\ \zeta_{p^u}\left(\boldsymbol{G}_{\mathscr{B}}\right) \boldsymbol{P} \zeta_{p^u}\left(\boldsymbol{H}_{\mathscr{A}}^T\right)\ \right\}.$$

n	k	D_B	D_{B^i}	D_A	$\#Sol$	Time (s)	Memory (MB)
50	3	424	22	22	13,824	$2^{2.8}$	2^5
	12	18	1	1	1	$2^{2.6}$	2^5
	25	1	1	1	1	$2^{-1.3}$	2^5
100	5	1,663	8	8	128	$2^{6.6}$	$2^{7.2}$
	25	65	1	1	1	$2^{6.7}$	$2^{7.7}$
	50	1	1	1	1	$2^{2.3}$	$2^{7.5}$
200	5	8,255	15	15	6,144	$2^{11.8}$	$2^{10.9}$
	50	201	1	1	1	$2^{11.6}$	$2^{11.2}$
	100	1	1	1	1	$2^{8.5}$	$2^{11.0}$
300	10	16,953	1	1	1	$2^{14.1}$	$2^{13.3}$
	75	122	1	1	1	$2^{13.8}$	$2^{13.6}$
	150	1	1	1	1	$2^{11.6}$	$2^{13.3}$
500	250	1	1	1	1	$2^{14.8}$	$2^{16.2}$

Table 3.7: Block linearization for random codes with trivial hull over $\mathbb{F}_{2^2}$

- D_B is the dimension of the null space using block equations only.
- D_{B^i} is the dimension of the null space using iterated block linearization.
- D_A is the dimension of the linear space spanned by the actual solution set.
- $\#Sol$ is the cardinality of the solution set.

Proposition 3.5.1. *If $\mathscr{B} = \mathscr{A}\boldsymbol{P}$ where $\boldsymbol{P}$ belongs to S_n then $\boldsymbol{P}$ satisfies $\mathfrak{L}^+$. Furthermore,*

$$\texttt{rank}(\mathfrak{L}^+ \cup \mathfrak{P}) \leqslant m\left(2k(n-k) - h^2 - \delta\right) + 2n - 1 \tag{3.18}$$

with δ as defined in Theorem 3.2.25.

Hence, applying the Frobenius map adds new linear equations and in many cases we can obtain the "maximal" rank given by (3.18) in Proposition 3.5.1. In that particular case, and when the system has a unique solution or a small solution set, we can easily solve the equivalence problem. Experimentally we found that we can obtain a system of rank n^2 for random codes with trivial hull and unique solution in the range $1/m < R < 1 - 1/m$. The time complexity in this context is $O(n^{2\omega})$ operations over $\mathbb{F}_{p^m}$.

It is however to assess precisely the impact of the Frobenius action because there is no easy way to estimate the rank of $\mathfrak{L}^+$, besides the simple upper-bound mentioned in 3.18. In particular, $\mathfrak{L}^+ \cup \mathfrak{P}$ cannot be of full rank if $n^2 > m\Big(2k(n-k)\Big) + 2n$ which is equivalent to

$$2R(1-R) + \frac{2}{n} < \frac{1}{m}. \tag{3.19}$$

Experimentally applying Frobenius action with iterated block linearization when the extension degree is $\geqslant 2$ can handle the cases of codes with non-trivial hull but not selfdaul or weakly self-dual codes, see Table 3.8.

3.6 Codes with Non-Trivial Hulls (General Approach)

3.6.1 Preliminary observations

We turn now to the case of a code $\mathscr{C}$ of dimension k and length n with a non trivial hull. We set $h_{\mathscr{C}} \stackrel{\text{def}}{=} \dim \mathcal{H}(\mathscr{C}) > 0$. Without loss of generality, as $\mathscr{C} + \mathscr{C}^\perp = \mathcal{H}(\mathscr{C})^\perp$, we know that there exist two matrices $\boldsymbol{\Xi}_{\mathscr{C}}$, $\boldsymbol{\Xi}_{\mathscr{C}^\perp}$ from $\mathcal{M}_{n-h_{\mathscr{C}},n}(\mathbb{F})$ and $\boldsymbol{M}_{\mathscr{C}}$ from $\mathcal{M}_{n-h_{\mathscr{C}},h_{\mathscr{C}}}(\mathbb{F})$ such that $\boldsymbol{\Xi}_{\mathscr{C}} + \boldsymbol{\Xi}_{\mathscr{C}^\perp} = \boldsymbol{Z}_{\mathscr{C}}$ where

$$\boldsymbol{Z}_{\mathscr{C}} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{I}_{n-h_{\mathscr{C}}} & \boldsymbol{M}_{\mathscr{C}} \end{bmatrix}, \tag{3.20}$$

and such that the rows of $\boldsymbol{\Xi}_{\mathscr{C}}$ (*resp.* $\boldsymbol{\Xi}_{\mathscr{C}^\perp}$) belong to $\mathscr{C}$ (*resp.* $\mathscr{C}^\perp$). By using the same kind of argument as in the proof of Proposition 3.4.14, as there exist $\boldsymbol{D}_{\mathscr{C}}$ in $\mathcal{M}_{n-h_{\mathscr{C}},k}(\mathbb{F})$ and $\boldsymbol{D}_{\mathscr{C}^\perp}$ in $\mathcal{M}_{n-h_{\mathscr{C}},n-k}(\mathbb{F})$ such that $\boldsymbol{\Xi}_{\mathscr{C}} = \boldsymbol{D}_{\mathscr{C}}\boldsymbol{G}_{\mathscr{C}}$ and $\boldsymbol{\Xi}_{\mathscr{C}^\perp} = \boldsymbol{D}_{\mathscr{C}^\perp}\boldsymbol{H}_{\mathscr{C}}$ we have the following fact.

Proposition 3.6.1. *Any permutation $\boldsymbol{P}$ satisfying $\mathfrak{L}$ also verifies the following linear equations*

$$\boldsymbol{Z}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{\Xi}_{\mathscr{B}}^T = \boldsymbol{\Xi}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{Z}_{\mathscr{B}}^T. \tag{3.21}$$

Proof. We consider the following equations deduced from the linear system $\mathfrak{L}$

$$\begin{aligned}
(\boldsymbol{D}_{\mathscr{B}} \otimes \boldsymbol{D}_{\mathscr{A}^\perp}, -\boldsymbol{D}_{\mathscr{B}^\perp} \otimes \boldsymbol{D}_{\mathscr{A}}) \begin{bmatrix} \boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \\ \boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}} \end{bmatrix} &= \boldsymbol{\Xi}_{\mathscr{B}} \otimes \boldsymbol{\Xi}_{\mathscr{A}^\perp} - \boldsymbol{\Xi}_{\mathscr{B}^\perp} \otimes \boldsymbol{\Xi}_{\mathscr{A}} \\
&= \boldsymbol{\Xi}_{\mathscr{B}} \otimes (\boldsymbol{Z}_{\mathscr{A}} - \boldsymbol{\Xi}_{\mathscr{A}}) \\
&\quad - (\boldsymbol{Z}_{\mathscr{B}} - \boldsymbol{\Xi}_{\mathscr{B}}) \otimes \boldsymbol{\Xi}_{\mathscr{A}} \\
&= \boldsymbol{\Xi}_{\mathscr{B}} \otimes \boldsymbol{Z}_{\mathscr{A}} - \boldsymbol{Z}_{\mathscr{B}} \otimes \boldsymbol{\Xi}_{\mathscr{A}}.
\end{aligned}$$

n	k	h	D_F	D_{FB^i}	D_A
50	10	9	1,004	1	1
		10	1,044	122	1
	25	24	1,153	1	1
		25	1,250	1,250	1
100	20	19	4,204	1	1
		20	4,282	4,282	1
	50	49	4,803	1	1
		50	5,000	5,000	–
200	20	19	26,004	1	1
	100	99	19,603	1	1
300	40	39	51,004	1	1

Table 3.8: Frobenius Action and block linearization for random codes with non-trivial hull over $\mathbb{F}_{2^2}$

- D_F is the dimension of the null space using Frobenius action only.
- D_{FB^i} is the dimension of the null space using Frobenius then iterated block linearization.
- D_A is the dimension of the linear space spanned by the actual solution set.
- $\#Sol$ is the cardinality of the solution set.

We recall that $\left(\boldsymbol{\Xi}_{\mathscr{B}} \otimes \boldsymbol{Z}_{\mathscr{A}} - \boldsymbol{Z}_{\mathscr{B}} \otimes \boldsymbol{\Xi}_{\mathscr{A}}\right)\overline{\boldsymbol{P}} = \mathbf{0}$ is equivalent to (3.21), and this concludes the proof. ■

There is no easy way to solve (3.21) because the methods we developed in the previous section cannot be used directly. We propose in the following two approaches: the first one tries to find a good supplementary to the hull. The second approach is based on a very well-known problem in coding theory called the closest vector problem.

3.6.2 An approach based on the supplementary of the hull

We recall the definition of the shortened code.

Definition 3.6.2. Let $\mathscr{C}$ be a linear code of length n over $\mathbb{F}$ and consider a non empty set $\mathcal{I} \subset [1, n]$. The shortened code of $\mathscr{C}$ over $\mathcal{I}$ denoted by $\mathcal{S}_{\mathcal{I}}(\mathscr{C})$ is the linear code formed by vectors $\boldsymbol{u}$ from $\mathscr{C}$ such that we have the following

$$\forall i \in \mathcal{I}, \;\; u_i = 0. \tag{3.22}$$

Proposition 3.6.3. *Let $\mathscr{C}$ be a linear code of length n and let us set $h_{\mathscr{C}} \stackrel{def}{=} \dim \mathcal{H}(\mathscr{C})$. We consider a set $\mathcal{I} \subset [1, n]$ of cardinality $h_{\mathscr{C}}$ such that there exists a generator matrix $\boldsymbol{G}_{\mathcal{H}(\mathscr{C})}$ where the columns restricted to $\mathcal{I}$ form the identity matrix of order $h_{\mathscr{C}}$. Then the shortened code $\mathcal{S}_{\mathcal{I}}(\mathscr{C})$ has a trivial hull and is a supplementary of $\mathcal{H}(\mathscr{C})$ in $\mathscr{C}$.*

Proof. Firstly, we prove that $\mathcal{S}_{\mathcal{I}}(\mathscr{C}) \cap \mathcal{H}(\mathscr{C}) = \{\mathbf{0}\}$. Indeed, the vectors in $\mathcal{S}_{\mathcal{I}}(\mathscr{C})$ satisfy (3.22) while non-zero vectors in $\mathcal{H}(\mathscr{C})$ must have at least one i in $\mathcal{I}$ such that the coordinate at position i is different from zero. Next, it is easy to see that $\mathscr{C} = \mathcal{S}_{\mathcal{I}}(\mathscr{C}) + \mathcal{H}(\mathscr{C})$, hence $\mathscr{C} = \mathcal{S}_{\mathcal{I}}(\mathscr{C}) \oplus \mathcal{H}(\mathscr{C})$. In particular, if there is $\boldsymbol{v}$ in the hull of $\mathcal{S}_{\mathcal{I}}(\mathscr{C})$ then $\boldsymbol{v}$ belongs both to $\mathscr{C}$ and to the orthogonal of $\mathcal{S}_{\mathcal{I}}(\mathscr{C}) \oplus \mathcal{H}(\mathscr{C}) = \mathscr{C}$ which entails that $\boldsymbol{v}$ lies in $\mathcal{H}(\mathscr{C})$. But this is possible only if $\boldsymbol{v} = \mathbf{0}$ which proves that the hull of $\mathcal{S}_{\mathcal{I}}(\mathscr{C})$ is trivial. ■

Proposition 3.6.4. *Let $\mathscr{A}$ and $\mathscr{B}$ two codes such that there exists a permutation $\boldsymbol{P}$ satisfying $\mathscr{B} = \mathscr{A}\boldsymbol{P}$. Let $\mathcal{A} \subset [1, n]$ and let us define $\mathcal{B} \stackrel{def}{=} \{\boldsymbol{P}(a) : a \in \mathcal{A}\}$ then $\mathcal{S}_{\mathcal{B}}(\mathscr{B}) = \mathcal{S}_{\mathcal{A}}(\mathscr{A})\boldsymbol{P}$.*

We have therefore a direct approach that exploits Proposition 3.6.3 and 3.6.4 for deciding the equivalence between $\mathscr{A}$ and $\mathscr{B}$ by considering two sets $\mathcal{A} \subset [1, n]$ and $\mathcal{B} \subset [1, n]$ of cardinality $h = \dim \mathcal{H}(\mathscr{A}) = \dim \mathcal{H}(\mathscr{B})$ and testing the equivalence between the shortened codes $\mathcal{S}_{\mathcal{A}}(\mathscr{A})$ and $\mathcal{S}_{\mathcal{B}}(\mathscr{B})$. The probability that $\mathcal{S}_{\mathcal{A}}(\mathscr{A})$ and $\mathcal{S}_{\mathcal{B}}(\mathscr{B})$ are equivalent is at least $\frac{h!}{\binom{n}{h}}$.

3.6.3 An approach based on closest vectors

Definition 3.6.5. Let $\mathscr{C}$ be a linear code of length n over $\mathbb{F}$, and let $\boldsymbol{z}$ be a vector of $\mathbb{F}^n$. The closest vector to $\boldsymbol{z}$ in $\mathscr{C}$ is a codeword $\boldsymbol{u}_*$ in $\mathscr{C}$ such that the following holds:

$$\forall \boldsymbol{u} \in \mathscr{C}, \;\; wt(\boldsymbol{z} - \boldsymbol{u}_*) \leqslant wt(\boldsymbol{z} - \boldsymbol{u}).$$

Remark 3.6.6. It is not difficult to see that for any permutation σ the vector $\boldsymbol{u}_*^{\sigma}$ is a closest vector to $\boldsymbol{z}^{\sigma}$ in $\mathscr{C}^{\sigma}$.

An algorithm for finding the list $\mathcal{L}_{\mathscr{C}}(\boldsymbol{z})$ containing all solutions $\boldsymbol{u}_*$ consists in taking a parity check matrix $\boldsymbol{H}_{\mathscr{C}} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{I}_{n-k} & \boldsymbol{M} \end{bmatrix}$ in systematic form and computing $\boldsymbol{s} \stackrel{\text{def}}{=} \boldsymbol{H}_{\mathscr{C}}\boldsymbol{z}^T$. Next, for all $\boldsymbol{x} \in \mathbb{F}^k$, the algorithm computes $\boldsymbol{y} \stackrel{\text{def}}{=} \left(\boldsymbol{s} - \boldsymbol{M}\boldsymbol{x}^T\right)^T$ and keeps only couples $(\boldsymbol{y}, \boldsymbol{x})$ of minimum weight. The solutions are then $\boldsymbol{u}_* \stackrel{\text{def}}{=} \boldsymbol{z} - (\boldsymbol{y}, \boldsymbol{x})$. This algorithm recovers $\mathcal{L}_{\mathscr{C}}(\boldsymbol{z})$ with $O\big(k(n-k)q^k\big)$ operations.

Proposition 3.6.7. *For any two permutation equivalent codes $\mathscr{A}$ and $\mathscr{B} = \mathscr{A}\boldsymbol{P}$ where $\boldsymbol{P}$ lies in S_n, there exist $\xi_j^{\mathscr{A}}$ in $\mathcal{L}_{\mathscr{A}}(\boldsymbol{e}_j)$ and $\xi_j^{\mathscr{B}}$ in $\mathcal{L}_{\mathscr{B}}(\boldsymbol{e}_j)$ for each j in $[1, n]$ such that matrices $\boldsymbol{\Xi}_{\mathscr{A}} \stackrel{def}{=} \left[\xi_j^{\mathscr{A}}\right]$ and $\boldsymbol{\Xi}_{\mathscr{B}} \stackrel{def}{=} \left[\xi_j^{\mathscr{B}}\right]$ satisfy the equality*

$$\boldsymbol{P}\boldsymbol{\Xi}_{\mathscr{B}} = \boldsymbol{\Xi}_{\mathscr{A}}\boldsymbol{P}. \tag{3.23}$$

The time complexity for constructing $\boldsymbol{\Xi}_{\mathscr{A}}$ and $\boldsymbol{\Xi}_{\mathscr{B}}$ is $O\big(k(n-k)nq^k\big)$ operations.

Remark 3.6.8. We can also apply this algorithm on each $\boldsymbol{e}_j$ with either the linear code $\mathcal{H}(\mathscr{C})$ since for any permutation σ we have $\mathcal{H}(\mathscr{C}^{\sigma}) = \mathcal{H}(\mathscr{C})^{\sigma}$ from Proposition 1.3.2, or with $\mathscr{C}^{\perp}$ especially when $n - k < k$.

3.7 Finding a solution by Puncturing

In this section we study the effect of puncturing on the equivalent codes. We see, in some cases where the codes have trivial hull and trivial permutation group, that we can identify the permutation

by puncturing. We puncture the codes $\mathscr{A}$ and $\mathscr{B}$ in a chosen coordinates i and j, respectively, and we construct the linear system $\mathfrak{E}'$ for the punctured codes. Then we can identify if we have made a wrong guess for i and j by looking at the rank of $\mathfrak{E} \cup \mathfrak{E}'$ where with the correct guess we expect to get a smaller rank.

3.7.1 Method Description

The method processes the following steps in order to retrieve one permutation between two $[n, k]$ linear codes.

Input: Two generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$.
Output: A permutation fulfilling the equivalence.
Construct the linear system $\mathfrak{E}$.
$S \leftarrow \emptyset$.
for $i := 1$ **to** n **do**
  $\boldsymbol{G}'_{\mathscr{A}} \leftarrow \boldsymbol{G}_{\mathscr{A}}$ punctured in the coordinate i.
  for $j := 1$ **to** n **do**
    if j does not correspond to any previous i (check S) **then**
      $\boldsymbol{G}'_{\mathscr{B}} \leftarrow \boldsymbol{G}_{\mathscr{B}}$ punctured in the coordinate j.
      Construct the linear system $\mathfrak{E}'$ that corresponds to the punctured codes.
      if the "guess is correct" using $\texttt{rank}(\mathfrak{E} \cup \mathfrak{E}')$ **then**
        add (i, j) to S and go to the next i.
      end if
    end if
  end for
end for
return S.

Experimentally we found that we can distinguish between the correct and wrong guess by looking at the rank of the system $\mathfrak{E} \cup \mathfrak{E}'$ where the correct guess results in a smaller rank. The difference between ranks of correct and wrong guesses is 1. Thus the algorithm requires comparison of ranks that are computed in different steps. In the next section we analyze this method.

The algorithm does not always terminate successfully. We also present next when the algorithm succeeds and when it fails. Note that the algorithm tries to find only one solution and not all the solutions. It can be also used to find a small set of solutions.

3.7.2 Analysis of the Method

In order to see when the puncturing algorithm ends successfully and to show also the difference between ranks of correct and wrong guess is 1 we can look at the puncturing in a different way.

When we puncture columns i and j in $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$, respectively, this corresponds to the guess that the permutation $\boldsymbol{P}$ maps column i in $\boldsymbol{G}_{\mathscr{A}}$ to column j in $\boldsymbol{G}_{\mathscr{B}}$, since $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}$, thus the element $x_{i,j} = 1$ in $\boldsymbol{P}$. Setting the element $x_{i,j}$ to be 1 adds $2n - 1$ equations to the linear system. Thus we have the equations:

$$\mathfrak{G}_{i,j}: \quad \begin{cases} x_{i,j} = 1 \\ x_{i,k} = 0, & 1 \leqslant k \leqslant n, k \neq j \\ x_{k,j} = 0, & 1 \leqslant k \leqslant n, k \neq i. \end{cases}$$

Remark 3.7.1. Note that the equations of $\mathfrak{G}_{i,j}$ are linearly independent, thus the rank of $\mathfrak{G}_{i,j}$ is $2n-1$ and the equations $\sum_{k=1}^{n} x_{i,k} = 1$ and $\sum_{k=1}^{n} x_{k,j} = 1$ are linearly dependent on $\mathfrak{G}_{i,j}$.

Now instead of constructing the system $\mathfrak{E}'$ that corresponds to the column puncturing and considering $\mathfrak{E} \cup \mathfrak{E}'$ we can consider the system $\mathfrak{E}^* = \mathfrak{E} \cup \mathfrak{G}_{i,j}$. Let $\boldsymbol{E}^*$ be the matrix that corresponds to $\mathfrak{E}^*$. We can reorder the columns of $\boldsymbol{E}^*$ such that the columns that are related to the unknowns of $\mathfrak{G}_{i,j}$ appear at the right end of the matrix. Thus we have at least $2n-1$ equations in $2n-1$ unknowns by looking at the last block of $\boldsymbol{E}^*$. The following block matrix explains the structure of $\boldsymbol{E}^*$

$$\boldsymbol{E}^* = \left[\begin{array}{c|c|c} \multicolumn{3}{c}{\boldsymbol{E}} \\ \hline & 1 & 0 \\ & \ddots & \vdots \\ & 1 & 0 \\ 0 & 1 & 1 \\ & 1 & 0 \\ & \ddots & \vdots \\ & 1 & 0 \end{array}\right]$$

The way that we distinguish between the correct and wrong guess is by putting $\boldsymbol{E}^*$ in the row echelon form provided that with the wrong guess the system produces, by Gaussian elimination, one more equation that is independent of $\mathfrak{G}_{i,j}$ in the last block, thus the system has no solution. In the case of correct guess, even if the system produces an equation in the last block it will be linearly dependent on the existing equations.

Looking back to the case where we have block linear equation, in this case we know that the system always contains at least three equations in the block of the last $2n-1$ columns. Two equations are linearly dependent on $\mathfrak{G}_{i,j}$ according to Remark 3.7.1, namely $\sum_{k=1}^{n} x_{i,k} = 1$ and $\sum_{k=1}^{n} x_{k,j} = 1$. The status of the last equation, if dependent or not, enables to distinguish between the correct and wrong guess.

We denote by $\mathfrak{G}_{i,j}^*$ the subsystem of $\mathfrak{E}^*$ that contains only the unknowns of $\mathfrak{G}_{i,j}$. Then we have the following proposition:

Proposition 3.7.2. *We can identify our guess is wrong if the subsystem $\mathfrak{G}_{i,j}^*$ has no solution.*

Next we rewrite the puncturing algorithm in terms of the new description.

1: **Input**: Two generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$.
2: **Output**: A permutation fulfilling the equivalence.
3: Construct the system $\mathfrak{E}$ and its matrix $\boldsymbol{E}$.
4: $S \leftarrow \emptyset$.

5: **for** $i := 1$ **to** n **do**
6: **for** $j := 1$ **to** n **do**
7: **if** j does not correspond to any previous i (check S) **then**
8: Construct $\mathfrak{E}^* = \mathfrak{E} \cup \mathfrak{G}_{i,j}$ and let $\boldsymbol{E}^*$ be its matrix.
9: Bring $\boldsymbol{E}^*$ to the echelon form.
10: **if** the subsystem $\mathfrak{G}^*_{i,j}$ has no solution **then**
11: continue to the next j,
12: **else**
13: add (i, j) to S
14: continue to the next i.
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **return** S.

This algorithm tries to exclude wrong guesses and to find one solution. It terminates successfully when the hull is trivial, the system has a unique solution and $\mathbb{1}_n \notin \mathscr{C}$ and $\mathbb{1}_n \notin \mathscr{C}^\perp$ with $\mathscr{C} \in \{\mathscr{A}, \mathscr{B}\}$. If the system has many solutions and we want to find them we need to cancel the check in line 7 in the algorithm. Thus we have for each i many options of j and we have like a tree. If the tree of small size then we can validate all solutions against the system to find the correct ones.

The puncturing algorithm, to retrieve a single permutation, punctures the codes at most n^2 times and performs Gaussian elimination for each punctured codes. Thus the complexity is $O(n^2 \times n^{2\omega}) = O(n^{2(\omega+1)})$.

Problems of the Method

As we have seen from the description of the method it is clear that it has some shortages and in some cases does not terminate successfully, below we discuss these case.

- The first problem is that we need the system to produce at least on linearly independent equation of $\mathfrak{G}_{i,j}$ when we do the wrong guess, otherwise this guess is considered correct even if it is wrong.
- If the system has large solution set and we want to find all the solutions then the algorithm produces with each coordinate i a list of coordinates $j_1, \dots j_s$ that can be related to it by different permutations. Now we need to do additional work to find the correct permutations solutions. This might have high complexity depending on the size of the solution set.

3.8 ISD Approach

In this section we discuss how information set decoding (ISD) can be used to solve permutation equivalence.

3.8.1 ISD in Brief

Information Set Decoding (ISD) algorithms are the best known algorithms to decode linear codes without previous knowledge about the code structure. ISD algorithms are probabilistic where the algorithm succeed with some probability. The idea of ISD is to find a set of error-free coordinate positions in the encrypted message and the restriction of the code generator matrix to these coordinate positions is invertible. One can find the original message by multiplying the encrypted message by the inverse of the restricted matrix. ISD is also used to find a codeword of a specific weight w and this case is of interest for our equivalence problem.

There are many variants of ISD algorithms. They are introduced with the analysis of their complexity in Appendix C.

3.8.2 Solving PEP with ISD

Here we show how to adopt ISD to solve our equivalence problem PEP. Consider our linear system $\mathfrak{L}$ that is related to PEP, we can write $\mathfrak{L} : \boldsymbol{L}\overline{\boldsymbol{P}} = \mathbf{0}$ with

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}} \\ \boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}} \end{pmatrix}$$

The idea is that, we consider $\boldsymbol{L}$ as a parity check matrix of a linear code $\mathscr{L}$ of length n^2. Our target is to find a codeword of weight n in $\mathscr{L}$. The dimension of $\mathscr{L}$ is $n^2 - \mathtt{rank}(\boldsymbol{L})$. Since $\mathtt{rank}(\boldsymbol{L}) = 2k(n-k) - h^2$ as in Proposition 3.2.23 then the dimension of $\mathscr{L}$ is $k = n^2 - 2k(n-k) + h^2$.

Let R be the rate of the codes under the equivalence test and λ is the rate of their hull then the rate of the code $\mathscr{L}$ is $1 - 2R(1-R) + \lambda^2$. Looking at the code $\mathscr{L}$, our permutation solutions for PEP are codewords of weight n with a specific structure in $\mathscr{L}$. Thus we can use the ISD algorithms as presented in the previous section to find such codewords.

Complexity of PEP with ISD

Consider the bound of the asymptotic complexity of ISD algorithm introduced in [79], see Appendix C. Applying this bound to find the asymptotic complexity of solving PEP with ISD we have $w = n$ with $\lim_{n\to\infty} n/n^2 = 0$ and the rate of the code is $1 - 2R(1-R) + \lambda^2$. Thus we have the following proposition:

Proposition 3.8.1. *The work factor of solving PEP using ISD is about*

$$2^{\left(-n\log_2(2R(1-R)-\lambda^2)\right)\left(1+O(1)\right)}$$

3.9 Conclusion

This chapter was devoted to permutation equivalence problem. We introduced our algebraic model for the problem. We proved that the problem is well-described using our model. In an initial attempt to solve the system we used Groebner basis. Using Groebner basis we are able to solve the problem

but we get high complexity as the length of the code increases. We observed that when the hull is trivial we can add many linear equations before solving the system. This technique is called block linearization which utilizes block equations to find values of unknowns. When block equations exist for each block we can solve the problem efficiently. Frobenius action is used over extension fields to improve the linear part of the system. Experimentally, combining Frobenius action with block linearization enables to extend the range of solving for codes with non-trivial hull.

To handle the general case of no-trivial hull we introduced two approaches. The first approach is based on the supplementary of the hull which can be obtained by using the shortened codes. In this case we obtain codes with trivial hulls then we can apply block linearization to solve the problem. The probability that the resulting shortened codes are equivalent is at least $\frac{h!}{\binom{n}{h}}$. The second approach is based on the closest vector problem. We constructed n^2 linear equations in the n^2 unknowns of the permutation using the closest vectors to the standard vectors in the two codes. This method results in a complexity of $O\big(k(n-k)nq^k + n^{2\omega}\big)$.

In another method of solving permutation equivalence we focused on the linear part of the system. We used the ISD to find a codeword of weight n in a code of length n^2 which gave an exponential complexity. We also considered the punctured codes to find the permutation between the equivalent codes. In this case we were able to retrieve the permutation when we have a unique solution beside having block equation for each block.

Chapter 4

Diagonal Equivalence Problem (DEP)

In this chapter we introduce new approaches to solve DEP. We introduce algebraic model for the problem and use Groebner basis techniques to find solutions. Groebner basis computation for DEP seems to be much harder than PEP. We modify the algebraic model to get better performance with Groebner basis computation. We reduce DEP to PEP using the notion of the closure and use some of the techniques that we developed in the previous chapter to solve the problem. The main disadvantage of using the closure is that it expands the length of the code by a factor proportional to the underlying field size, thus we get extra variables in our model. We improve the model of the closure where we reduce the number of unknowns in the system by using the notion of *puncturing permutation*. This approach turns out to be very useful when combined with Frobenius action, block linearization and try and guess strategies over small fields, namely $\mathbb{F}_3$ and $\mathbb{F}_4$. As in PEP, information set decoding (ISD) is also used to solve DEP. Let us first recall the definition of DEP

Let $\mathscr{C}$ and $\mathscr{C}'$ be two $[n, k]$ linear codes over $\mathbb{F}_q$. We say $\mathscr{C}$ and $\mathscr{C}'$ are *diagonally equivalent* if there exist $(\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^*$ and a permutation $\pi \in S_n$ such that

$$\mathscr{C}' = \{(\alpha_1 c_{\pi(1)}, \alpha_2 c_{\pi(2)}, \ldots, \alpha_n c_{\pi(n)}) : (c_1, c_2, \ldots, c_n) \in \mathscr{C}\}.$$

In other words, the codewords of $\mathscr{C}'$ results from the codewords of $\mathscr{C}$ by permutation of coordinate entries and scaling each coordinate by a nonzero element from $\mathbb{F}_q$. Note that, when $\alpha_i = 1$ for $1 \leqslant i \leqslant n$ this is simply permutation equivalence.

In terms of generator matrices, let $\boldsymbol{G}$ and $\boldsymbol{G}'$ be the generator matrices of $\mathscr{C}$ and $\mathscr{C}'$, respectively. We say $\mathscr{C}$ and $\mathscr{C}'$ are *diagonally equivalent* if there exists a $k \times k$ invertible matrix $\boldsymbol{S}$, an $n \times n$ permutation matrix $\boldsymbol{P}$ and an invertible diagonal matrix $\boldsymbol{D}$ such that

$$\boldsymbol{G}' = \boldsymbol{SGPD}.$$

The matrix $\boldsymbol{PD}$ is called monomial matrix which is similar to permutation matrix but with exactly one nonzero element from the underlying field in each row and column. This element is not necessarily one as in the permutation case.

The *decisional* version of DEP is to decide if two given $[n, k]$ linear codes are diagonally equivalent or not where the *computational* version is to find the permutations and the elements $\alpha_i, 1 \leqslant i \leqslant n$ (the diagonal matrices) by which the two codes are equivalent.

4.1 Relation with Automorphism Group

In this section we see how the two problems of DEP and finding automorphism group of linear codes are closely related.

Proposition 4.1.1. *Let $\mathscr{A}$ be a linear code of length n, $\mathcal{A}_{\mathscr{A}}$ the automorphism group of $\mathscr{A}$ and σ a linear isomorphism of $\mathscr{A}$ that preserves the hamming distance. The set of linear isomorphisms of $\mathscr{A}$ that preserve the hamming distance γ such that $\mathscr{A} \xrightarrow{\gamma} \mathscr{A}^{\sigma}$ is exactly the set $\sigma\,\mathcal{A}_{\mathscr{A}} \stackrel{def}{=} \{\sigma\tau \ : \ \tau \in \mathcal{A}_{\mathscr{A}}\}$.*

Proof. Clearly any $\sigma\mathcal{A}_{\mathscr{A}}$ is a solution to the equivalence problem between $\mathscr{A}$ and $\mathscr{A}^{\sigma}$. Now let us assume that γ is such that $\mathscr{A} \xrightarrow{\gamma} \mathscr{A}^{\sigma}$. Since by assumption we also have $\mathscr{A} \xrightarrow{\sigma} \mathscr{A}^{\sigma}$, we deduce that $\sigma^{-1}\gamma$ belongs to $\mathcal{A}_{\mathscr{A}}$, which terminates the proof. ■

The next proposition shows that the permutation groups of the equivalent codes are isomorphic.

Proposition 4.1.2. *The automorphism groups of the diagonally equivalent codes are isomorphic.*

Proof. Let $\mathcal{A}_{\mathscr{A}}$ and $\mathcal{A}_{\mathscr{B}}$ be the automorphism groups of the two equivalent codes $\mathscr{B} = \mathscr{A}^{\sigma}$. Define the mapping $\phi : \mathcal{A}_{\mathscr{A}} \to \mathcal{A}_{\mathscr{B}}$ by $\phi(\pi) = \sigma\pi\sigma^{-1}$. It is easy to verify that ϕ is an isomorphism. ■

Proposition 4.1.2 shows that if we can find the automorphism groups of equivalent codes we can find the solution of the equivalence by finding the isomorphism between the two automorphism groups. The group isomorphism problem is known to be not harder than graph isomorphism problem [65]. We can look at the problem of finding the automorphism group of a code as a special case of the DEP where $\mathscr{A} = \mathscr{B}$.

4.2 Algebraic Model

In this section we introduce two algebraic models for DEP and give a proof that they describe the problem. We compare between these models and solve the problem using Groebner basis. We first proof some properties of diagonal equivalence.

Proposition 4.2.1. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent $[n, k]$ linear codes over a field $\mathbb{F}_q$ and let $\boldsymbol{D}$ be the diagonal matrix corresponding to the equivalence then $\boldsymbol{DZ}$ is another diagonal matrix that satisfies the equivalence, where $\boldsymbol{Z} = \alpha \boldsymbol{I}_n$ with $\alpha \in \mathbb{F}_q^*$.*

Proof. Let $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ be the generator matrices of $\mathscr{A}$ and $\mathscr{B}$ respectively, thus by diagonal equivalence $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$. Considering $\boldsymbol{Z}$ we have

$$\boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}\boldsymbol{Z} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}(\alpha\boldsymbol{I}_n) = (\alpha\boldsymbol{S})\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$$

where $(\alpha\boldsymbol{S})\boldsymbol{G}_{\mathscr{A}}$ is another basis for the code $\mathscr{A}$. Thus $\boldsymbol{DZ}$ also satisfies the equivalence between $\mathscr{A}$ and $\mathscr{B}$. ■

Corollary 4.2.2. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent $[n, k]$ linear codes over a field $\mathbb{F}_q$. The number of linear isometries between $\mathscr{A}$ and $\mathscr{B}$ that preserve the weight is multiple of $q - 1$.*

Proposition 4.2.3. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent $[n,k]$ linear codes and let $\boldsymbol{G}_{\mathscr{A}}, \boldsymbol{G}_{\mathscr{B}}, \boldsymbol{H}_{\mathscr{A}}$ and $\boldsymbol{H}_{\mathscr{B}}$ their generator and parity check matrices respectively. Then*

$$\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$$

Proof. Since $\mathscr{A}$ and $\mathscr{B}$ are equivalent then there exist an invertible matrix $\boldsymbol{S}$, a permutation matrix $\boldsymbol{P}$ and an invertible diagonal matrix $\boldsymbol{D}$ such that

$$\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}.$$

We know that $\boldsymbol{G}_{\mathscr{B}}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$ thus we have

$$\boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0} \Rightarrow \boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$$

■

Corollary 4.2.4. *Let $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}$ and let $\overline{\boldsymbol{X}}$ be the column vector results by stacking the columns of $\boldsymbol{X}$ then*

$$(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} = \boldsymbol{0}$$

Proposition 4.2.5. *Two linear codes are diagonal equivalent by the monomial matrix $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}$ if and only if their dual codes are diagonal equivalent by the monomial matrix $\boldsymbol{X}' = \boldsymbol{P}\boldsymbol{D}^{-1}$.*

Proof. Let $\mathscr{C}$ be a linear code, $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}$ an $n \times n$ monomial matrix, and $(\mathscr{C})\boldsymbol{X}$ a diagonal equivalent linear code of $\mathscr{C}$, we show that $((\mathscr{C})\boldsymbol{X})^{\perp} = (\mathscr{C}^{\perp})\boldsymbol{X}'$.

For all $\boldsymbol{c} \in \mathscr{C}$, $\boldsymbol{x} \in ((\mathscr{C})\boldsymbol{X})^{\perp} \Leftrightarrow \langle \boldsymbol{x}, \boldsymbol{c}\boldsymbol{P}\boldsymbol{D}\rangle = \boldsymbol{0} \Leftrightarrow \boldsymbol{x}(\boldsymbol{c}\boldsymbol{P}\boldsymbol{D})^{T} = \boldsymbol{0} \Leftrightarrow \boldsymbol{x}\boldsymbol{D}\boldsymbol{P}^{T}\boldsymbol{c}^{T} = \boldsymbol{0} \Leftrightarrow (\boldsymbol{x}\boldsymbol{D}\boldsymbol{P}^{T})\boldsymbol{c}^{T} = \boldsymbol{0} \Leftrightarrow \left\langle \boldsymbol{x}\boldsymbol{D}\boldsymbol{P}^{T}, \boldsymbol{c}\right\rangle = \boldsymbol{0} \Leftrightarrow \boldsymbol{x}\boldsymbol{D}\boldsymbol{P}^{T} \in \mathscr{C}^{\perp} \Leftrightarrow \boldsymbol{x} \in (\mathscr{C}^{\perp})(\boldsymbol{D}\boldsymbol{P}^{T})^{-1} \Leftrightarrow \boldsymbol{x} \in (\mathscr{C}^{\perp})\boldsymbol{P}\boldsymbol{D}^{-1}$, since $(\boldsymbol{P}^{T})^{-1} = \boldsymbol{P}$.

The other direction is obvious since the dual of the dual is the code itself. ■

If $\boldsymbol{H}_{\mathscr{B}}$ and $\boldsymbol{H}_{\mathscr{A}}$ are two parity check matrices of two diagonal equivalent codes then one can write $\boldsymbol{H}_{\mathscr{B}} = \boldsymbol{T}\boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}^{-1}$ where $\boldsymbol{T}$ is an invertible matrix.

In a similar way to Proposition 4.2.3 and Corollary 4.2.4 one can show the following

Corollary 4.2.6.

$$\boldsymbol{H}_{\mathscr{A}}\boldsymbol{X}'\boldsymbol{G}_{\mathscr{B}}^{T} = \boldsymbol{0} \quad \textit{and} \quad (\boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}})\overline{\boldsymbol{X}'} = \boldsymbol{0}$$

where $\overline{\boldsymbol{X}'}$ is the column vector results by stacking the columns of $\boldsymbol{X}' = \boldsymbol{P}\boldsymbol{D}^{-1}$.

We can state a proposition similar to the one in the permutation case as follows

Proposition 4.2.7. *Let $\mathscr{A}$ and $\mathscr{B}$ be two $[n,k]$ linear codes with generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ and parity check matrices $\boldsymbol{H}_{\mathscr{A}}$ and $\boldsymbol{H}_{\mathscr{B}}$ respectively, then the following statements are equivalent*

- *$\mathscr{A}$ and $\mathscr{B}$ are diagonal equivalent.*
- *There exists an $n \times n$ monomial matrix $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}$ such that $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{X}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$*
- *There exists an $n \times n$ monomial matrix $\boldsymbol{X}' = \boldsymbol{P}\boldsymbol{D}^{-1}$ such that $\boldsymbol{H}_{\mathscr{A}}\boldsymbol{X}'\boldsymbol{G}_{\mathscr{B}}^{T} = \boldsymbol{0}$*

Let α be a primitive element in the field $\mathbb{F}_q$, thus $\mathbb{F}_q = \langle\alpha\rangle$. The monomial matrix $\boldsymbol{X} = (x_{i,j})$ can be described by the following algebraic equations

$$x_{i,j}x_{i',j} = 0 \text{ for } i \neq i'$$
$$x_{i,j}x_{i,j'} = 0 \text{ for } j \neq j'$$

The previous equations does not prevent zero matrix and zero rows and columns thus we need the following equations

$$\prod_{k=1}^{q-1}\left(\sum_{i=1}^{n} x_{i,j} - \alpha^k\right) = 0, 1 \leqslant j \leqslant n$$
$$\prod_{k=1}^{q-1}\left(\sum_{j=1}^{n} x_{i,j} - \alpha^k\right) = 0, 1 \leqslant i \leqslant n$$

One can easily verify that a matrix $\boldsymbol{X} = (x_{i,j})$ is a monomial matrix if and only if it satisfies these set of equations. From one side any monomial matrix must verify these equations. From the other side the first two equations show that each row and each column in $\boldsymbol{X}$ contains at most one nonzero element. The last two equations show that it is not possible for all elements of row or column to be zero. Moreover if a row or a column contains more than one nonzero element this will violate the first two equations.

Below we present a minimized set of equations that describe the monomial matrix.

Proposition 4.2.8. *Let $\boldsymbol{X} = (x_{i,j})$ be $n \times n$ matrix then $\boldsymbol{X}$ is a monomial matrix if and only if it satisfies the following set of equations*

(1)

$$x_{i,j'}x_{i,j} = 0, \ \ 1 \leqslant j' < j \leqslant n, 1 \leqslant i \leqslant n$$

(2)

$$\prod_{k=1}^{q-1}\left(\sum_{i=1}^{n} x_{i,j} - \alpha^k\right) = 0, 1 \leqslant j \leqslant n$$

Proof. It is clear that if $\boldsymbol{X}$ is a monomial matrix it satisfies the equations in (1) and (2).

On the other hand, the first set of equations states that any row has at most one nonzero element thus $\boldsymbol{X}$ contains no more than n nonzero elements. The second set of equations states that $\boldsymbol{X}$ cannot have zero column.

Suppose $\boldsymbol{X}$ has zero row, there are two cases:

1. The matrix $\boldsymbol{X}$ contains less than n nonzero elements thus it has zero column, this contradicts the equations in (2).
2. The matrix $\boldsymbol{X}$ contains exactly n nonzero elements thus there is a row with more than one nonzero element. This violates the first set of equations.

Now assume that there is a column contains more than one nonzero elements this implies that there must be zero column since $\boldsymbol{X}$ contains at most n nonzero elements. ■

Corollary 4.2.9. *Let $\boldsymbol{X} = (x_{i,j})$ be $n \times n$ matrix then $\boldsymbol{X}$ is a monomial matrix if and only if it satisfies the following set of equations*

$$x_{i',j}x_{i,j} = 0, \;\; 1 \leqslant i' < i \leqslant n, 1 \leqslant j \leqslant n$$

$$\prod_{k=1}^{q-1} \left(\sum_{j=1}^{n} x_{i,j} - \alpha^k \right) = 0, \;\; 1 \leqslant i \leqslant n$$

We can distinguish between two algebraic models which differ according to the method that we use to express the monomial matrix $\boldsymbol{X}$.

4.2.1 First Model

The first model contains n^2 unknowns that correspond to $\boldsymbol{X} = (x_{i,j})$ and contains equations of maximum degree $q-1$. The model has the following equations

$$\mathfrak{D}_1 \; : \begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} & = & \boldsymbol{0} \\ x_{i,j'}x_{i,j} & = & 0, \;\; 1 \leqslant j' < j \leqslant n, 1 \leqslant i \leqslant n \\ \prod_{k=1}^{q-1} \left(\sum_{i=1}^{n} x_{i,j} - \alpha^k \right) & = & 0, \;\; 1 \leqslant j \leqslant n \end{cases}$$

Theorem 4.2.10. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent $[n,k]$ linear codes over a field $\mathbb{F}_q$. The matrix $\boldsymbol{X}$ fulfills the equivalence between $\mathscr{A}$ and $\mathscr{B}$ if and only if it is a solution for the system $\mathfrak{D}_1$.*

Proof. ($\Rightarrow$) Assume that $\boldsymbol{X}$ is a matrix that fulfills the equivalence between $\mathscr{A}$ and $\mathscr{B}$ thus it satisfies the set of monomial equations. From Corollary 4.2.4 and 4.2.6, $\boldsymbol{X}$ satisfies also the system of equations

$$(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} = \boldsymbol{0}$$

with $\boldsymbol{X} = \boldsymbol{PD}$.

($\Leftarrow$) To prove the other direction, assume that $\boldsymbol{X}$ is a solution for the system $\mathfrak{D}_1$ thus by Proposition 4.2.8, $\boldsymbol{X}$ is a monomial matrix. The matrix $\boldsymbol{X}$ satisfies the system

$$(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} = \boldsymbol{0}$$

Thus

$$\boldsymbol{G}_{\mathscr{A}}\boldsymbol{PD}\boldsymbol{H}_{\mathscr{B}}^T = \boldsymbol{0} \Rightarrow \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{PD}\boldsymbol{H}_{\mathscr{B}}^T = \boldsymbol{0}$$

for any $k \times k$ invertible matrix $\boldsymbol{S}$. Thus we can write $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{PD}$ hence, by the definition of diagonal equivalence, $\boldsymbol{X}$ fulfills the equivalence between $\mathscr{A}$ and $\mathscr{B}$. ■

Proposition 4.2.11. *The rank of the linear part of $\mathfrak{D}_1$ is $k(n-k)$.*

n	k	#Sol	Deg	Time (s)	Memory (MB)
6	3	8	4	$2^{5.059}$	2^5
8	4	8	4	$2^{1.737}$	2^5
10	5	2	3	2^2	2^6
12	6	2	3	$2^{5.858}$	$2^{7.539}$
14	7	2	3	$2^{9.984}$	$2^{9.885}$
16	8	2	3	$2^{13.643}$	$2^{12.580}$
18	9	2	3	$2^{16.188}$	$2^{14.752}$
20	10	2	3	$2^{18.987}$	$2^{16.9726}$

Table 4.1: Solving DEP using $\mathfrak{D}_1'$ with F_4 algorithm for random codes over $\mathbb{F}_3$

- #Sol: Number of actual solutions.
- Deg: Maximum degree reached during Groebner basis computation.

We add some extra linear equations to $\mathfrak{D}_1$ in order to see the effect in Groebner basis computation. Unfortunately in this case adding extra linear equations leads to extra variables in the system or extra equations of large degree.

Consider the system $\mathfrak{D}_1'$ as follows

$$\mathfrak{D}_1' \ : \begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} & = & \boldsymbol{0} \\ (\boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}})\overline{\boldsymbol{X'}} & = & \boldsymbol{0} \\ x_{i,j'}x_{i,j} & = & 0, \ 1 \leqslant j' < j \leqslant n, 1 \leqslant i \leqslant n \\ \prod_{k=1}^{q-1}\left(\sum_{i=1}^{n} x_{i,j} - \alpha^k\right) & = & 0, \ 1 \leqslant j \leqslant n \\ x'_{i,j} & = & x_{i,j}^{q-2}, \ 1 \leqslant i,j \leqslant n \end{cases}$$

where $\boldsymbol{X'} = (x'_{i,j}) = \boldsymbol{P}\boldsymbol{D}^{-1}$ thus we have $x'_{i,j} = x_{i,j}^{q-2}, 1 \leqslant i,j \leqslant n$.

This system is constructed from $\mathfrak{D}_1$ by adding extra linear equations this leads to extra variables. Using Groebner basis to solve DEP is complex in general. By experiments we have not managed to finish the computation of Groebner basis for $q \geqslant 5$ for $n \geqslant 20$. The experiments are implemented in a server Intel Xeon with 2.30GHz processor and 128GB memory.

Table 4.1 uses the system $\mathfrak{D}_1'$ and considers the favorable case where the codes are random (the hull is zero or of small size) with rate 0.5 over $\mathbb{F}_3$ to get a linear system of a larger rank. Note that over $\mathbb{F}_3$ we have $\alpha^{-1} = \alpha$ for all $\alpha \in \mathbb{F}_3^*$ thus $\boldsymbol{X'} = \boldsymbol{X}$.

4.2.2 Second Model

This model contains n^2+n variables that correspond to the variables of permutation matrix $\boldsymbol{P} = (p_{i,j})$ plus the variables of diagonal matrix $\boldsymbol{D} = diag(\lambda_1, \ldots, \lambda_n)$ since $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}$. The maximum degree appears in these equation is $q-1$.

In this model instead of considering the monomial matrix as one matrix we take care of its structure as a permutation matrix multiplied by a diagonal matrix.

Next we recall Proposition 3.2.8 which gives a minimized set of equations that describe the permutation $\boldsymbol{P}$.

Proposition 4.2.12. *Let $\boldsymbol{P} = (p_{i,j})$ be $n \times n$ matrix then $\boldsymbol{P}$ is a permutation matrix if and only if it satisfies the set of equations:*

1. $p_{i,j}p_{i',j} = 0$ *such that* $i \neq i'$ *and,*
2. $\sum_{j'=1}^{n} p_{i,j'} = 1$ *with* $1 \leqslant i \leqslant n$

The diagonal matrix can be described by n variables $\lambda_1, \ldots, \lambda_n \in \mathbb{F}_q$ with $\lambda_i \neq 0$ for $1 \leqslant i \leqslant n$. Thus satisfy the following equation

$$\prod_{k=1}^{q-1} \left(\prod_{i=1}^{n} \lambda_i - \alpha^k \right) = 0$$

or in an equivalent way one can write

$$\prod_{k=1}^{q-1} \left(\lambda_i - \alpha^k \right) = 0, \ 1 \leqslant i \leqslant n$$

Proposition 4.2.13. *Let $\boldsymbol{X}$ be an $n \times n$ matrix with entries from $\mathbb{F}_q$, $\boldsymbol{X}$ is a monomial matrix if and only if it can be written as $\boldsymbol{X} = \boldsymbol{PD}$ with $\boldsymbol{P} = (p_{i,j})$ an $n \times n$ matrix and $\boldsymbol{D}$ is a diagonal matrix with $\lambda_1, \ldots, \lambda_n$ in the diagonal with $\boldsymbol{P}$ and $\boldsymbol{D}$ satisfy the following system of equations*

1. $p_{i,j}p_{i',j} = 0$ *such that* $i \neq i'$,
2. $\sum_{j'=1}^{n} p_{i,j'} = 1$ *with* $1 \leqslant i \leqslant n$,
3. $\prod_{k=1}^{q-1} \left(\lambda_i - \alpha^k \right) = 0, \ 1 \leqslant i \leqslant n$.

$$\mathfrak{D}_2 \ : \begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{PD}} &= \boldsymbol{0} \\ \\ \sum_{j'=1}^{n} p_{i,j'} &= 1, \ 1 \leqslant i \leqslant n \\ p_{i,j}p_{i',j} &= 0, \ i \neq i' \\ \\ \prod_{k=1}^{q-1} \left(\lambda_i - \alpha^k \right) &= 0, \ 1 \leqslant i \leqslant n \end{cases}$$

The next theorem shows that the system $\mathfrak{D}_2$ is quite sufficient to describe DEP.

Theorem 4.2.14. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent $[n, k]$ linear codes over a field $\mathbb{F}_q$. Let $\boldsymbol{X} = \boldsymbol{PD}$ with $\boldsymbol{P} = (p_{i,j})$ is an $n \times n$ permutation matrix and $\boldsymbol{D} = diag(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix. The matrices $\boldsymbol{P}$ and $\boldsymbol{D}$ fulfill the equivalence between $\mathscr{A}$ and $\mathscr{B}$ if and only if they are solution for the system $\mathfrak{D}_2$.*

Proof. We prove only ($\Leftarrow$). Assume that $\boldsymbol{P}$ and $\boldsymbol{D}$ satisfy $\mathfrak{D}_2$. It is clear, from Proposition 4.2.13, that $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{D}$ is a diagonal matrix thus $\boldsymbol{X}$ is a monomial matrix. Since $\boldsymbol{X}$ satisfies the system

$$(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} = \boldsymbol{0}$$

thus $\boldsymbol{X}$ satisfies the equivalence between $\mathscr{A}$ and $\mathscr{B}$, see Theorem 4.2.10. ■

Proposition 4.2.15. *The rank of the linear part of $\mathfrak{D}_2$ is $\leqslant k(n-k) + n$.*

n	k	#Sol	Deg	Time (s)	Memory (MB)
6	3	48	4	$2^{0.578}$	2^5
8	4	16	3	$2^{0.926}$	2^5
10	5	2	3	$2^{4.225}$	$2^{7.170}$
12	6	2	3	$2^{5.833}$	$2^{8.006}$
14	7	4	3	$2^{11.369}$	$2^{11.420}$
16	8	2	3	$2^{11.978}$	$2^{12.001}$
18	9	2	3	$2^{11.956}$	$2^{12.682}$
20	10	2	3	$2^{16.801}$	$2^{14.702}$

Table 4.2: Solving DEP using $\mathfrak{D}'_2$ with F_4 algorithm for random codes over $\mathbb{F}_3$

- #Sol: Number of actual solutions.
- Deg: Maximum degree reached during Groebner basis computation.

Considering the system $\mathfrak{D}_2$. It is also complex to solve DEP with this model as explained in Table 4.2 where the favorable parameters are used in the tests. Note that this model does not contain linear equations except the equations that correspond to the permutation.

We can slightly modify $\mathfrak{D}_2$ and consider the modified system $\mathfrak{D}'_2$

$$\mathfrak{D}'_2 : \begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{PD}} &= \mathbf{0} \\ (\boldsymbol{G}_{\mathscr{B}} \otimes \boldsymbol{H}_{\mathscr{A}})\overline{\boldsymbol{PD'}} &= \mathbf{0} \\ \\ \sum_{j'=1}^{n} p_{i,j'} &= 1, \ 1 \leqslant i \leqslant n \\ p_{i,j}p_{i',j} &= 0, \ i \neq i' \\ \\ \prod_{k=1}^{q-1} (\lambda_i - \alpha^k) &= 0, \ 1 \leqslant i \leqslant n \\ \lambda'_i &= \lambda_i^{q-2}, \ 1 \leqslant i \leqslant n \end{cases}$$

This system contains more equations but this also increases the number of variables where we have $n^2 + 2n$ variables. We know that $\boldsymbol{D'} = diag(\lambda'_1, \ldots, \lambda'_n) = \boldsymbol{D}^{-1}$ thus we have the equations $\lambda'_i = \lambda_i^{q-2}, \ 1 \leqslant i \leqslant n$.

The favorable case for the system $\mathfrak{D}'_2$ is to be used for random codes over $\mathbb{F}_3$ where we have $\alpha^{-1} = \alpha$ for all $\alpha \in \mathbb{F}_3^*$ and we do not need to define extra variable. It is also useful in terms of algorithmic aspect to add all linear equations of the permutation. Table 4.2 shows experimental results in this case. The experiments show that using $\mathfrak{D}'_2$ is less complex than $\mathfrak{D}'_1$ as parameters are getting larger.

4.3 DEP and ISD

In this section we describe how to use the ISD approach to solve DEP. We have described similar approach for PEP. Brief background about ISD can be found in Appendix C.

We consider the linear system

$$(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})\overline{\boldsymbol{X}} = \mathbf{0}$$

We assume that the matrix $(\boldsymbol{H}_{\mathscr{B}} \otimes \boldsymbol{G}_{\mathscr{A}})$ is a parity check matrix for an $[n^2, K]$ linear code $\mathscr{L}$ with $K = n^2 - k(n-k)$. Our target is to find a codeword of weight n in $\mathscr{L}$. The ISD algorithm can be used to find such a codeword.

Proposition 4.3.1. *Let R be the rate of the original codes under the equivalence test then the rate of the code $\mathscr{L}$ is $1 - R(1-R)$.*

Looking at the code $\mathscr{L}$, our solutions of DEP are codewords of weight n (vector representation of a monomial matrix) and have a specific structure in $\mathscr{L}$. Thus we can use the ISD algorithms to find such codewords.

We recall the bound of ISD complexity introduced in [79].

Let n be length of a code $\mathscr{C}$, R is the code rate and ISD is looking for a codeword of weight w with $\lim_{n\to\infty} w/n = 0$ the work factor of ISD is: $2^{cw(1+O(1))}$ when n grows, and $c = -\log_2(1-R)$ is a constant.

Proposition 4.3.2. *The work factor of solving DEP using ISD is about*

$$2^{\left(-n\log_2(R(1-R))\right)\left(1+O(1)\right)}$$

PEP with ISD over $\mathbb{F}_3$

Over $\mathbb{F}_3$ as before we can use linear system of larger number of equations. In this case the bound of ISD is similar to the one introduced for PEP since the rank is $2k(n-k) - h^2$. Assuming that the rate of the hull is λ, thus we have work factor

$$2^{\left(-n\log_2(2R(1-R)-\lambda^2)\right)\left(1+O(1)\right)}$$

4.4 Reduction to PEP

In this section we study how to use the reduction of DEP to PEP that is introduced in Section 1.4. We use the techniques developed in the previous chapter to solve the reduced problem. The resulting code has larger length thus we need to consider larger number of unknowns while the dimension remains unchanged. Firstly we recall the definition of the closure of the code

Definition 4.4.1. Let $\mathscr{C}$ be $[n,k]$ linear code over $\mathbb{F}_q = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$, we define the closure $\widetilde{\mathscr{C}}$ of $\mathscr{C}$ to be the $[n(q-1), k]$ linear code:

$$\begin{aligned} \widetilde{\mathscr{C}} &= \{\boldsymbol{c} \otimes (\alpha, \ldots, \alpha^{q-1}) : \boldsymbol{c} \in \mathscr{C}\} \\ &= \mathscr{C} \otimes (\alpha, \ldots, \alpha^{q-1}) \end{aligned}$$

Assume that we want to solve DEP between two $[n,k]$ linear codes $\mathscr{A}$ and $\mathscr{B}$ over $\mathbb{F}_q$ with closures $\widetilde{\mathscr{A}}$ and $\widetilde{\mathscr{B}}$ respectively, the closures are $[N,k]$ linear codes with $N = n(q-1)$. Let $\widetilde{\boldsymbol{G}_{\mathscr{A}}}, \widetilde{\boldsymbol{G}_{\mathscr{B}}}$ and $\widetilde{\boldsymbol{H}_{\mathscr{A}}}, \widetilde{\boldsymbol{H}_{\mathscr{B}}}$ the generator and parity check matrices of $\widetilde{\mathscr{A}}$ and $\widetilde{\mathscr{B}}$ respectively. The algebraic model that describes the reduced problem is

$$\mathfrak{S}\ :\ \begin{cases} (\widetilde{\boldsymbol{H}_{\mathscr{B}}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}}) & \overline{\boldsymbol{P}} = \mathbf{0} \\ (\widetilde{\boldsymbol{G}_{\mathscr{B}}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}) & \overline{\boldsymbol{P}} = \mathbf{0} \\ (\mathbb{1}_N^T \otimes \boldsymbol{I}_N) & \overline{\boldsymbol{P}} = \mathbb{1}_N \\ (\boldsymbol{I}_N \otimes \mathbb{1}_N^T) & \overline{\boldsymbol{P}} = \mathbb{1}_N \\ x_{i',j}x_{i,j} & = 0, \quad 1 \leqslant i' < i \leqslant N, 1 \leqslant j \leqslant N \end{cases}$$

Here $\overline{\boldsymbol{P}}$ is an $N^2 \times 1$ matrix. Let $\widetilde{\boldsymbol{P}}$ be the permutation matrix that corresponds to $\overline{\boldsymbol{P}}$. One can notice that the equations in the system $\mathfrak{S}$ that correspond to $\widetilde{\boldsymbol{P}}$ describe the general structure of a permutation. Our permutation $\widetilde{\boldsymbol{P}}$ has more internal structure that can be used to improve the system and add a lot of equations.

The permutation $\widetilde{\boldsymbol{P}}$ is composed of $n \times n$ blocks of size $q-1 \times q-1$ each block is either $\mathbf{0}$ or a permutation. The following matrix gives an example with $n = 4$ for the structure of $\widetilde{\boldsymbol{P}}$ over $\mathbb{F}_q$.

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{P}_1 & \mathbf{0} \\ \boldsymbol{P}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{P}_3 \\ \mathbf{0} & \boldsymbol{P}_4 & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where each block is of size $(q-1) \times (q-1)$. Before we continue to introduce the equations we need to consider the group structure of a monomial matrix. Let $\mathcal{D}(n, \mathbb{F}_q)$ be the group of $n \times n$ invertible diagonal matrices over $\mathbb{F}_q$ and $\mathcal{S}_n$ the group of $n \times n$ permutation matrices. Note that

$$\mathcal{D}(n, \mathbb{F}_q) \equiv (\mathbb{F}_q^*)^n$$

We can look to the group of $n \times n$ monomial matrices as a semidirect product

$$\mathcal{D}(n, \mathbb{F}_q) \rtimes \mathcal{S}_n$$

This imposes a cyclic structure on $\widetilde{\boldsymbol{P}}$. That is if we assume $\boldsymbol{P}$ is a $(q-1) \times (q-1)$ permutation block in $\widetilde{\boldsymbol{P}}$ then $\boldsymbol{P}$ has the following structure

$$\boldsymbol{P} = \begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & \dots & 0 \\ & & \vdots & & & \ddots & \\ 0 & & \dots & & & 0 & 1 \\ 1 & 0 & & \dots & & & 0 \\ & \ddots & & & \dots & & \vdots \\ 0 & 0 & 1 & & & \dots & 0 \end{pmatrix}$$

Thus $\boldsymbol{P}_i$ is completely determined by a location of one 1.

Considering this structure of $\widetilde{\boldsymbol{P}}$ one can add large number of linear equations. Consider the block $B = (x_{e,f})_{1 \leqslant e,f \leqslant q-1}$ in $\widetilde{\boldsymbol{P}}$ then we have the following linear equations

$$x_{1,k} = x_{1+a,\ k+a-1 \bmod (q-1)\ +1},\ 1 \leqslant a \leqslant q-2,\ 1 \leqslant k \leqslant q-1$$

We can write the equations for each block in a similar way. All these equations are linearly independent since they use different set of variables. The number of equations per block is $(q-2)(q-1)$ thus for all blocks is $n^2(q-2)(q-1)$. Note that the number of variables in the system is $n^2(q-1)^2$.

The linear equations added by using the structure of $\widetilde{\boldsymbol{P}}$ contributes positively to Groebner basis computation but the complexity is higher than the permutation case for all $q \geqslant 3$. Note that over extension field, $q = p^m$ we can improve the system using $m-1$ Frobenius action.

Experimentally, using this approach with the methods that we have developed for PEP such as block linearization do not solve DEP. That is due to the small rate of the closure and the large number of variables which results in a linear system of small rank in addition to that the closure is weakly self-dual over $\mathbb{F}_q$ with $q \geqslant 4$.

4.5 New Modeling Based on the Closure

In this section we present a new algebraic model based on the closure of the code. This model is more efficient than the previous model since it involves less number of variables in the initial system. The matrix of variables $\widetilde{\boldsymbol{P}}$ is not a permutation, it has similar structure to the permutation but with zero rows allowed, we call it *puncturing permutation*.

Definition 4.5.1. Let n and r be positive integers and $\boldsymbol{P}$ be an $nr \times n$ binary matrix. $\boldsymbol{P}$ is called *puncturing permutation* if every column contains exactly one 1 and each block of r rows R_i, that contains the rows of indexes $(i-1)r+1 \leqslant j \leqslant ir$ for $1 \leqslant i \leqslant n$, contains exactly one 1.

Note that, when $r = 1$ the above definition coincides with the definition of the normal permutation.

Example 4.5.2. An example of a puncturing permutation with $n = 4$ and $r = 2$.

$$\widetilde{\boldsymbol{P}} = \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent codes with $\boldsymbol{G}_{\mathscr{A}}, \boldsymbol{G}_{\mathscr{B}}$ and $\boldsymbol{H}_{\mathscr{A}}, \boldsymbol{H}_{\mathscr{B}}$ their generator and parity check matrices thus we have

$$\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$$

In terms of closure of the code we have

$$\widetilde{\boldsymbol{G}_{\mathscr{B}}} = \widetilde{\boldsymbol{S}}\widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}$$

We can adopt a hybrid model between the original one and the model of the closure.

Proposition 4.5.3. *Let $\boldsymbol{G}_{\mathscr{A}}$ be a generator matrix of an $[n,k]$ linear code $\mathscr{A}$ over $\mathbb{F}_q = \langle\alpha\rangle$, $\boldsymbol{P}$ an $n \times n$ permutation matrix and $\boldsymbol{D}$ an $n \times n$ diagonal matrix such that $\boldsymbol{D} = diag(\alpha^{l_1}, \ldots, \alpha^{l_n})$. We can always find an $n(q-1) \times n$ puncturing permutation matrix $\widetilde{\boldsymbol{P}}$ such that*

$$\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D} = \widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}$$

Proof. Let $a_{i,j}$ be the element in row i and column j in $\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$, then we can write $a_{i,j} = \alpha^{l_j}\boldsymbol{g}_i\boldsymbol{P}_j$ where $\boldsymbol{g}_i$ is row i in $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{P}_j$ is column j in $\boldsymbol{P}$. To obtain the same element in $\widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}$ let $\widetilde{\boldsymbol{P}}_j$ be column j in $\widetilde{\boldsymbol{P}}$. The column $\widetilde{\boldsymbol{P}}_j$ can be seen as an n blocks of size $q-1$ where every block is zero except one block that contains one 1. The index of the nonzero block in $\widetilde{\boldsymbol{P}}_j$ corresponds to which column is scaled where the position of the 1 in the nonzero block corresponds to the scalar, in this case the index is l_j. Thus we write

$$\widetilde{\boldsymbol{P}}_j^T = \begin{pmatrix}\boldsymbol{0}_{q-1}^T & \ldots & \boldsymbol{0}_{q-1}^T & \boldsymbol{I}_{lj}^T & \boldsymbol{0}_{q-1}^T & \ldots & \boldsymbol{0}_{q-1}^T\end{pmatrix}$$

where $\boldsymbol{0}_{q-1}$ is zero vector of size $q-1$ and $\boldsymbol{I}_{lj}$ is the lj column in the $q-1 \times q-1$ identity matrix $\boldsymbol{I}$.

To prove that each block of rows contains exactly one 1, assume that there is no duplicate or zero columns in $\boldsymbol{G}_{\mathscr{A}}$ otherwise they can be removed. Let R be the block of $q-1$ rows of $\widetilde{\boldsymbol{P}}$ that contains $\boldsymbol{P}_j$, the j^{th} column from $\boldsymbol{P}$. This block contains exactly one 1 that corresponds to $\boldsymbol{P}_j$ all other elements in R are zeros, otherwise there is one column scaled by two different scalars, this contradicts that we do not have duplicate or zero column. ■

This shows that there is exactly n ones in $\widetilde{\boldsymbol{P}}$ these ones must spread in different columns otherwise $\widetilde{\boldsymbol{P}}$ contains zero column. Thus $\widetilde{\boldsymbol{P}}$ is a puncturing permutation matrix of $n(q-1)$ rows and n columns.

Consider the puncturing permutation $\widetilde{\boldsymbol{P}}$ in Example 4.5.2. We give the corresponding permutation $\boldsymbol{P}$ and diagonal matrix $\boldsymbol{D}$ over $\mathbb{F}_3$.

$$\boldsymbol{P} = \begin{pmatrix}0 & 1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix} \boldsymbol{D} = \begin{pmatrix}2 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 2 & 0\\ 0 & 0 & 0 & 2\end{pmatrix}$$

Note that, for any generator matrix

$$\boldsymbol{G} = \begin{pmatrix}\boldsymbol{G}_1 & \ldots & \boldsymbol{G}_n\end{pmatrix}$$

we have

$$\widetilde{\boldsymbol{G}}\widetilde{\boldsymbol{P}} = \boldsymbol{G}\boldsymbol{P}\boldsymbol{D}$$

4.5.1 Algebraic Model

In this section we develop algebraic model that describes DEP with the new approach. We deduce equations from the equivalence and equations that describe the puncturing permutation.

Proposition 4.5.4. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent codes with generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ respectively, such that $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$ for some permutation and diagonal matrices $\boldsymbol{P}$ and $\boldsymbol{D}$ respectively. Then we have*

$$\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}$$

where $\widetilde{\boldsymbol{G}_{\mathscr{A}}}$ is the generator matrix of the closure of $\mathscr{A}$ and $\widetilde{\boldsymbol{P}}$ is a puncturing permutation matrix.

Proposition 4.5.5. *Let $\mathscr{C}$ be a linear code with parity check matrix $\boldsymbol{H}$ and closure $\widetilde{\mathscr{C}}$. Let $\widetilde{\boldsymbol{H}}^{-1} = \boldsymbol{H} \otimes \begin{pmatrix}\alpha^{-1} & \dots & \alpha^{-(q-1)}\end{pmatrix}$. Then $\widetilde{\boldsymbol{H}}^{-1}$ generates a subcode of $\widetilde{\mathscr{C}}^{\perp}$.*

Proof. Let $\boldsymbol{G}$ be the generator matrix of $\mathscr{C}$ and

$$\widetilde{\boldsymbol{G}} = \boldsymbol{G} \otimes \begin{pmatrix}\alpha & \dots & \alpha^{(q-1)}\end{pmatrix}.$$

be the generator matrix of $\widetilde{\mathscr{C}}$ then it is easy to check that

$$\widetilde{\boldsymbol{H}}^{-1}\widetilde{\boldsymbol{G}}^{T} = \boldsymbol{0}$$

Let $\boldsymbol{x}$ be a row from $\boldsymbol{G}$ and $\boldsymbol{y}$ a row from $\boldsymbol{H}$ then we have $\boldsymbol{x}.\boldsymbol{y} = 0$. The corresponding product for the closure will give $(q-1)(\boldsymbol{x}.\boldsymbol{y}) = 0$. ■

Next we provide the first set of equations in our algebraic model

Theorem 4.5.6. *Let $\mathscr{A}$ and $\mathscr{B}$ be two diagonal equivalent codes with generator matrices $\boldsymbol{G}_{\mathscr{A}}$ and $\boldsymbol{G}_{\mathscr{B}}$ respectively, such that $\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}$ for some permutation and diagonal matrices $\boldsymbol{P}$ and $\boldsymbol{D}$ respectively. Let $\boldsymbol{H}_{\mathscr{A}}$ and $\boldsymbol{H}_{\mathscr{B}}$ be the parity check matrices for $\mathscr{A}$ and $\mathscr{B}$, respectively Then we have*

$$\begin{cases} (\boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}}) & \overline{\widetilde{\boldsymbol{P}}} = \boldsymbol{0} \\ (\boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}) & \overline{\widetilde{\boldsymbol{P}}} = \boldsymbol{0} \end{cases}$$

Where $\widetilde{\boldsymbol{G}_{\mathscr{A}}}$ is the generator matrix of the closure of $\mathscr{A}$, $\widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}$ as defined before and $\overline{\widetilde{\boldsymbol{P}}}$ is the column representation of the puncturing permutation $\widetilde{\boldsymbol{P}}$.

Proof. From Proposition 4.5.4 we have

$$\boldsymbol{G}_{\mathscr{B}} = \boldsymbol{S}\boldsymbol{G}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D} = \boldsymbol{S}\widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}$$

Since we know

$$\boldsymbol{G}_{\mathscr{B}}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$$

Thus we have

$$\widetilde{\boldsymbol{G}_{\mathscr{A}}}\widetilde{\boldsymbol{P}}\boldsymbol{H}_{\mathscr{B}}^{T} = \boldsymbol{0}$$

$$\Rightarrow (\boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}})\overline{\widetilde{\boldsymbol{P}}} = \boldsymbol{0}$$

To prove the second part we know

$$\boldsymbol{H}_{\mathscr{B}} = \boldsymbol{T}\boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}^{-1} \qquad \text{and}$$

$$\boldsymbol{H}_{\mathscr{A}}\boldsymbol{P}\boldsymbol{D}^{-1} = \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}\widetilde{\boldsymbol{P}}$$

Thus we have

$$\boldsymbol{H}_{\mathscr{B}}\boldsymbol{G}_{\mathscr{B}}^{T} = \mathbf{0}$$
$$\Rightarrow \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}\widetilde{\boldsymbol{P}}\boldsymbol{G}_{\mathscr{B}}^{T} = \mathbf{0}$$
$$\Rightarrow (\boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1})\overline{\widetilde{\boldsymbol{P}}} = \mathbf{0}$$

■

Let $\widetilde{\boldsymbol{P}} = (x_{i,j})$ be an $n(q-1) \times n$ puncturing permutation. Then $\widetilde{\boldsymbol{P}}$ is described by the following system of equations

$$\mathfrak{P} : \begin{cases} \sum_{i=1}^{n(q-1)} x_{i,j} & = & 1, & 1 \leqslant j \leqslant n \\ \sum_{j=1}^{n} \sum_{i=(l-1)(q-1)+1}^{l(q-1)} x_{i,j} & = & 1, & 1 \leqslant l \leqslant n \\ x_{i,j}^2 & = & x_{i,j}, & 1 \leqslant i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \\ x_{i',j}x_{i,j} & = & 0, & 1 \leqslant i' < i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \\ x_{i',j'}x_{i,j} & = & 0, & (l-1)(q-1)+1 \leqslant i', i \leqslant l(q-1), 1 \leqslant l \leqslant n, \\ & & & 1 \leqslant j', j \leqslant n, j' \neq j. \end{cases}$$

One can easily verify that if an $n(q-1) \times n$ matrix $\widetilde{\boldsymbol{P}}$ satisfies $\mathfrak{P}$ then it is a puncturing permutation with the required structure.

In fact we do not need all these equations to describe $\widetilde{\boldsymbol{P}}$. The following proposition presents a reduced set of equations.

Proposition 4.5.7. *Let $\widetilde{\boldsymbol{P}} = (x_{i,j})$ be an $n(q-1) \times n$ puncturing permutation. Then $\widetilde{\boldsymbol{P}}$ is described by the following system of equations*

$$\mathfrak{P}' : \begin{cases} \sum_{j=1}^{n} \sum_{i=(k-1)(q-1)+1}^{k(q-1)} x_{i,j} & = & 1, & 1 \leqslant k \leqslant n \\ x_{i',j}x_{i,j} & = & 0, & 1 \leqslant i' < i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \end{cases}$$

Proof. Assume that $\widetilde{\boldsymbol{P}}$ is an $n(q-1) \times n$ matrix that satisfies the algebraic system $\mathfrak{P}'$. The second set of equations states that each column contains at most one nonzero element, thus $\widetilde{\boldsymbol{P}}$ contains at most n nonzero elements. The first set of equations shows that the sum of each row block is 1. Assume that there is one row block contains more than one nonzero elements thus there exists one row block is $\mathbf{0}$ which contradicts the first set of equations. Thus there is exactly one nonzero element for each block which is equal to 1 and we have exactly n ones in $\widetilde{\boldsymbol{P}}$. The n ones are distributed in different columns otherwise the second set of equations is violated. ■

Now we are ready to present our algebraic system that models DEP. As we have done in the permutation case, in this model we add the maximum number of linear equations. This enhances the system when solving. We can also add the equations that restrict the solutions to binary elements.

$$\mathfrak{S}\ :\ \begin{cases} \left(\boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbf{0} \\ \left(\boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbf{0} \\ \sum_{i=1}^{n(q-1)} x_{i,j} & = 1, \quad 1 \leqslant j \leqslant n \\ \sum_{j=1}^{n} \sum_{i=(k-1)(q-1)+1}^{k(q-1)} x_{i,j} & = 1, \quad 1 \leqslant k \leqslant n \\ x_{i,j}^2 & = x_{i,j}, \quad 1 \leqslant i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \\ x_{i',j} x_{i,j} & = 0, \quad 1 \leqslant i' < i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \end{cases}$$

Proposition 4.5.8. *The set of equations*

$$\sum_{i=1}^{n(q-1)} x_{i,j} = 1,\ 1 \leqslant j \leqslant n$$

can be written in terms of Kronecker product

$$\left(\boldsymbol{I}_n \otimes \mathbb{1}_{n(q-1)}\right) \overline{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T$$

Proposition 4.5.9. *The set of equations*

$$\sum_{j=1}^{n} \sum_{i=(k-1)(q-1)+1}^{k(q-1)} x_{i,j} = 1,\ 1 \leqslant k \leqslant n$$

can be written in terms of Kronecker product

$$\left(\boldsymbol{M} \otimes \mathbb{1}_{(q-1)}\right) \overline{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T$$

where $\boldsymbol{M}$ is an $n \times n^2$ matrix with each row has exactly n ones and the other elements are zeros. The locations of the ones in row $r, 1 \leqslant r \leqslant n$ are $i \times n + r, 0 \leqslant i \leqslant n-1$.

Now we can rewrite the system $\mathfrak{S}$ in terms of Kronecker product using the previous propositions as follows

$$\mathfrak{S}\ :\ \begin{cases} \left(\boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbf{0} \\ \left(\boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbf{0} \\ \left(\boldsymbol{I}_n \otimes \mathbb{1}_{n(q-1)}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T \\ \left(\boldsymbol{M} \otimes \mathbb{1}_{(q-1)}\right) & \overline{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T \\ x_{i,j}^2 & = x_{i,j}, \quad 1 \leqslant i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \\ x_{i',j} x_{i,j} & = 0, \quad 1 \leqslant i' < i \leqslant n(q-1),\ 1 \leqslant j \leqslant n \end{cases}$$

Let $\boldsymbol{D} = diag(\lambda_1, \ldots, \lambda_n)$ be the diagonal matrix that corresponds to the equivalence. According to Proposition 4.2.1 one can fix one element of the diagonal to be one, say $\lambda_1 = 1$. The following proposition explains the effect of fixing $\lambda_1 = 1$ to $\widetilde{\boldsymbol{P}}$.

Proposition 4.5.10. *Let $\boldsymbol{D} = diag(\lambda_1, \ldots, \lambda_n)$ be the diagonal matrix that corresponds to the equivalence between two codes and let $\widetilde{\boldsymbol{P}} = (x_{i,j})$ as defined before. Fixing $\lambda_1 = 1$ corresponds to fixing the first column in $\widetilde{\boldsymbol{P}}$ to be zero except the positions $i(q-1)$ with $1 \leqslant i \leqslant n$.*

Proof. Fixing $\lambda_1 = 1$ shows that the first column in $\boldsymbol{G}_{\mathscr{A}}$ is left unscaled. All columns in $\widetilde{\boldsymbol{G}_{\mathscr{A}}}$ are scaled except the columns in the positions $i(q-1)$ with $1 \leqslant i \leqslant n$ thus the first column in $\widetilde{\boldsymbol{P}}$ has zeros except there is a 1 in one of these positions. The exact position of the 1 is not known since it corresponds to the permutation. ■

4.5.2 Properties of the Linear System

We discuss in this section the properties of the linear part of $\mathfrak{S}$, denoted $\mathfrak{L}$, where we give an upper bound of the rank of the system. This bound depends on the field, in $\mathbb{F}_3$ for example is affected by the hull where in $\mathbb{F}_q$ with $q \geqslant 4$ it does not. Before we proceed we recall Proposition 1.4.6.

Proposition. *Let $\widetilde{\mathscr{A}}$ be a closure of an $[n,k]$ linear code $\mathscr{A}$ over $\mathbb{F}_q$ with hull of dimension h and let $\widetilde{h}$ be the dimension of the hull of $\widetilde{\mathscr{A}}$ then*

1. *If $q = 3$ then $\widetilde{h} = h$.*
2. *If $q \geqslant 4$ then $\widetilde{h} = k$ thus $\widetilde{\mathscr{A}}$ is weakly self-dual code*

Proposition 4.5.11. *Let $\mathfrak{L}$ be the linear system corresponding to the diagonal equivalence between two $[n,k]$ linear codes $\mathscr{A}$ and $\mathscr{B}$ over $\mathbb{F}_q$ and let $\boldsymbol{L}$ its corresponding matrix, then*

- *if $q = 3$, then $\mathtt{rank}(\mathfrak{L}) \leqslant 2k(n-k) + 2n - 1 - h^2$ and*
- *if $q > 3$, then $\mathtt{rank}(\mathfrak{L}) \leqslant 2k(n-k) + 2n - 1$.*

where h is the dimension of the hull.

Proof. The linear part $\mathfrak{L}$ is

$$\mathfrak{L}: \begin{cases} \left(\boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}}\right) & \widetilde{\widetilde{\boldsymbol{P}}} = \boldsymbol{0} \\ \left(\boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}\right) & \widetilde{\widetilde{\boldsymbol{P}}} = \boldsymbol{0} \\ \left(\boldsymbol{I}_n \otimes \mathbb{1}_{n(q-1)}\right) & \widetilde{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T \\ \left(\boldsymbol{M} \otimes \mathbb{1}_{(q-1)}\right) & \widetilde{\widetilde{\boldsymbol{P}}} = \mathbb{1}_n^T \end{cases}$$

One can verify that the rank of

$$\begin{pmatrix} \boldsymbol{I}_n \otimes \mathbb{1}_{n(q-1)} & -\mathbb{1}_n^T \\ \boldsymbol{M} \otimes \mathbb{1}_{(q-1)} & -\mathbb{1}_n^T \end{pmatrix}$$

is $2n-1$.

The rank of

$$\begin{pmatrix} \boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}} \\ \boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1} \end{pmatrix}$$

is $\leqslant 2k(n-k)$.

Let $\boldsymbol{G}_{\mathcal{H}(\mathscr{B})}$ be the generator matrix of the hull of code $\mathscr{B}$, $\widetilde{\boldsymbol{G}}_{\mathcal{H}(\mathscr{A})}$ and $\widetilde{\boldsymbol{G}}^{-1}_{\mathcal{H}(\mathscr{A})}$ the generator matrices of the hull of codes $\widetilde{\mathscr{A}}$ and $\widetilde{\mathscr{A}^{-1}}$ (the code generated by $\widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1}$), respectively. Thus one can write

$$\begin{pmatrix} \boldsymbol{H}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{G}_{\mathscr{A}}} \\ \boldsymbol{G}_{\mathscr{B}} \otimes \widetilde{\boldsymbol{H}_{\mathscr{A}}}^{-1} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \boldsymbol{G}_{\mathcal{H}(\mathscr{B})} \\ \boldsymbol{H}'_{\mathscr{B}} \end{pmatrix} \otimes \begin{pmatrix} \widetilde{\boldsymbol{G}}_{\mathcal{H}(\mathscr{A})} \\ \widetilde{\boldsymbol{G}_{\mathscr{A}}}' \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{G}_{\mathcal{H}(\mathscr{B})} \\ \boldsymbol{G}'_{\mathscr{B}} \end{pmatrix} \otimes \begin{pmatrix} \widetilde{\boldsymbol{G}}^{-1}_{\mathcal{H}(\mathscr{A})} \\ \widetilde{\boldsymbol{H}_{\mathscr{A}}}' \end{pmatrix} \end{pmatrix}$$

Note that when $q = 3$ we have $\widetilde{\boldsymbol{G}}_{\mathcal{H}(\mathscr{A})} = \widetilde{\boldsymbol{G}}^{-1}_{\mathcal{H}(\mathscr{A})}$ thus we have h^2 redundant equations and the rank is bounded by $2k(n-k) + 2n - 1 - h^2$. For $q > 3$, $\widetilde{\boldsymbol{G}}_{\mathcal{H}(\mathscr{A})}$ not necessarily equals to $\widetilde{\boldsymbol{G}}^{-1}_{\mathcal{H}(\mathscr{A})}$ thus we have different systems and the rank is bounded by $2k(n-k) + 2n - 1$ ■

Let $q = p^m$. The number of variables in the system is $n^2(p^m - 1)$ while the number of equations in the best case when the rate is 0.5 ($k = n/2$) and Frobenius action produces full rank matrices is bounded by $mn^2/2 + 2n - 1$. If we ignore the term $2n - 1$ we find the ratio between the number of variables and the rank is about $\dfrac{2(p^m - 1)}{m}$. Figure 4.3 illustrates this ratio and shows that solving DEP using this algorithm becomes more complex as q tends to infinity, here we set $p = 2$.

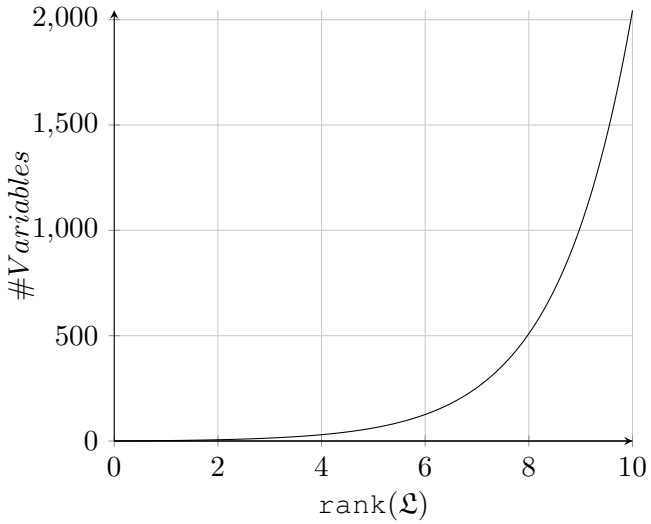


Figure 4.3: Growth of number of variable compared to the rank of the system

4.5.3 Randomized Algorithm for Solving the Polynomial System

In this section we present a polynomial time algorithm to solve DEP using the linear part of the system $\mathfrak{S}$ denoted by $\mathfrak{L}$ over the fields $\mathbb{F}_3$ and $\mathbb{F}_4$. We use try and guess strategy in addition to the techniques that were developed for PEP.

1: **Input:** Two linear codes $\mathscr{A}$ and $\mathscr{B}$ over $\mathbb{F}_q$.
2: **Output:** Set of puncturing permutations corresponding to the diagonal equivalence between $\mathscr{A}$ and $\mathscr{B}$.

Build the linear system $\mathfrak{L}$ and its matrix $\boldsymbol{L}$.
Add the equations that correspond to fixing one element in the diagonal matrix $\boldsymbol{D}$ to $\boldsymbol{L}$.
Apply Frobenius action if applicable.
Process $\boldsymbol{L}$ by using block linearization technique and update the system by adding equations of the form $x_{i,j} = 0$. The block size is $n(q-1)$ thus we have n blocks.
for all variables that correspond to columns 2 to n in $\widetilde{\boldsymbol{P}}$ **do**
    $\boldsymbol{L}' = \boldsymbol{L}$
    Guess one variable x to be 1, this adds $(2n-1)(q-1)$ equations to $\boldsymbol{L}'$.
    Solve the updated system $\boldsymbol{L}'$ by block linearization.
    if The null space of $\boldsymbol{L}'$ is **0 then**
        Update $\boldsymbol{L}$ with the equation $x = 0$.
        Continue to the next iteration of the loop.
    else if The null space of $\boldsymbol{L}'$ is small **then**
        Search this space for puncturing permutations.
        Break the loop.
    else
        $\boldsymbol{L} = \boldsymbol{L}'$.
    end if
end for
return The set of puncturing permutations.

Experimentally we found that this algorithm solves DEP successfully for random codes over $\mathbb{F}_3$ and $\mathbb{F}_4$. For $\mathbb{F}_q$ with $q = p^m \geqslant 5$ it does not find a small enough null space thus the problem is complex to solve. The reason is that the length of the closure increases exponentially in m. Thus the number of variables also becomes exponential in m while the enhancement in the rank of the linear system using Frobenius action is linear in m. Thus as the size of the field increases the dimension of the null space increases with almost the same rate.

This approach fails over $\mathbb{F}_3$ when the hull is not trivial. That is because according to Proposition 4.5.11 the the rank of the linear system decreases by h^2, h is the dimension of the hull, thus some important equations that help in solving by block linearization will disappear. This is not the case over $\mathbb{F}_4$, again see Proposition 4.5.11, where the rank does not decrease as in $\mathbb{F}_3$ and the approach can solve DEP for codes with large hull and weakly self-dual codes but not self-dual codes, see the tables of experiments 4.4 and 4.5.

4.5.4 Complexity

The matrix $\boldsymbol{L}$ has $n^2(q-1)$ columns. We assume q is constant thus the cost of Gaussian elimination is $O(n^{2\omega})$. The block linearization needs to perform the Gaussian elimination n times (we have n blocks of size $n(q-1)$). Thus the complexity of block linearization is $O(n^{1+2\omega})$. For each variable in the columns from 2 to n in $\widetilde{\boldsymbol{P}}$ we guess and solve by block linearization, thus the total complexity is $O(n^{3+2\omega})$.

Experimentally for random codes over $\mathbb{F}_3$ and $\mathbb{F}_4$ with a small enough solution set we only need to guess exactly one or two correct position of 1 in the column from 2 to n. Thus we can assume the number of of guesses is $O(n)$ to find a subset of the solution set. Thus the complexity in practice is $O(n^{2+2\omega})$.

n	k	Codim	#Sol	#Guess	#Cols	Time (s)	Memory (MB)
30	4	10	384	77	2	$2^{6.229}$	2^5
	6	1	2	51	1	$2^{5.858}$	2^5
	15	1	2	60	1	$2^{6.629}$	2^5
50	5	12	4,096	95	1	$2^{9.077}$	2^5
	10	1	2	61	1	$2^{8.966}$	2^6
	25	1	2	69	1	$2^{9.644}$	2^6
100	10	1	2	171	1	$2^{13.736}$	$2^{8.140}$
	20	1	2	312	2	$2^{15.328}$	$2^{8.267}$
	50	1	2	20	1	$2^{12.254}$	$2^{11.315}$

Table 4.4: Solving DEP for random codes with trivial hull over $\mathbb{F}_3$

n	k	h	Codim	#Sol	#Guess	#Cols	Time (s)	Memory (MB)
30	6	0	1	3	73	1	$2^{7.665}$	2^5
	6	6	2	6	52	1	$2^{7.180}$	2^5
	15	0	1	3	52	1	$2^{7.585}$	2^5
		6	1	3	73	1	$2^{8.061}$	2^5
		14	1	3	88	1	$2^{8.276}$	2^5
50	10	0	1	3	61	1	$2^{8.966}$	2^6
	10	10	1	3	20	1	$2^{8.611}$	2^6
	25	0	1	3	141	1	$2^{11.845}$	2^6
		10	1	3	64	1	$2^{10.755}$	2^6
		24	1	3	297	2	$2^{12.520}$	2^6
100	20	0	1	3	180	1	$2^{15.616}$	$2^{8.366}$
	20	20	1	3	250	1	$2^{16.183}$	$2^{8.531}$
	50	0	1	3	15	1	$2^{13.052}$	$2^{9.180}$
		20	1	3	79	1	$2^{15.313}$	$2^{9.180}$
		49	1	3	260	1	$2^{16.959}$	$2^{9.180}$

Table 4.5: Solving DEP for random codes over $\mathbb{F}_4$

- Codim: Dimension of the null space of the final linear system.
- #Sol: Number of actual solutions.
- #Guess: Number of guesses tried.
- #Cols: Number of columns tried.

4.6 Conclusion

This chapter was dedicated to diagonal equivalence. We introduced two algebraic models to solve the problem. Solving DEP with Groebner basis is much harder in computation than PEP. We reduce diagonal equivalence to permutation equivalence by using the notion of the closure of the code. The closure has the disadvantage of extending the length of the code, thus the number of variables in the algebraic model increases. A new model that combines the code and the closure was introduced in order to reduce the number of variables. This model was useful to solve the problem over $\mathbb{F}_3$ and $\mathbb{F}_4$ by using our techniques for solving PEP. In this case we obtain a polynomial complexity in n. The ISD approach turned out to be useful to solve the problem in the general case where we obtain a complexity

$$2^{\left(-n\log_2(R(1-R))\right)\left(1+O(1)\right)}.$$

Chapter 5

Application to Graph Isomorphism

In this chapter we show how our approaches of solving code equivalence can be used to solve graph isomorphism. We give a reduction from permutation equivalence of codes with trivial hull to graph isomorphism. Then we solve the problem according to our strategies that we have developed in Chapter 3. For quick review of the relationship between the two problems we refer the reader to Section 1.5. Firstly we recall some basic definitions and notation about graphs.

A graph $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is composed of a finite set of vertices $V_\mathcal{G}$ and a set of edges $E_\mathcal{G} \subset V_\mathcal{G} \times V_\mathcal{G}$. For the sake of simplicity, we will always assume that $V_\mathcal{G} = [1, n]$. The graph $\mathcal{G}$ is undirected if for each (i, j) in $E_\mathcal{G}$ we also have (j, i) in $E_\mathcal{G}$, otherwise $\mathcal{G}$ is a directed graph. Classically, the adjacency matrix $\boldsymbol{A}^\mathcal{G} = \left[a^\mathcal{G}_{i,j}\right]$ of a graph $\mathcal{G}$ is a binary $n \times n$ matrix such that $a^\mathcal{G}_{i,j} = 1$ if $(i, j) \in E_\mathcal{G}$ and $a^\mathcal{G}_{i,j} = 0$ otherwise. In particular, $\boldsymbol{A}^\mathcal{G}$ is symmetric when $\mathcal{G}$ is undirected. However, as explained in [59], it is possible to slightly change the definition so that the adjacency matrix of a directed graph is symmetric. For that purpose, if $\mathcal{D}$ is a directed graph then its adjacency matrix $\boldsymbol{A}^\mathcal{D}$ is defined as

$$a^\mathcal{D}_{i,j} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } (i,j) \in E_\mathcal{G} \text{ and } (j,i) \in E_\mathcal{G}, \\ 2 & \text{if } (i,j) \in E_\mathcal{G} \text{ and } (j,i) \notin E_\mathcal{G}, \\ 3 & \text{if } (i,j) \notin E_\mathcal{G} \text{ and } (j,i) \in E_\mathcal{G}, \\ 0 & \text{otherwise.} \end{cases}$$

Notice also that in our way of representing graphs, we do not consider here multi-edge graphs. Since $\boldsymbol{A}^\mathcal{G}$ are symmetric and real, all its n eigenvalues are real. They are denoted by $\lambda_1(\mathcal{G}), \ldots, \lambda_n(\mathcal{G})$. The spectrum $\Lambda(\mathcal{G})$ of $\mathcal{G}$ is the set of its eigenvalues. The eigenspace associated to an eigenvalue λ is denoted by $V_\lambda(\mathcal{G})$. We recall that for any pair of distinct eigenvalues λ and β the eigenspace $V_\lambda(\mathcal{G})$ and $V_\lambda(\mathcal{G})$ are orthogonal.

5.1 Code Equivalence and Graph Isomorphism

Recall the definition of $\boldsymbol{\Sigma}_\mathscr{A}$ from Section 3.4

$$\boldsymbol{\Sigma}_\mathscr{A} \stackrel{\text{def}}{=} \boldsymbol{G}^T_\mathscr{A}\left(\boldsymbol{G}_\mathscr{A}\boldsymbol{G}^T_\mathscr{A}\right)^{-1}\boldsymbol{G}_\mathscr{A}.$$

Note that $\mathbf{\Sigma}_{\mathscr{A}}$ is an $n \times n$ symmetric matrix hence we may interpret it as an adjacency matrix of a weighted undirected graph. Referring to Proposition 3.4.14 which states the following

Proposition. *The linear systems* $\mathfrak{L}$ *and* $\left\{\mathbf{\Sigma}_{\mathscr{A}}\boldsymbol{P} - \boldsymbol{P}\mathbf{\Sigma}_{\mathscr{B}}\right\}$ *are equivalent.*

we obtain the following result.

Theorem 5.1.1. *Let* $\mathscr{A}$ *and* $\mathscr{B}$ *two linear codes with trivial hull of length* n *over a field* $\mathbb{F}$*. There is an* $O\left(n^{\omega}\right)$ *reduction from the permutation equivalence between* $\mathscr{A}$ *and* $\mathscr{B}$ *to the graph isomorphism between two weighted undirected graphs having* n *vertices and weights in* $\mathbb{F}$*.*

5.2 Graph Isomorphism Testing

Definition 5.2.1. Let $\mathbb{F} \subseteq \mathbb{R}$ be a field such that $\boldsymbol{A}^{\mathcal{G}}$ is viewed as a matrix with entries in $\mathbb{F}$. The linear subspace of $\mathbb{F}^n$ generated by the rows of $\boldsymbol{A}^{\mathcal{G}}$ is the linear code $C_{\mathcal{G}}$ associated to $\mathcal{G}$. This code is only interesting when $\boldsymbol{A}^{\mathcal{G}}$ is not of full rank. Another way of seeing $C_{\mathcal{G}}$ is by remarking that the eigenvector space $V_0(\mathcal{G})$ associated to 0 is actually the dual of $C_{\mathcal{G}}$, and consequently entails the following

$$C_{\mathcal{G}} = V_0(\mathcal{G})^{\perp}$$

For any permutation σ from S_n and graph $\mathcal{G}$ we define $\mathcal{G}^{\sigma}$ as the graph where $(i,j) \in E_{\mathcal{G}}$ if and only if $(\sigma(i), \sigma(j)) \in E_{\mathcal{G}^{\sigma}}$. An automorphism for $\mathcal{G}$ is a permutation σ such that $\mathcal{G}^{\sigma} = \mathcal{G}$. The automorphism group of $\mathcal{G}$ is the set $\Pi\left(\mathcal{G}\right) \stackrel{\text{def}}{=} \left\{\sigma \in S_n \ : \ \mathcal{G}^{\sigma} = \mathcal{G}\right\}$.

Recall that two graphs $\mathcal{G} = ([1,n], E_{\mathcal{G}})$ and $\mathcal{H} = ([1,n], E_{\mathcal{H}})$ are isomorphic if there exists σ in S_n such that $(i,j) \in E_{\mathcal{G}}$ if and only if $(\sigma(i), \sigma(j)) \in E_{\mathcal{H}}$. There exists therefore an $n \times n$ matrix $\boldsymbol{P}$ in S_n such that

$$\boldsymbol{A}^{\mathcal{H}} = \boldsymbol{P}^T \boldsymbol{A}^{\mathcal{G}} \boldsymbol{P}. \tag{5.1}$$

Recall that we have also $\mathbb{1}_n \boldsymbol{P} = \mathbb{1}_n \boldsymbol{P}^T = \mathbb{1}_n$. Hence, we associate a polynomial system modeling the graph isomorphism problem as $\mathfrak{I}(\mathcal{G}, \mathcal{H}) \cup \mathfrak{Q}$ with

$$\mathfrak{I}(\mathcal{G}, \mathcal{H}) \ \stackrel{\text{def}}{=} \ \left\{\mathbb{1}_n \boldsymbol{P} - \mathbb{1}_n \ , \ \mathbb{1}_n \boldsymbol{P}^T - \mathbb{1}_n \ , \ \boldsymbol{P}\boldsymbol{A}^{\mathcal{H}} - \boldsymbol{A}^{\mathcal{G}}\boldsymbol{P}\right\}. \tag{5.2}$$

Remark 5.2.2. Another way of viewing (5.1) is to rewrite it as $\boldsymbol{P}\boldsymbol{A}^{\mathcal{H}} = \boldsymbol{A}^{\mathcal{G}}\boldsymbol{P}$ which is in turn equivalent to

$$\left(\boldsymbol{A}^{\mathcal{H}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}^{\mathcal{G}}\right)\overline{\boldsymbol{P}} = \boldsymbol{0}. \tag{5.3}$$

When $\boldsymbol{A}^{\mathcal{G}}$ is invertible then (5.3) is equivalent to the linear system $\left(\boldsymbol{A}^{\mathcal{G}}\right)^{-1}\boldsymbol{P}\boldsymbol{A}^{\mathcal{H}} = \boldsymbol{P}$, that is to say

$$\left(\boldsymbol{A}^{\mathcal{H}} \otimes \left(\boldsymbol{A}^{\mathcal{G}}\right)^{-1} - \boldsymbol{I}_{n^2}\right)\overline{\boldsymbol{P}} = \boldsymbol{0}. \tag{5.4}$$

We now state important facts about eigenspaces for isomorphic graphs. We start by a lemma reminding a well-known property and which entails that eigenspaces of isomorphic graphs are permutation equivalent.

Lemma 5.2.3. *Let $\mathcal{G}$ be a graph with n vertices and $\boldsymbol{P}$ be permutation from S_n, then if $\boldsymbol{v}$ is an eigenvector associated to the eigenvalue λ for $\boldsymbol{A}^{\mathcal{G}}$ then $\boldsymbol{P}^T\boldsymbol{v}$ is an eigenvector associated to λ for $\boldsymbol{P}^T\boldsymbol{A}^{\mathcal{G}}\boldsymbol{P}$.*

The next theorem follows then immediately from this lemma.

Theorem 5.2.4. *Let σ be a permutation from S_n. Then $\mathcal{H} = \mathcal{G}^\sigma$ if and only if for all λ in Λ we have*

$$V_\lambda(\mathcal{H}) = V_\lambda(\mathcal{G})^\sigma.$$

In particular, if $C_{\mathcal{G}}$ and $C_{\mathcal{H}}$ are of dimension $< n$ over $\mathbb{F}$ then $C_{\mathcal{H}} = (C_{\mathcal{G}})^\sigma$.

We recall that two isomorphic graphs have to satisfy (at least) the following assumptions.

Assumption 5.2.5 (Folklore)**.** *We consider throughout this section two graphs $\mathcal{G}$ and $\mathcal{H}$ with n vertices such that the following holds for any field $\mathbb{F} \subseteq \mathbb{R}$*

1. $\mathtt{rank}_{\mathbb{F}}\, \boldsymbol{A}^{\mathcal{H}} = \mathtt{rank}_{\mathbb{F}}\, \boldsymbol{A}^{\mathcal{G}}$
2. $\Lambda \overset{def}{=} \Lambda(\mathcal{G}) = \Lambda(\mathcal{H})$ *and*

$$\forall \lambda \in \Lambda,\ \ d_{\mathbb{F}}(\lambda) \overset{def}{=} \dim_{\mathbb{F}} V_\lambda(\mathcal{G}) \cap \mathbb{F}^n = \dim_{\mathbb{F}} V_\lambda(\mathcal{H}) \cap \mathbb{F}^n.$$

3. $h \overset{def}{=} \dim_{\mathbb{F}} \mathcal{H}(C_{\mathcal{G}}) = \dim_{\mathbb{F}} \mathcal{H}(C_{\mathcal{H}})$.

One would expect that the graph isomorphism problem would be solved completely just by solving $\mathfrak{I}(\mathcal{G}, \mathcal{H})$ since the number of variables is n^2 and the number of linear equations is $n^2 + 2n - 1$, or at least, when the graphs are not isomorphic, the rank of the linear system should be n^2 so that the only solution is $\mathbf{0}$. For randomly generated graphs this happens with high probability but this not the case for "structured" graphs like for instance those that do not have a trivial automorphism group.

Actually, the first important issue about the solving of $\mathfrak{I}(\mathcal{G}, \mathcal{H}) \cup \mathfrak{Q}$ is to fix the underlying field on which the vector spaces are considered. Indeed, the linear system can be considered over $\mathbb{R}$, $\mathbb{Q}$ and any prime field $\mathbb{F}_p$ where p is a prime number $\geqslant 2$. This ambiguity comes mainly from the fact that the solutions of $\mathfrak{I}(\mathcal{G}, \mathcal{H})$ are sought among the binary vectors, and by construction the entries of an adjacency matrix lie in $\mathbb{Z}$. The choice of the underlying field is crucial as it is shown by the following theorem.

Theorem 5.2.6. *Assume that $\boldsymbol{A}^{\mathcal{G}}$ and $\boldsymbol{A}^{\mathcal{H}}$ are matrices with entries in a field $\mathbb{F}$.*

1. *If $\boldsymbol{A}^{\mathcal{G}}$ is invertible then*

$$\mathtt{rank}_{\mathbb{F}} \left(\boldsymbol{A}^{\mathcal{H}} \otimes \left(\boldsymbol{A}^{\mathcal{G}}\right)^{-1} - \boldsymbol{I}_{n^2} \right) = n^2 - \sum_{\lambda \in \Lambda} d_{\mathbb{F}}(\lambda)^2. \tag{5.5}$$

2. *If $\boldsymbol{A}^{\mathcal{G}}$ is not invertible then*

$$\mathtt{rank}_{\mathbb{F}} \Big(\boldsymbol{A}^{\mathcal{H}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}^{\mathcal{G}} \Big) = 2 d_{\mathbb{F}}(0) \Big(n - d_{\mathbb{F}}(0) \Big). \tag{5.6}$$

Proof. The equality (5.5) when $\boldsymbol{A}^{\mathcal{G}}$ is invertible comes from the fact that

$$\ker\left(\boldsymbol{A}^{\mathcal{H}} \otimes \left(\boldsymbol{A}^{\mathcal{G}}\right)^{-1} - \boldsymbol{I}_{n^2}\right) = \bigoplus_{\lambda\in\Lambda} V_{\lambda}(\mathcal{H}) \otimes V_{\lambda^{-1}}(\mathcal{G}).$$

Let us assume that $\boldsymbol{A}^{\mathcal{G}}$ is not invertible. We denote by $\boldsymbol{G}_{\mathcal{G}}$ and $\boldsymbol{H}_{\mathcal{G}}$ a generator matrix and a parity check matrix of $C_{\mathcal{G}}$. There exists then $\boldsymbol{K}_{\mathcal{G}}$ such that $\boldsymbol{G}_{\mathcal{G}} = \boldsymbol{K}_{\mathcal{G}}\boldsymbol{A}^{\mathcal{G}}$. We extend these notation to the graph $\mathcal{H}$. Hence we have the following equalities

$$(\boldsymbol{K}_{\mathcal{H}} \otimes \boldsymbol{H}_{\mathcal{G}})\left(\boldsymbol{A}^{\mathcal{H}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}^{\mathcal{G}}\right) = \boldsymbol{G}_{\mathcal{H}} \otimes \boldsymbol{H}_{\mathcal{G}} \tag{5.7}$$

$$(\boldsymbol{H}_{\mathcal{H}} \otimes \boldsymbol{K}_{\mathcal{H}})\left(\boldsymbol{A}^{\mathcal{H}} \otimes \boldsymbol{I}_n - \boldsymbol{I}_n \otimes \boldsymbol{A}^{\mathcal{G}}\right) = \boldsymbol{H}_{\mathcal{H}} \otimes \boldsymbol{G}_{\mathcal{G}} \tag{5.8}$$

Since by assumption $C_{\mathcal{G}}$ has a trivial hull, by using the same arguments to prove Proposition 3.4.14, we can see that the linear system (5.3) is equivalent to the linear system

$$\begin{bmatrix} \boldsymbol{G}_{\mathcal{H}} \otimes \boldsymbol{H}_{\mathcal{G}} \\ \boldsymbol{H}_{\mathcal{H}} \otimes \boldsymbol{G}_{\mathcal{G}} \end{bmatrix} \overline{\boldsymbol{P}} = \boldsymbol{0}.$$

The equality (5.6) comes then from Proposition 3.2.23 and $\mathtt{rank}_{\mathbb{F}}\boldsymbol{H}_{\mathcal{H}} = \mathtt{rank}_{\mathbb{F}}\boldsymbol{H}_{\mathcal{G}} = d_{\mathbb{F}}(0)$. ■

In light of Theorem 5.2.4 and Theorem 5.2.6, the most favorable situation is to take a field $\mathbb{F}$ such that at least one of following properties hold, namely

1. Either $d_{\mathbb{F}}(\lambda) = 0$ for all λ in Λ,
2. the eigenspaces $V_{\lambda}(\mathcal{G})$ have trivial hulls.

That is why we choose $\mathbb{F}$ to be either $\mathbb{Q}$ or $\mathbb{F}_p$ where p is such that at least one eigenspace $V_{\lambda}(\mathcal{G})$ has a trivial hull when it exists.

Tables 5.1 and 5.2 are experiments for random graphs for isomorphic and non-isomorphic case. We also implemented experiments for some connected 3-regular (cubic) simple graphs including Petersen graph, cubical graph, Wagner graph, and graphs with 12 vertices, see Figures 5.3, 5.4 and 5.5. For these regular graphs the experiments include isomorphic and non-isomorphic cases where we are able to decide for non-isomorphic case using block linearization and to retrieve one solution by using guess strategy for isomorphic graphs.

n	#ISO	codim	Time (s)	Memory (MB)
50	1	1	$2^{2.807}$	$2^{6.687}$
100	1	1	$2^{7.524}$	$2^{10.255}$
150	1	1	$2^{11.222}$	$2^{12.305}$
200	1	1	$2^{13.203}$	$2^{13.943}$
250	1	1	$2^{14.267}$	$2^{15.509}$
300	1	1	$2^{14.644}$	$2^{16.309}$

Table 5.1: Finding isomorphism for random graphs with unique solution

n	Time (s)	Memory (MB)
50	$2^{2.322}$	$2^{6.687}$
100	$2^{5.044}$	$2^{10.032}$
150	$2^{8.707}$	$2^{12.305}$
200	$2^{11.232}$	$2^{13.973}$
250	$2^{14.200}$	$2^{15.228}$
300	$2^{14.505}$	$2^{16.278}$

Table 5.2: Deciding non-isomorphism for random graphs

5.3 Conclusion

In this chapter we studied graph isomorphism problem. We used our algebraic model for permutation equivalence to model the problem. We introduced a polynomial time reduction from permutation equivalence problem for codes with trivial hull to graph isomorphism. The algebraic model corresponding to graph isomorphism contains more linear equations than the number of variables. This enables to decide graph isomorphism in the case of random graphs with trivial automorphism groups. Our experiments also included some small regular graphs.

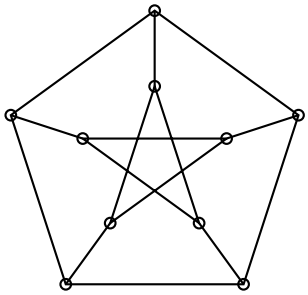


Figure 5.3: Petersen Graph

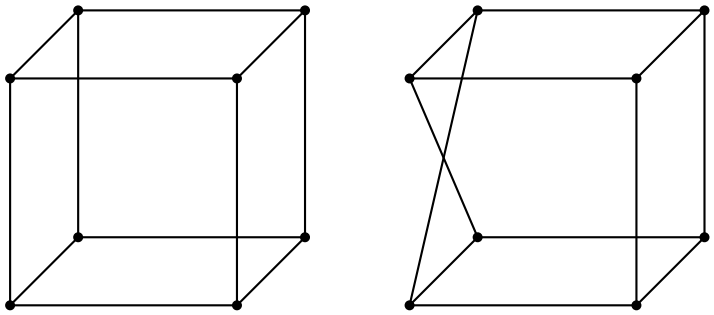


Figure 5.4: Cubical and Wagner Graphs

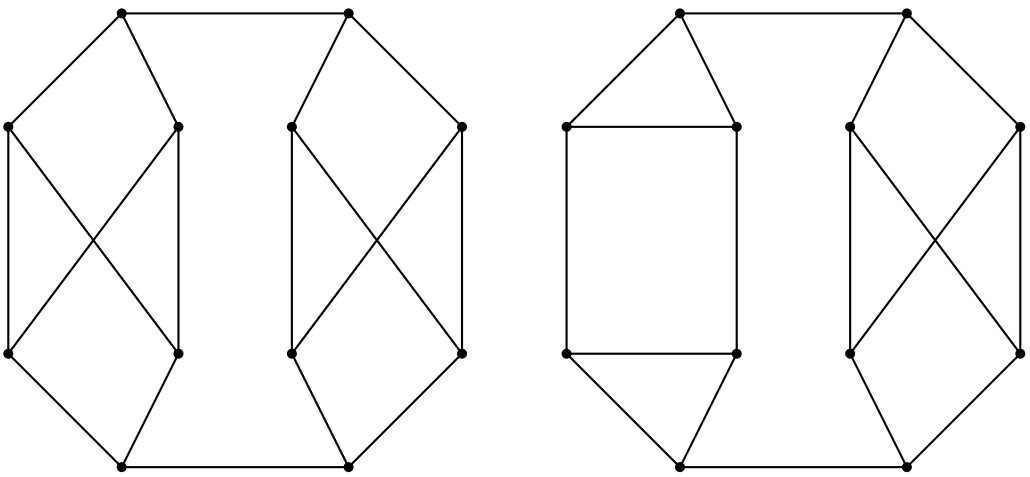


Figure 5.5: 12 vertices connected 3-regular graphs

Conclusion and Future Work

This thesis was the first attempt to solve code equivalence problem using algebraic approach. We developed algebraic models for permutation equivalence as well as diagonal equivalence. In both cases we proved that the problems are well-described using these models. We solved the algebraic systems using Groebner basis $\mathbb{F}_4$ algorithm implemented in Magma. When used directly with these models, Groebner basis computation has high complexity. We improved the algebraic system by enhancing the linear part and/or reducing the number of variables. These improvements were very efficient in identifying large number of easy instances of the problem. To improve the linear part of the system we used block linearization, Frobenius action and guessing strategy. Beside algebraic approaches we also considered other approaches to solve code equivalence such as ISD and punctured codes.

Using our methods of block linearization and Frobenius action we found that permutation equivalence problem seems to be easy for large instances of linear codes where we were able to provide polynomial-time algorithms. Diagonal equivalence is at least as hard as permutation equivalence. It can be reduced to permutation equivalence in polynomial-time. We introduced a new modeling for diagonal equivalence using the notion of the closure of the code. Using this with our strategies for solving permutation equivalence we were able to solve efficiently the instances of diagonal equivalence over $\mathbb{F}_3$ with trivial hull and over $\mathbb{F}_4$.

We introduced a new reduction from permutation equivalence problem to graph isomorphism when the hull is trivial. This reduction enabled to use our algebraic modeling that was developed for permutation equivalence to solve graph isomorphism. The algebraic system of the graph isomorphism has interesting properties in terms of the rank of the linear part and the number of variables. We implemented experiments for random graphs and regular graphs. In both cases we were able to solve the problem efficiently. For random graphs usually the linear system is sufficient to decide graph isomorphism and to get the solutions while for regular graphs we need to use block linearization. If the set of isomorphism between the graphs is large we can use guessing strategy to obtain a small subset of solutions or one solution.

Comparing our results with Leon's algorithm implemented in Magma we can solve for wider range of parameters over large fields. Leon's algorithm works only for codes of small length over small prime fields or $\mathbb{F}_4$. Beside that Leon's algorithm has exponential complexity in the dimension of the code because it computes the minimum weight codewords. The SSA algorithm works efficiently only for codes with hull of small dimension (constant). Its complexity becomes intractable for codes with large hull. In our case we can solve efficiently for codes with large hull over extension fields. Our result for diagonal equivalence confirm the results of [78] using the SSA. We were able to solve efficiently diagonal equivalence for random codes over $\mathbb{F}_3$ and $\mathbb{F}_4$. For $\mathbb{F}_q$ with $q \geqslant 5$ our method did not work.

The future work of our thesis represented in many areas. The first is the area of Groebner basis behavior and complexity. The complexity of Groebner basis in the worst case is double exponential but for random systems it is simply exponential. The questions regarding our algebraic systems of code equivalence are: what is complexity bound to solve the system using Groebner basis? What is the effect of the hull? Experimentally, we found that when the hull is trivial and there is a unique solution Groebner basis performs only the step of linearization. That means the maximum degree of polynomials is 2. We performed the linearization by our own implementation and the bound of the maximum degree is confirmed. In this case the complexity of Groebner basis computation is polynomial. If there are many solutions or the hull is large we can reach higher polynomials degree. These experimental results need to be proven.

For our methods, the hull was a source of difficulty especially over prime fields. One reason is that it decreases the rank of the linear part of the system. Over non-prime fields we can use Frobenius action to overcome the effect of the hull in reducing the rank. The effect of the large hull does not seem to be only in degrading the rank of the linear part of the system but also in the structure of codes. That is because when we compare two systems with linear parts of equal rank but one is coming from codes with large hull and the other from codes with lower rate and trivial hull the later behaves better in computation. The questions are: What are the other effects of the hull? and how to overcome them especially for codes over prime fields? One idea that we introduced is to eliminate the hull by using the shortened code and to obtain new codes with trivial hull.

N. Sendrier and D. Simos conjectured in [78] that the diagonal equivalence problem might be hard for almost all instances over $\mathbb{F}_q$ with $q \geqslant 5$. In spite of using different technique from SSA we found that we were able to solve diagonal equivalence efficiently over $\mathbb{F}_3$ and $\mathbb{F}_4$ and it was difficult over $\mathbb{F}_q$ with $q \geqslant 5$. Thus proving (or disproving) the conjecture of [78] is an open problem of research.

Our modeling seems to be of interest for solving graph isomorphism problem. We implemented many experiments in random and regular graphs and we were able to decide the isomorphism of these graphs efficiently. We raise the following question: Can we prove something or improve the complexity bound of graph isomorphism problem using this approach?

Solving code equivalence problem efficiently is an important step toward efficient cryptanalysis of code-based cryptosystem. In code equivalence problem we assume that we know both the secret and public codes. In cryptosystems this is not the case where we know only the public code. Thus the problem for cryptanalysis is much harder. Experimentally we treated successfully some types of codes that are used for cryptosystems such binary Goppa codes. In this case, we used linearization to enhance the system which has polynomial cost but it becomes impractical when the length of the code is large. Optimization need to be done in order to treat codes with large length as used in cryptosystems.

McEliece problem is more related to the problem of subcode equivalence. Given two codes the subcode equivalence problem is to answer the question: Is one code equivalent to a subcode of the other? In the case of McEliece cryptosystem, Goppa codes are subfield subcodes of the generalized Reed-Solomon codes. The subcode equivalence problem is `NP-complete` in general, see [10]. The question is: can we identify the easy instances of this problem? This enables to identify the weak instances of McEliece cryptosystem.

Bibliography

[1] Adámek, J. (1991). Foundations of Coding, Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory. John Wiley & Sons, Czech Technical University in Prague.

[2] Ayad, A. (2010). A survey on the complexity of solving algebraic systems. International Mathematical Forum, 5:333–353.

[3] Babai, L., Codenotti, P., Grochow, J., and Qiao, Y. (2011). Code equivalence and group isomorphism. In Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Dis-crete Algorithms, pages 1395–1408. SIAM.

[4] Babai, L. and Luks, E. M. (1983). Canonical labeling of graphs. In Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, STOC '83, pages 171–183, New York, NY, USA. ACM.

[5] Bardet, M., Faugère, J.-C., S., B., and Yang, B. (2005). Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry.

[6] Bardet, M., Faugère, J.-C., and Salvy, B. (2004). On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In Proc. International Conference on Polynomial System Solving (ICPSS), pages 71–75.

[7] Bardet, M., Faugère, J.-C., and Salvy, B. (2015). On the complexity of the F_5 Gröbner basis algorithm. J. Symb. Comput., 70(C):49–70.

[8] Bardet, M. (2004). Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. PhD thesis, University Paris 6.

[9] Bayer, D. and Stillman, M. (1987). A criterion for detecting m-regularity. Invent. Math., 87:1–11.

[10] Berger, T. P., Gueye, C. T., and Klamti, J. B. (2017). A NP-Complete Problem in Coding Theory with Application to Code Based Cryptography, pages 230–237. Springer International Publishing, Cham.

[11] Berlekamp, E., McEliece, R., and van Tilborg, H. (1978). On the inherent intractability of certain coding problems. IEEE Trans. Inf. Theor., 24(3):384–386.

[12] Bernstein, D., Buchmann, J., and E, D., editors (2009). Post-Quantum Cryptography. Springer, Berlin Heidelberg.

[13] Bernstein, D. J., Lange, T., and Peters, C. (2008). Attacking and Defending the McEliece Cryptosystem. In PQCrypto, pages 31–46.

[14] Bettale, L., Faugère, J.-C., and Perret, L. (2009). Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology, 3(3):177–197.

[15] Bogart, K., Goldberg, D., and Gordon, J. (1978). An elementary proof of the MacWilliams theorem on equivalence of codes. Information and Control, 37(1):19–22.

[16] Bosma, W., Cannon, J., and Playoust, C. (1997). The Magma algebra system. I. The user language. J. Symbolic Comput., 24(3-4):235–265. Computational algebra and number theory (London, 1993).

[17] Bouyukliev, I. (2007). About the Code Equivalence. In Shaska, T., Huffman, W. C., Joyner, D., and Ustimenko, V., editors, Advances in Coding Theory and Cryptography, volume 3 of On Coding Theory and Cryptology, pages 126–151, Singapore. WSPC Proceedings.

[18] Buchberger, B. (2006). Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symb. Comput., 41(3-4):475–511.

[19] Canteaut, A. and Chabaud, F. (1998). A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Transactions on Information Theory, 44(1):367–378.

[20] Christiane, P. (2010). Information-set decoding for linear codes over $\mathbb{F}_q$. In PQCrypto 2010, pages 81–94.

[21] Collart, S., Kalkbrener, M., and Mall, D. (1997). Converting bases with the Gröbner walk. Journal of Symbolic Computation, 24:465–469.

[22] Coppersmith, D. and Winograd, S. (1990). Matrix multiplication via arithmetic progressions. J. Symb. Comput., 9(3):251–280.

[23] Courtois, N., Goubin, L., Meier, W., and Tacier, J.-D. (2002). Solving underdefined systems of multivariate quadratic equations. In Public Key Cryptography, volume 2274 of Lecture Notes in Computer Science, pages 211–227. Springer.

[24] Courtois, N., Klimov, A., Patarin, J., and Shamir, A. (2000). Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Advances in Cryptology EUROCRYPT 2000, volume 1807 of Lecture Notes in Computer Science, pages 392–407. Springer.

[25] Cox, D. A., Little, J. B., and O'Shea, D. (2001). Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra. Undergraduate Texts in Mathematics, Springer-Verlag, New York.

[26] Delsarte, P. (1975). On subfield subcodes of modified Reed-Solomon codes. IEEE Transactions on Information Theory, 21:515–516.

[27] Ding, J., Gower, J. E., and Schmidt, D. (2006). Multivariate Public Key Cryptosystems (Advances in Information Security). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[28] Dubé, T. W. (1990). The structure of polynomial ideals and Gröbner bases. SIAM J. Comput., 19(4):750–773.

[29] Engelbert, D., Overbeck, R., and Schmidt, A. (2007). A Summary of McEliece-Type Cryptosystems and their Security. Journal of Mathematical Cryptology, 1:151–199.

[30] Faugère, J.-C. (1999). A New Efficient Algorithm for Computing Gröbner Bases (F4). Journal of Pure and Applied Algebra, 139(1-3):61–88.

[31] Faugère, J.-C. (2002). A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero : F5. In ISSAC 02, pages 75–83. ACM press.

[32] Faugère, J.-C., Gauthier-Umana, V., Otmani, A., Perret, L., and Tillich, J. (2011). Distinguisher for High Rate McEliece Cryptosystems. In Proceedings of the 2011 IEEE Information Theory Workshop (ITW 2011), Paraty, Brazil.

[33] Faugère, J.-C., Gianni, P. M., Lazard, D., and Mora, T. (1993). Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. J. Symb. Comput., 16(4):329–344.

[34] Faugère, J.-C. and Gwénolé, A. (2004). Comparison of XL and Gröbner basis algorithms over Finite Fields. Technical Report RR-5251, INRIA.

[35] Faugère, J.-C. and Joux, A. (2003). Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases, pages 44–60. Springer Berlin Heidelberg, Berlin, Heidelberg.

[36] Faugère, J.-C., Otmani, A., Perret, L., and Tillich, J. (2010). Distinguisher for High Rate McEliece Cryptosystem. In Yet Another Conference on Cryptography (YACC 2010), Porquerolles Island, France.

[37] Feulner, T. (2009). The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. Advances in Mathematics of Communications, 3:363–383.

[38] Gabidulin, E. M. (1995). Public-key Cryptosystems Based on Linear Codes. Technical report, Delft University of Technology.

[39] Gallian, J. (2009). Contemporary Abstract Algebra. Brooks Cole, 7 edition.

[40] Giorgetti, M. and Previtali, A. (2010). Galois invariance, trace codes and subfield subcodes. Finite Fields Appl., 16(2):96–99.

[41] Giusti, M. (1984). Some effectivity problems in polynomial ideal theory. In Proceedings of the International Symposium on Symbolic and Algebraic Computation, volume 174 of EUROSAM '84, pages 159–171, London, UK. Springer-Verlag.

[42] Guenda, K. and Gulliver, T. (2013). On the permutation groups of cyclic codes. Journal of Algebraic Combinatorics, 38:197–208.

[43] Hall, J. (2010). Notes on Coding Theory. lecture notes.

[44] Hankerson, D., Hoffman, D., Leonard, D., Lindner, C., Phelps, K., Rodger, C., and Wall, J. (2000). Coding Theory and Cryptography the Essentials. CRC Press, New York, second edition.

[45] Horn, R. A. (1986). Topics in Matrix Analysis. Cambridge University Press, New York, NY, USA.

[46] Huffman, W. and Pless, V. (2003). Fundamentals of Error-Correcting Codes. Cambridge University Press, Cambridge, first edition.

[47] Johansson, T. and Löndahl, C. (2011). An improvement to Sterns algorithm. Technical report, Lund University.

[48] Kaski, P. and Östergård, P. (2006). Classification Algorithms for Codes and Designs. Algorithms and Computation in Mathematics. Springer-Verlag, Netherland.

[49] Kipnis, A. and Shamir, A. (1999). Cryptanalysis of the HFE public key cryptosystem by Relinearization. In Advances in Cryptology CRYPTO 99, volume 1666 of Lecture Notes in Computer Science, pages 19–30. Springer.

[50] Köbler, J., Schöning, U., and Torán, J. (1993). The Graph Isomorphism Problem: Its Structural Complexity. Birkhauser Verlag, Basel, Switzerland, Switzerland.

[51] Lazard, D. (1983). Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In van Hulzen, J. A., editor, Computer Algebra, volume 162 of Lecture Notes in Computer Science, pages 146–156. Springer.

[52] Lee, P. and Brickell, E. (1988). An observation on the security of McElieces public-key cryptosystem. In Advances in Cryptology EUROCRYPT 88, volume 330 of Lecture Notes in Computer Science, pages 275–280. Springer.

[53] Leon, J. (1982). Computing automorphism groups of error-correcting codes. IEEE Transactions on Information Theory, 28:469–511.

[54] Leon, J. S. (1988). A probabilistic algorithm for computing minimum weights of large error-correcting codes. IEEE Transactions on Information Theory, 34(5):1354–1359.

[55] Leon, J. S. (1991). Permutation group algorithms based on partitions, I: Theory and algorithms. J. Symb. Comput., 12(4-5):533–583.

[56] Li, Y. X., Deng, R. H., and Wang, X. M. (2006). On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Trans. Inf. Theor., 40(1):271–273.

[57] Lidl, R., Niederreiter, H., and Cohn, P. (1997). Finite Fields. Cambridge University Press, second edition.

[58] Loidreau, P. and Sendrier, N. (2001). Weak keys in the McEliece public-key cryptosystem. IEEE Transactions on Information Theory, 47(3):1207–1211.

[59] López-Presa, J. L. and Fernández Anta, A. (2009). Fast algorithm for graph isomorphism testing. In Vahrenhold, J., editor, Experimental Algorithms: 8th International Symposium, SEA 2009, Dortmund, Germany, June 4-6, 2009, pages 221–232, Berlin, Heidelberg.

[60] MacWilliams, F. (1961). Error-correcting codes for multiple-level transmission. The Bell System Technical Journal, 40:281–308.

[61] MacWilliams, F. J. and Sloane, N. J. A. (1986). The Theory of Error-Correcting Codes. North–Holland, Amsterdam, fifth edition.

[62] Macwilliams, J. (1963). A theorem on the distribution of weights in a systematic code. Bell System Technical Journal, 42(1):79–94.

[63] Mayr, E. and Meyer, A. (1982). The complexity of the word problem for commutative semigroups and polynomial ideals. Adv. Math., 46:305–329.

[64] McEliece, R. (1978). A Public Key Cryptosystem Based on Algebraic Coding Theory. DSN Progress Report, 42–44:114–116.

[65] Miller, G. L. (1977). Graph isomorphism, general remarks. In Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, STOC '77, pages 143–150, New York, NY, USA. ACM.

[66] Mohamed, M. (2011). Improved Strategies for Solving Multivariate Polynomial Equation Systems over Finite Fields. PhD thesis, Technischen Universität Darmstadt genehmigte.

[67] Möller, H. M. and Mora, F. (1984). Upper and lower bounds for the degree of Groebner bases. In Proceedings of the International Symposium on Symbolic and Algebraic Computation, EUROSAM '84, pages 172–183, London, UK. Springer-Verlag.

[68] Niederreiter, H. (1986). Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory, 15:19–34.

[69] Pellikaan, R. Wu, X. and Bulygin, S. Jurrius, R. (2012). Error-correcting codes and cryptology. Book Preliminary Version.

[70] Petrank, E. and Roth, R. M. (1997). Is code equivalence easy to decide? IEEE Transactions on Information Theory, 43(5):1602–1604.

[71] Pless, V. (1998). Introduction to the Theory of Error-Correcting Codes. In Discrete Mathematics And Optimization. John Wiley & Sons, third edition.

[72] Purser, M. (1995). Introduction to Error-Correcting Codes. Artech House Telecommunications Library. Artech House Publishers, Boston.

[73] Roth, R. (2006). Introduction to Coding Theory. Cambridge University Press.

[74] Sendrier, N. (1995). On the Dimension of the Hull. Technical Report RR-2682, INRIA.

[75] Sendrier, N. (1999). The Support Splitting Algorithm. Technical Report RR-3637, INRIA.

[76] Sendrier, N. (2000). Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Transactions on Information Theory, 46(4):1193–1203.

[77] Sendrier, N. and Simos, D. (2013a). The Hardness of code equivalence over $\mathbb{F}_q$ and its application to code-based cryptography. In Post-Quantum Cryptography, Lecture Notes in Computer Science, pages 203–216, Limoges, France. Springer.

[78] Sendrier, N. and Simos, D. E. (2013b). How easy is code equivalence over Fq? In International Workshop on Coding and Cryptography - WCC 2013, Bergen, Norway.

[79] Sendrier, N. and Torres, R. (2016). Analysis of Information Set Decoding for a Sub-linear Error Weight. In Post Quantum Cryptography, Lecture Notes in Computer Science, pages 144–161. Springer.

[80] Skersys, G. (2003). The average dimension of the hull of cyclic codes. Discrete Applied Mathematics, 128:275–292.

[81] Stegers, T. (2007). Faugères F5 Algorithm Revisited. Master's thesis, Department Of Mathematics, Technische Universität Darmstadt.

[82] Stern, J. (1988). A method for finding codewords of small weight. In Cohen, G. D. and Wolfmann, J., editors, Coding Theory and Applications, volume 388 of Lecture Notes in Computer Science, pages 106–113. Springer.

[83] Stichtenoth, H. (1990). On the dimension of subfield subcodes. IEEE Transactions on Information Theory, 36:90–93.

[84] Stichtenoth, H. (2009). Algebraic Function Fields and Codes. Graduate Texts in Mathematics. Springer-Verlag, Stanbul, Turkey, second edition.

[85] Storjohann, A. (2000). Algorithms for Matrix Canonical Forms. Phd thesis, Department of Computer Science, ETH, Zürich. Available from `http://www.scg.uwaterloo.ca/~astorjoh/publications.html`.

[86] Thomae, E. and Wolf, C. (2010). Solving systems of multivariate quadratic equations over finite fields or: From Relinearization to MutantXL.

[87] Tilborg, H. (1993). Coding Theory a First Course. Preliminary Textbook.

[88] Trappe, W. and Washington, L. (2006). Introduction to Cryptography with Coding Theory. Pearson Education International, second edition.

[89] Vanstone, S. and van Oorschot, P. (1989). An Introduction to Error Correcting Codes with Applications. Kluwer Academic Publishers, first edition.

[90] Xambo-Descamps, S. (2003). Block Error-Correcting Codes, A Computational Primer. Springer, Universitat Politcnica de Catalunya, Barcelona.

Appendix A

Coding Theory

This chapter introduces the basic definitions and background needed in the rest of the thesis regarding coding theory and code-based cryptography. We define linear codes and discuss some of their properties and parameters. We give the definition of weight enumerators and introduce MacWilliams identity that relates the weight enumerators of the code and its dual. There are many classes of codes that can be constructed from a given code such as punctured codes subfield subcodes, trace codes ... etc. These classes are discussed in Section A.3. We define the encoding and decoding operations since they are the corner stones of coding theory and code-based cryptography. Important families of linear codes are introduced and discussed in Section A.5. In the last section we introduce code-based cryptography and discuss briefly the original McEliece and Niederreiter cryptosystems. We see how code-based cryptography relies on some `NP-hard` problem, thus it will be resistant to quantum computers.

A.1 Bounds on the Parameters

In this section we give the important bounds introduced on the parameters of linear codes.

A.1.1 The Singleton Bound

Theorem A.1.1 (Singleton Bound)**.** *[73] For any $[n, k, d]$ linear code over $\mathbb{F}_q$ we have $d \leqslant n - k + 1$.*

An $[n, k, d]$ linear code is called *Maximum Distance Separable*, MDS, if it satisfies $d = n - k + 1$.

A.1.2 The Sphere-packing Bound (Hamming Bound)

Let $\boldsymbol{w}$ be a codeword of an $[n, k, d]$ code defined over $\mathbb{F}_q$. A sphere of radius t and center $\boldsymbol{w}$ in $\mathbb{F}_q^n$ is the set of words in $\mathbb{F}_q^n$ at Hamming distance t or less from $\boldsymbol{w}$. The number of words in this sphere is given by:

$$V_q(n, t) = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i$$

Theorem A.1.2 (Sphere-packing Bound)**.** *[73] For any $[n, k, d]$ linear code over $\mathbb{F}_q$ we have:*

$$V_q(n, \lfloor \tfrac{d-1}{2} \rfloor) \leqslant q^{n-k}$$

An $[n, k, d]$ linear code is called perfect if it satisfies the upper limit of the inequality, i.e $V_q(n, \lfloor \frac{d-1}{2} \rfloor) = q^{n-k}$.

A.1.3 The Gilbert-Varshamov Bound

Theorem A.1.3 (Gilbert-Varshamov Bound)**.** *[73] Consider the field $\mathbb{F}_q$ and the positive integers n, k and d that satisfy*

$$V_q(n-1, d-2) < q^{n-k}$$

then there exists an $[n, k]$ linear code over $\mathbb{F}_q$ with minimum distance at least d.

It has been proven that most of the linear codes satisfy Gilbert-Varshamov bound [73].

A.2 Weight Enumerators

In this section we define weight enumerators and state the important MacWilliams Identity that relates the weight enumerators of the code and its dual.

Definition A.2.1. Let $\mathscr{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_q$, we define

$$A_i(\mathscr{C}) = \#\{\boldsymbol{c} \in \mathscr{C} : wt(\boldsymbol{c}) = i\}$$

The list $[A_i(\mathscr{C})]_{i \in \{0,\dots,n\}}$ is called *weight distribution* of $\mathscr{C}$.

One can easily see that $\sum_{i=0}^{n} A_i(\mathscr{C}) = q^k$ and $A_i(\mathscr{C}) = 0$ for $1 \leqslant i \leqslant d-1$.

The *weight enumerator* of $\mathscr{C}$ is:

$$\mathcal{W}_{\mathscr{C}}(x) = \sum_{i=0}^{n} A_i(\mathscr{C}) x^i$$

and the *homogeneous weight enumerator* is:

$$\mathcal{W}_{\mathscr{C}}(x, y) = \sum_{i=0}^{n} A_i(\mathscr{C}) x^i y^{n-i}$$

A.2.1 MacWilliams Identity

Knowing the weight distribution of a linear code $\mathscr{C}$, MacWilliams identity enables to determine the weight distribution of its dual code.

Theorem A.2.2 (MacWilliams Identity)**.** *[62] Let $\mathscr{C}$ be a linear code over $\mathbb{F}_q$ and $\mathscr{C}^\perp$ is its dual code then their weight enumerators satisfy the following relation*

$$\mathcal{W}_{\mathscr{C}^\perp}(x,y) = \frac{1}{|\mathscr{C}|}\mathcal{W}_{\mathscr{C}}(x+(q-1)y, x-y)$$

If the code is self-dual then the weight enumerator is invariant under the MacWilliams identity.

A.3 Modifying Codes

Given $\mathscr{C}$ an $[n,k,d]$ linear code we show how to construct other codes from $\mathscr{C}$. The resulting codes sometimes are of great importance and appear in many places throughout the thesis such as punctured codes, square codes, subfield subcodes and trace codes.

A.3.1 Punctured Codes

Let $\mathscr{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. We puncture $\mathscr{C}$ in the coordinate i by deleting the coordinate i in all codewords of $\mathscr{C}$. The resulting code is called the *punctured code* of $\mathscr{C}$ at coordinate i and denoted $\mathscr{C}_i$. The generator matrix of $\mathscr{C}_i$ is obtained from the generator matrix of $\mathscr{C}$ by deleting column i. Puncturing can be done in a set of coordinates $\{i_1, ..., i_t\}$ with $t < n$.

Sometimes puncturing is done in such a way that we preserve the length of the code. Thus we replace coordinate i by zero in all codewords instead of deleting the coordinate.

The following theorem gives the dimension and minimum distance of the punctured code.

Theorem A.3.1. *[46] Let $\mathscr{C}$ be an $[n,k,d]$ linear code over $\mathbb{F}_q$ and let $\mathscr{C}_i$ be the punctured code at coordinate i then we have two cases*

1. *if $d > 1$, $\mathscr{C}_i$ is an $[n-1,k,d_i]$ linear code where $d_i = d-1$ if $\mathscr{C}$ has a minimum weight codeword with a nonzero ith coordinate otherwise $d_i = d$.*
2. *if $d = 1$, $\mathscr{C}_i$ is an $[n-1,k,1]$ linear code if $\mathscr{C}$ has no codeword of weight 1 whose nonzero entry is in coordinate i otherwise, if $k > 1$, $\mathscr{C}_i$ is an $[n-1,k-1,d_i]$ code with $d_i \geqslant 1$.*

A.3.2 Shortened Codes

Let $\mathscr{C}$ be an $[n,k,d]$ linear code and consider the set of coordinates $\mathcal{T}$ of t elements and select all the codewords of $\mathscr{C}$ that have 0 in the coordinates of $\mathcal{T}$, this set is a subcode of $\mathscr{C}$. Puncturing this subcode on $\mathcal{T}$ will result in the *shortened code* $\mathcal{S}_{\mathcal{T}}(\mathscr{C})$. The length of $\mathcal{S}_{\mathcal{T}}(\mathscr{C})$ is $n-t$.

There is a strong connection between the punctured and shortened codes that is captured by the following theorem [46].

Theorem A.3.2. *Let $\mathscr{C}$ be an $[n,k,d]$ linear code over $\mathbb{F}_q$ and $\mathcal{T}$ a set of coordinates then we have*

1. *$\mathcal{S}_{\mathcal{T}}\left(\mathscr{C}^\perp\right) = (\mathscr{C}_{\mathcal{T}})^\perp$ and $(\mathscr{C}^\perp)_{\mathcal{T}} = (\mathcal{S}_{\mathcal{T}}(\mathscr{C}))^\perp$.*
2. *If $t < d$, $\mathscr{C}_{\mathcal{T}}$ has dimension k and $(\mathscr{C}_{\mathcal{T}})^\perp$ has dimension $n-t-k$.*

3. *If $t = d$ and $\mathcal{T}$ is the set of coordinates where a minimum weight codeword is nonzero, then the dimension of $\mathscr{C}_{\mathcal{T}}$ is $k-1$ and the dimension of $(\mathscr{C}_{\mathcal{T}})^{\perp}$ is $n-d-k+1$.*

A.3.3 Extended Codes

The linear code $\mathscr{C}$ can be extended to other linear codes of bigger length by adding new coordinates. There are many ways to do that but the most common way is to add new coordinate in such a way all codewords sum up to zero. We denote the extended code by $\widehat{\mathscr{C}}$.

$$\widehat{\mathscr{C}} = \{(c_1, ..., c_{n+1}) : (c_1, ..., c_n) \in \mathscr{C}, \sum_{i=1}^{n+1} c_i = 0\}$$

The extended code $\widehat{\mathscr{C}}$ has parameters $[n+1, k, \widehat{d}]$ where $\widehat{d} = d$ or $d+1$. The generator matrix of $\widehat{\mathscr{C}}$ can be obtained from the generator matrix of $\mathscr{C}$ by adding extra column to the end so that the sum of each row is zero. and the parity check matrix $\widehat{\boldsymbol{H}}$ will be of the form:

$$\left(\begin{array}{ccc|c} 1 & \dots & 1 & 1 \\ \hline & & & 0 \\ & \boldsymbol{H} & & 0 \\ & & & 0 \end{array}\right)$$

A.3.4 The $(\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v})$ Construction

Let $\mathscr{C}_i$ be $[n, k_i, d_i]$ linear codes of the same length over $\mathbb{F}_q$, $i \in \{1, 2\}$, $(\boldsymbol{u}|\boldsymbol{u}+\boldsymbol{v})$ construction will give the linear code

$$\mathscr{C} = \{(\boldsymbol{u}, \boldsymbol{u}+\boldsymbol{v}) : \boldsymbol{u} \in \mathscr{C}_1, \boldsymbol{v} \in \mathscr{C}_2\}$$

The code $\mathscr{C}$ is $[2n, k_1+k_2, min\{2d_1, d_2\}]$ linear code. Its generator and check matrices are

$$\begin{pmatrix} \boldsymbol{G}_1 & \boldsymbol{G}_1 \\ \boldsymbol{0} & \boldsymbol{G}_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \boldsymbol{H}_1 & \boldsymbol{0} \\ -\boldsymbol{H}_2 & \boldsymbol{H}_2 \end{pmatrix}$$

A.3.5 Subfield Subcodes and Trace Codes

Here we define the subfield subcodes and trace codes and give the relation connecting the two codes introduced by Delsarte in [26] and we give some bounds on their dimensions.

Definition A.3.3. Let $\mathscr{C}$ be an $[n, k]$ linear code over a finite field $\mathbb{F}$ and $\mathbb{K} \subseteq \mathbb{F}$, we define the *subfield subcode* $\mathscr{C}|_{\mathbb{K}}$ to be $\mathscr{C} \cap \mathbb{K}^n$.

Definition A.3.4. Let $\mathbb{F} = \mathbb{F}_{q^m}$ be finite field and $\mathbb{K} = \mathbb{F}_q$, we define the trace map $Tr_{\mathbb{F}/\mathbb{K}} : \mathbb{F} \longrightarrow \mathbb{K}$ to be $Tr_{\mathbb{F}/\mathbb{K}}(\alpha) = \alpha + \alpha^q + \ldots + \alpha^{q^{m-1}}$, for $\alpha \in \mathbb{F}$. The trace of a word $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}^n$ is $Tr_{\mathbb{F}/\mathbb{K}}(\boldsymbol{c}) = (Tr_{\mathbb{F}/\mathbb{K}}(c_1), \ldots, Tr_{\mathbb{F}/\mathbb{K}}(c_n))$. We define the trace code of $\mathscr{C}$, $Tr_{\mathbb{F}/\mathbb{K}}(\mathscr{C}) = \{Tr_{\mathbb{F}/\mathbb{K}}(\boldsymbol{c}) : \boldsymbol{c} \in \mathscr{C}\}$. When it is clear from the context we simply write $Tr(.)$.

Subfield subcodes and trace codes are tightly related by the theorem of Delsarte [26].

Theorem A.3.5 (Delsarte)**.**

$$(\mathscr{C}|_{\mathbb{K}})^{\perp} = Tr_{\mathbb{F}/\mathbb{K}}(\mathscr{C}^{\perp})$$

The subfield subcode and trace code have the same length as the original code. If $\mathbb{F} = \mathbb{F}_{q^m}$ and $\mathbb{K} = \mathbb{F}_q$ then the dimension of the subfield subcode, $\dim(\mathscr{C}|_{\mathbb{F}_q})$, satisfies [84]

$$n - m(n-k) \leqslant \dim(\mathscr{C}|_{\mathbb{F}_q}) \leqslant k$$

and the dimension of the trace code satisfies [84]

$$k \leqslant \dim(Tr_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C})) \leqslant mk$$

A.3.6 Schur Product Code

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, ..., y_n)$ from $\mathbb{F}^n$ then we define *Schur Product* of $\boldsymbol{x}$ and $\boldsymbol{y}$ as follows

$$\boldsymbol{x} * \boldsymbol{y} = (x_1 y_1, \ldots, x_n y_n) \in \mathbb{F}^n$$

It is also called component-wise product.

Let $\mathscr{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}$, we define *Schur product code* $\mathscr{C} * \mathscr{C}$ to be the linear span of

$$\{\boldsymbol{c} * \boldsymbol{c}' : \boldsymbol{c}, \boldsymbol{c}' \in \mathscr{C}\}$$

It is often denoted $\mathscr{C}^2$ and called *the square code*. Clearly the code $\mathscr{C}^2$ has length n. Informally if we look at the injective map $\boldsymbol{c} \mapsto \boldsymbol{c} * \boldsymbol{c}$, we can assume we have a copy of $\mathscr{C}$ embedded in $\mathscr{C}^2$ thus $k' \geqslant k$, where k' is the dimension of $\mathscr{C}^2$ and $d' \leqslant d$, d' is the minimum distance of $\mathscr{C}^2$. By induction, $\mathscr{C}^{i+1} = \mathscr{C}^i * \mathscr{C}, \mathscr{C}^1 = \mathscr{C}$.

A.4 Encoding and Decoding

Let $\mathscr{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_q$ with generator matrix $\boldsymbol{G}$, we can *encode* any message $\boldsymbol{m} \in \mathbb{F}_q^k$ by the following mapping: $\boldsymbol{m} \mapsto \boldsymbol{m}\boldsymbol{G} = \boldsymbol{c}$. This mapping is one-to-one correspondence since $\boldsymbol{G}$ is of rank k. If $\boldsymbol{G}$ is in the standard form, $\boldsymbol{m}$ will allocate the first k coordinates of the codeword $\boldsymbol{c}$.

When a message is sent through a noisy channel an error $\boldsymbol{e}$ is expected, thus the received word $\boldsymbol{w} \in \mathbb{F}_q^n$ is different from the codeword $\boldsymbol{c}$. When $\boldsymbol{w}$ is received, the decoding process is to determine the corresponding codeword $\boldsymbol{c}$, i.e to remove the error from the received word.

The known decoding algorithms that work for all linear codes have exponential time complexity. Thus although the encoding is very simple the decoding in general is not easy. Hence the decoding can be defined as follows:

Given a received word $\boldsymbol{w} \in \mathbb{F}_q^n$, the problem is to find the codeword $\boldsymbol{c} \in \mathscr{C}$ such that $d(\boldsymbol{w}, \boldsymbol{c})$ is minimal. Or equivalently, find an error vector of minimum weight $\boldsymbol{e} \in \mathbb{F}_q^n$ such that $\boldsymbol{w} - \boldsymbol{e} \in \mathscr{C}$.

A.4.1 Syndrome Decoding

Let $\mathscr{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_q^n$ with parity check matrix $\boldsymbol{H}$. We define the *syndrome* of the word $\boldsymbol{w}$ to be $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{w}^T$. Note that $\boldsymbol{w} = \boldsymbol{c} + \boldsymbol{e}$ thus $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{c}^T + \boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{H}\boldsymbol{e}^T$. The codewords of $\mathscr{C}$ are exactly the words of syndrome $\mathbf{0}$ since $\boldsymbol{H}\boldsymbol{c}^T = \mathbf{0}$ if and only if $\boldsymbol{c} \in \mathscr{C}$ and two words in $\mathbb{F}_q^n$ have the same syndrome if and only if their difference is in $\mathscr{C}$. Thus the syndrome decoding performs the following two steps:

1. Compute the syndrome of the received word, $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{w}^T$,
2. Find a minimum-weight word $\boldsymbol{e} \in \mathbb{F}_q^n$ such that $\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s}$.

The difficulty of decoding is in the second step which is known to be `NP-hard` problem for general matrices $\boldsymbol{H}$ and $\boldsymbol{s}$ [73], since it corresponds to finding the minimum set of columns of $\boldsymbol{H}$ such that their span contains $\boldsymbol{s}$. For the codes with a specific structure in their parity check matrix there may be efficient algorithm to find the syndrome and this is the case for most of the codes used in practice.

A.5 Important Classes of Linear Codes

In this section we introduce important types of codes that are extensively used in practice, that is due to their structures and existence of fast decoding algorithm.

A.5.1 Cyclic Codes

Let $\mathscr{C}$ be a linear code of length n, $\mathscr{C}$ is said to be *cyclic* if for each codewords $\boldsymbol{c} = (c_0, ..., c_{n-1}) \in \mathscr{C}$ the word $(c_{n-1}, c_1, ..., c_{n-2})$ is also in $\mathscr{C}$. Here we prefer to use index from 0 to $n-1$ to be able to use the $\mod$ operation on the indexes. Another reason is that sometimes we need to use polynomial expression to express the cyclic codes. In fact we have the bijection $\psi : \mathbb{F}_q^n \to \mathbb{F}_q[x]/(x^n - 1)$ defined by:

$$(c_0, c_1, ..., c_{n-1}) \leftrightarrow c_0 + c_1 x + ... + c_{n-1}x^{n-1} = c(x)$$

shifting a codeword $\boldsymbol{c}$ by one position corresponds to multiplying $c(x)$ by x in the polynomial ring $\mathbb{F}_q[x]/(x^n - 1)$, thus

$$xc(x) = c_0 x + c_1 x^2 + ... + c_{n-1}x^n \mod (x^n - 1) = c_{n-1} + c_0 x + c_1 x^2 + ... + c_{n-2}x^{n-1}$$

The shift is not necessarily by one position but it can also by many positions this corresponds to multiplying by a power of x or a polynomial in $\mathbb{F}_q[x]/(x^n - 1)$.

The following theorem states important properties of the cyclic codes.

Theorem A.5.1. *[87] Let $\mathscr{C}$ be an $[n, k]$ cyclic code over $\mathbb{F}_q$,*

1. *There exists a unique monic polynomial $g(x)$ over $\mathbb{F}_q[x]$ that divides $(x^n - 1)$ such that $\mathbf{c} \in \mathscr{C}$ if and only if $g(x)$ divides $c(x)$. The polynomial $g(x)$ is called the generator polynomial of $\mathscr{C}$. The degree of $g(x)$ is $n - k$.*

2. *If we write the generator polynomial* $g(x) = g_0 + g_1x + ... + g_{n-k}x^{n-k}$*, the generator matrix of* $\mathscr{C}$ *is*

$$\boldsymbol{G} = \begin{pmatrix} g_0 & g_1 & \dots & g_{n-k} & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{n-k} & 0 & \dots & 0 \\ 0 & 0 & g_0 & g_1 & \dots & g_{n-k} & \dots & 0 \\ \vdots & & & \ddots & \ddots & & \ddots & \\ 0 & 0 & \dots & 0 & g_0 & g_1 & \dots & g_{n-k} \end{pmatrix}$$

3. *If* $h(x) = (x^n - 1)/g(x)$ *then* $\mathbf{c} \in \mathscr{C}$ *if and only if* $c(x)h(x) = 0$ *in* $\mathbb{F}_q[x]/(x^n - 1)$*. The polynomial* $h(x)$ *is called parity check polynomial of* $\mathscr{C}$ *and has degree* k*. The corresponding parity check matrix is given by*

$$\boldsymbol{H} = \begin{pmatrix} 0 & \dots & 0 & 0 & h_k & \dots & h_1 & h_0 \\ 0 & \dots & 0 & h_k & \dots & h_1 & h_0 & 0 \\ \vdots & \iddots & & & \iddots & & \iddots & \\ h_k & \dots & h_1 & h_0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

4. *The matrix* $\boldsymbol{H}$ *generates* $\mathscr{C}^{\perp}$ *which is in turn cyclic code and its generator polynomial is* $\frac{1}{h_0}(h_k + h_{k-1}x + ... + h_1x^{k-1} + h_0x^k) = \frac{1}{h_0}x^k h(1/x)$.

A.5.2 Generalized Reed-Solomon Codes (GRS)

Let $\alpha_1, \dots, \alpha_n$ be distinct nonzero elements from $\mathbb{F}_q$ and $v_1, \dots, v_n$ nonzero elements from $\mathbb{F}_q$. We define the *GRS* code to be the $[n, k, d]$ linear code over $\mathbb{F}_q$ with parity check matrix

$$\boldsymbol{H}_{GRS} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \dots & \alpha_n^{n-k-1} \end{pmatrix} \begin{pmatrix} v_1 & & & \\ & v_2 & & 0 \\ & & \ddots & \\ 0 & & & v_n \end{pmatrix}$$

As one can notice by definition GRS length cannot exceed $q - 1$ since $\alpha_i, 0 \leqslant i \leqslant n$, must be distinct nonzero elements from $\mathbb{F}_q$. The elements α_i are called *code locators* and v_i are called *column multipliers*. The following proposition gives some of the properties of GRS codes.

Proposition A.5.2. *[73] Let* $\mathscr{C}$ *be an* $[n, k, d]$ *GRS code with parity check matrix as defined above.*

1. *The code* $\mathscr{C}$ *is an MDS code, i.e* $d = n - k + 1$.
2. *The generator matrix of* $\mathscr{C}$ *is*

$$\boldsymbol{G}_{GRS} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} v'_1 & & & \\ & v'_2 & & 0 \\ & & \ddots & \\ 0 & & & v'_n \end{pmatrix}$$

That means the dual code $\mathscr{C}^{\perp}$ is an $[n, n-k]$ GRS code that can be defined through the same code locators.

As in the cyclic code we can use the polynomial notation to express the GRS codes. For every word $\boldsymbol{u} = (u_0, \ldots, u_{k-1}) \in \mathbb{F}_q^k$ we associate a polynomial $u(x) = u_0 + u_1 x + \ldots + u_{k-1}x^{k-1} \in \mathbb{F}_q[x]/(x^k-1)$. The encoded word of $\boldsymbol{u}$ is $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}_{GRS}$ and in terms of polynomials $\boldsymbol{c} = (v'_1 u(\alpha_1), v'_2 u(\alpha_2), \ldots, v'_n u(\alpha$ where coordinate i is related to the evaluation of $u(x)$ at α_i, $1 \leqslant i \leqslant n$.

A.5.3 Reed-Solomon Codes (RS)

Reed-Solomon codes are special case of GRS codes. Let n be a positive integer dividing $q-1$ and choose $\alpha \in \mathbb{F}_q$ such that $\alpha^n = 1$ and let b be an integer. The $[n, k, d]$ RS code, $\mathscr{C}$, is a GRS code with code locators $\alpha_i = \alpha^{i-1}$ and column multipliers $v_i = \alpha^{b(i-1)}$ for $1 \leqslant i \leqslant n$. Thus the parity check matrix will be

$$\boldsymbol{H}_{RS} = \begin{pmatrix} 1 & \alpha^b & \ldots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \ldots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{b+d-2} & \ldots & \alpha^{(n-1)(b+d-2)} \end{pmatrix}$$

Note that $d = n - k + 1$. If we use the polynomial notation as defined before, for $\boldsymbol{c} \in \mathscr{C}$ we have $\boldsymbol{c} \mapsto c(x)$. Thus $\boldsymbol{c} \in \mathscr{C}$ if and only if $\boldsymbol{H}_{RS}\boldsymbol{c}^T = \boldsymbol{0}$ implies that $c(\alpha^i) = 0$ for $b \leqslant i \leqslant b + d - 2$. These elements are called the roots of the RS code. Thus the word $\boldsymbol{c}$ belongs to $\mathscr{C}$ if and only if all roots of $\mathscr{C}$ are roots of $c(x)$.

We define the *generator polynomial* to be $g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \ldots (x - \alpha^{b+d-2})$. Thus the word $\boldsymbol{c} \in \mathscr{C}$ if and only if $g(x)$ divides $c(x)$. From this $\mathscr{C}$ can be redefined in terms of polynomial notation as follows

$$\mathscr{C} = \{u(x)g(x) : u(x) \in \mathbb{F}_q[x]/(x^k - 1)\}$$

Note that the elements of $\mathscr{C}$ are in $\mathbb{F}_q[x]/(x^n - 1)$ since the degree of $g(x)$ is $d - 1 = n - k$ and also $g(x)$ divides $x^n - 1$. Thus one can notice that RS code is cyclic code.

A.5.4 Reed-Muller Codes (RM)

The r-th Reed-Muller code over $\mathbb{F}_2$ of length $n = 2^m$, denoted $RM(r, m)$, is defined by

$$RM(r, m) = \{f(\boldsymbol{u}_0), f(\boldsymbol{u}_1), \ldots, f(\boldsymbol{u}_{n-1}) : degree(f) \leqslant r\},$$

where $\boldsymbol{u}_i, 0 \leqslant i \leqslant n-1$ ranges over all binary vectors of length m and f is a function of binary coefficients in m variables and degree at most r.Note that over $\mathbb{F}_2$, each variable $x_i, 1 \leqslant i \leqslant m$ satisfies $x_i^2 = x_i$.

By definition RM codes satisfy the inclusion $R(0, m) \subset R(1, m) \subset \ldots \subset R(m, m)$. The dimension of $RM(r, m)$ is $\sum_{i=0}^{r} \binom{m}{i}$ and the minimum distance is 2^{m-r}. The dual code of $RM(r, m) = RM(m - r - 1, m)$ [61].

A.5.5 Goppa Codes

Goppa codes were introduced by V. D. Goppa in 1970 using extensive results from algebraic geometry [46]. Goppa codes are one of the important codes that are extensively used in code-based cryptography especially the irreducible binary Goppa code. Binary Goppa codes were suggested to be used by McEliece in his first code-based cryptosystem and they proved to be secured in practice. Although a lot of research has been invested in breaking McEliece cryptosystem based on Goppa codes there were no successful attack to fully break the system. Binary Goppa codes seem to be interesting in cryptography for the following reasons [29]:

1. The lower bound of the minimum distance is easy to compute.
2. Knowing the generating polynomial allows for efficient decoding.
3. No efficient decoding method is known without the knowledge of the generating polynomial.

Definition A.5.3. [29] Let m and t be positive integers and

$$g(x) = \sum_{i=0}^{t} g_i x^i \in \mathbb{F}_{2^m}[x]$$

be a monic polynomial of degree t called *Goppa polynomial* and $L = (\gamma_0, \gamma_1, \ldots, \gamma_{n-1}) \in \mathbb{F}_{2^m}^n$ a tuple of n distinct elements are called code support such that

$$g(\gamma_i) \neq 0, \quad 0 \leqslant i \leqslant n-1$$

The *binary Goppa code* $\Gamma(L, g(x))$ over $\mathbb{F}_2$ is defined by

$$\Gamma(L, g(x)) = \{\boldsymbol{c} \in \mathbb{F}_2^n : \sum_{i=0}^{n-1} \frac{c_i}{x - \gamma_i} \equiv 0 \mod g(x)\}$$

We define the *syndrome* of the word $\boldsymbol{c}$ by

$$S_c(x) = -\sum_{i=0}^{n-1} \frac{c_i}{g(\gamma_i)} \frac{g(x) - g(\gamma_i)}{x - \gamma_i} \mod g(x)$$

Thus $\boldsymbol{c} \in \Gamma(L, g(x))$ if and only if $S_c(x) \equiv \sum_{i=0}^{n-1} \frac{c_i}{x-\gamma_i} \equiv 0 \mod g(x)$. The code $\Gamma(L, g(x))$ is called *irreducible binary Goppa code* if $g(x)$ is irreducible over $\mathbb{F}_{2^m}$.

The parity check matrix of $\Gamma(L, g(x))$ can be decomposed into three matrices $\boldsymbol{H} = \boldsymbol{XYZ}$ where

$$\boldsymbol{X} = \begin{pmatrix} g_t & & & 0 \\ g_{t-1} & g_t & & \\ \vdots & \vdots & \vdots & \ddots \\ g_1 & g_2 & g_3 & \cdots & g_t \end{pmatrix} \qquad \boldsymbol{Y} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \gamma_0 & \gamma_1 & \cdots & \gamma_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_0^{t-1} & \gamma_1^{t-1} & \cdots & \gamma_{n-1}^{t-1} \end{pmatrix}$$

$$\boldsymbol{Z} = \begin{pmatrix} \frac{1}{g(\gamma_0)} & & & \\ & \frac{1}{g(\gamma_1)} & & 0 \\ & & \ddots & \\ 0 & & & \frac{1}{g(\gamma_{n-1})} \end{pmatrix}$$

Hence $\boldsymbol{c} \in \Gamma(L, g(x))$ if and only if $\boldsymbol{XYZc}^T = \boldsymbol{0}$. The matrix $\boldsymbol{H}$ has dimensions $t \times n$ and the elements of $\boldsymbol{H}$ are from $\mathbb{F}_{2^m}$. Thus we can decompose the elements to be in the base field by considering $\mathbb{F}_{2^m}$ an m dimensional vector space over $\mathbb{F}_2$ thus $\boldsymbol{H}$ will be over $\mathbb{F}_2$ with dimension $mt \times n$. Goppa code $\Gamma(L, g(x))$ has dimension k with $k \geqslant n - mt$. The minimum distance of $\Gamma(L, g(x))$ is at least $2t + 1$ thus it can always correct up to t errors. When $g(x)$ is irreducible over $\mathbb{F}_{2^m}$ that means it has no root in $\mathbb{F}_{2^m}$ therefore the elements of L can range over all the elements of $\mathbb{F}_{2^m}$. Note that Goppa codes can also be defined over arbitrary field.

A.6 Code-based Cryptography

Code-based cryptography is the branch of cryptography that uses the notions of coding theory to design cryptosystem. Code-based cryptography is one of the strong nominated successors of the number theoretic based cryptosystem. Research in quantum computer is rapidly progressing and with appearance of quantum computers most of the currently used cryptosystem will turn to be insecure. Thus it is very crucial to develop a solid and concrete theoretical and practical aspects of an alternative cryptosystems that remain secure.

In code-based cryptography, the first developed cryptosystem is McEliece public key cryptosystem which is introduced by Robert McEliece in 1978 [64]. McEliece original cryptosystem based on binary Goppa code remains unbroken up to date. In spite of existence of serious tries to weaken or degrade the security of this cryptosystem, see for example [36], [13], and [58], it remains secure. Harald Niederreiter in 1986 in [68] suggested another variant of McEliece cryptosystem that uses the parity check matrix instead of generator matrix which is equivalent to McEliece cryptosystem in terms of security.

Next we briefly present McEliece and Niederreiter cryptosystems.

A.6.1 McEliece Cryptosystem

Let $n, t \in \mathbb{N}$ be public parameters with $t \ll n$. McEliece Cryptosystem is composed of three parts: Key generation, Encryption and Decryption.

Key Generation

- Generate the matrices $\boldsymbol{G}'$, $\boldsymbol{S}$ and $\boldsymbol{P}$ where $\boldsymbol{G}'$ is a $k \times n$ generator matrix of an $[n, k, \geqslant 2t + 1]$ irreducible binary Goppa code Γ and k is about $n/2$. The matrix $\boldsymbol{S}$ is a $k \times k$ random invertible matrix, and $\boldsymbol{P}$ is an $n \times n$ random permutation matrix.
- Compute $\boldsymbol{G} = \boldsymbol{SG}'\boldsymbol{P}$.

The public key is $(\boldsymbol{G}, t)$ and the private key is $(\boldsymbol{S}, D_\Gamma, \boldsymbol{P})$ where D_Γ is an efficient decoding algorithm of Γ.

Encryption

Let $\boldsymbol{m} \in \mathbb{F}_2^k$ and $\boldsymbol{e} \in \mathbb{F}_2^n$ is random of weight t, the ciphertext $\boldsymbol{c} = \boldsymbol{mG} + \boldsymbol{e}$.

Decryption

When $\boldsymbol{c}$ is received, the decryption process is performed as follows:

- $\boldsymbol{cP}^{-1} = \boldsymbol{mSG'} + \boldsymbol{eP}^{-1}$.
- Apply the efficient decoding algorithm D_Γ, thus $D_\Gamma(\boldsymbol{cP}^{-1}) = \boldsymbol{mSG'}$ since the word $\boldsymbol{eP}^{-1}$ is of weight t and can be corrected.
- Select a set $I \subseteq \{1, \ldots, n\}$ of k columns index such that the matrix $\boldsymbol{G}'_I$, that results from $\boldsymbol{G'}$ by restriction to columns indexed by I, is invertible.
- Compute the plaintext, $\boldsymbol{m} = (\boldsymbol{mSG'})_I(\boldsymbol{G}'_I)^{-1}\boldsymbol{S}^{-1}$

Note that, McEliece cryptosystem can be defined for other families of codes other than Goppa codes and arbitrary field according to the definition of the code. McEliece original parameters is to use $[1024, 524, 101]$ irreducible binary Goppa codes, thus $t = 50$ (security level 60 bits).

A.6.2 Niederreiter Cryptosystem

Niederreiter cryptosystem is a variant of McEliece that uses the dual code for the encryption process. The security of Niederreiter cryptosystem is proved to be equivalent to McEliece [56] but in practice Niederreiter is much faster than McEliece. Niederreiter cryptosystem is described as follows

As in McEliece cryptosystem, let $n, t \in \mathbb{N}$ be public parameters with $t \ll n$. Niederreiter cryptosystem is composed of three parts: Key generation, Encryption and decryption.

Key Generation

- Generate the matrices $\boldsymbol{H'}, \boldsymbol{T}$ and $\boldsymbol{P}$ where $\boldsymbol{H'}$ is an $(n-k) \times n$ parity check matrix of an $[n, k, \geqslant 2t+1]$ irreducible binary Goppa code Γ and k is about $n/2$. The matrix $\boldsymbol{T}$ is an $(n-k) \times (n-k)$ random invertible matrix, and $\boldsymbol{P}$ is an $n \times n$ random permutation matrix.
- Compute $\boldsymbol{H} = \boldsymbol{TH'P}$.

The public key is $(\boldsymbol{H}, t)$ and the private key is $(\boldsymbol{T}, D_\Gamma, \boldsymbol{P})$ where D_Γ is an efficient syndrome decoding algorithm of Γ.

Encryption

Let $\boldsymbol{m} \in \mathbb{F}_2^k$ be the original message, we represent $\boldsymbol{m}$ by a word $\boldsymbol{e} \in \mathbb{F}_2^n$ of weight t and is called plaintext. Find the ciphertext by computing the syndrome of $\boldsymbol{e}$, $\boldsymbol{s} = \boldsymbol{He}^T$.

Decryption

When $\boldsymbol{s}$ is received, the decryption process is performed as follows:

- $\boldsymbol{T}^{-1}\boldsymbol{s} = \boldsymbol{H}'\boldsymbol{P}\boldsymbol{e}^T$.
- Apply the efficient syndrome decoding algorithm D_Γ to get $\boldsymbol{P}\boldsymbol{e}^T$, thus $D_\Gamma(\boldsymbol{T}^{-1}\boldsymbol{s}) = \boldsymbol{P}\boldsymbol{e}^T$ since the word $\boldsymbol{P}\boldsymbol{e}^T$ is of weight t.
- Compute the plaintext, $\boldsymbol{e}$, since $\boldsymbol{e}^T = \boldsymbol{P}^{-1}(\boldsymbol{P}\boldsymbol{e}^T)$.

A.6.3 Hard Problems in Code-based Cryptography

Code-based cryptography relies on some proven `NP-hard` problems. These problems are the general decoding problem and the problem of finding codewords of certain weights. Code equivalence problem is another problem that plays an important role in the security of code-based cryptosystems. This problem is not proved to be `NP-hard` but it has been shown also it is not easy [70]. The later problem is the topic of our thesis and will be discussed in detail in the next chapters.

The General Decoding Problem

Let $\mathscr{C}$ be an $[n, k, d]$ linear code over $\mathbb{F}_q$ and $\boldsymbol{u} \in \mathbb{F}_q^n$, *the general decoding problem* is to find $\boldsymbol{w} \in \mathscr{C}$ such that $d(\boldsymbol{u}, \boldsymbol{w})$ is minimal.

Let $\boldsymbol{w} = \boldsymbol{u} + \boldsymbol{e}$ where $\boldsymbol{e}$ is an error vector. The decoding is unique if $wt(\boldsymbol{e}) \leqslant t = \lfloor \frac{d-1}{2} \rfloor$. It has been proven in [11] that the general decoding problem is `NP-hard`.

The Problem of Finding Codewords of Certain Weights

Consider an $[n, k, d]$ linear code over $\mathbb{F}_q$ and let $a \in \mathbb{N}$, *the problem of finding codewords of certain weights* is to find $\boldsymbol{w} \in \mathscr{C}$ such that $d(\boldsymbol{w}, \boldsymbol{0}) = a$.

It has been proven in [11] that the problem of finding codewords of certain weights is `NP-hard`.

McEliece Problem

Let $(\boldsymbol{G}, t)$ be McEliece public key as defined above and $\boldsymbol{c} \in \mathbb{F}_2^n$ a ciphertext, *McEliece problem* is to find the unique message $\boldsymbol{m} \in \mathbb{F}_2^k$ such that $d(\boldsymbol{m}\boldsymbol{G}, \boldsymbol{c}) = t$.

McEliece problem is like the general decoding problem but in specific class of code, in this case binary irreducible Goppa codes. Thus if one is able to solve the general decoding problem then he can solve McEliece problem. The opposite may not be true since Goppa codes are just one class of linear codes. Thus McEliece problem is not proved to be `NP-hard`.

Appendix B

Notes about the Hull

As we have seen in the previous sections the hull of the code has central impact in many aspects of our modeling of equivalence problem, PEP. The first impact is on the rank of the linear system $\mathfrak{E}$ that describes PEP as stated in Proposition 3.2.23. We found that as the dimension of the hull increases we get less linear equations and this of course affects the complexity of solving the system. Also as in section Block Linearization we have seen the existence of the hull might prevent block linearization technique to work and we need to go for further improvement by linearization. We found that as the hull is getting larger as the complexity increases unless we have an extension field then we can apply Frobenius action to eliminate the effect of the hull. This section gives some background results about the hull and we introduce new representation of linear codes considering the hull which we call hull representation.

B.0.1 Construction of the Hull

As we have defined before the hull of a code $\mathscr{C}$ is the intersection of the code with its dual, i.e. $\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^{\perp}$.

Proposition B.0.1. *Let $\boldsymbol{G}$ and $\boldsymbol{H}$ be the generator and parity check matrices of the code $\mathscr{C}$ then a parity check matrix of the hull $\boldsymbol{H}_{\mathscr{H}}$ is obtained by $\begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{H} \end{pmatrix}$.*

Proof. Since $\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^{\perp}$ thus $(\mathcal{H}(\mathscr{C}))^{\perp} = \mathscr{C} + \mathscr{C}^{\perp}$ and the result follows. ■

Construction of Code with Arbitrary Hull

Here we introduce an algorithm to generate an $[n, k]$ linear code over a field $\mathbb{F}_q$ with hull of a chosen dimension $h \geqslant 0$.

1: **Input**: $n, k, h, \mathbb{F}_q$.
2: **Output**: $[n, k]$ linear code over $\mathbb{F}_q$ with hull of dimension h.
3: Let $V \leftarrow \mathbf{0}$ the zero vector space of length n over $\mathbb{F}_q$
4: $\boldsymbol{G} \leftarrow 1 \times n$ zero matrix
5: $i \leftarrow 1$.

while $i \leqslant h$ **do**
  Select a vector $\mathbf{0} \neq \boldsymbol{v} \in V^{\perp}$ with $\boldsymbol{v}.\boldsymbol{v} = \mathbf{0}$ and $\boldsymbol{v}$ is linearly independent of the rows of $\boldsymbol{G}$.
  $\boldsymbol{G} \leftarrow \begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{v} \end{pmatrix}$.
  $V \leftarrow Span(rows(\boldsymbol{G}))$
  $i \leftarrow i + 1$.
end while
$V_d \leftarrow V$
while $i \leqslant k$ **do**
  Select a vector $\mathbf{0} \neq \boldsymbol{v} \in {V_d}^{\perp}$ such that $\boldsymbol{v}.\boldsymbol{v} \neq \mathbf{0}$ **and** $\boldsymbol{v}$ is linearly independent of the rows of $\boldsymbol{G}$.
  $\boldsymbol{G} \leftarrow \begin{pmatrix} \boldsymbol{G} \\ \boldsymbol{v} \end{pmatrix}$.
  $i \leftarrow i + 1$.
end while
return the linear code generated by $\boldsymbol{G}$.

B.0.2 Properties of the Hull

In this section we present some properties of the hull. We have seen how the hull affects negatively solving PEP but from the other side the hull has some interesting properties that make it useful when considering equivalence.

1. The hull is invariant by permutation equivalence: As we have proved in Proposition 1.3.2 if $\mathcal{H}(\mathscr{C})$ is a hull of code $\mathscr{C}$ and σ is a permutation then $(\mathcal{H}(\mathscr{C}))^{\sigma} = \mathcal{H}(\mathscr{C}^{\sigma})$. Thus sometimes instead of considering PEP between the codes we can consider it between their hull. This approach has been adopted by SSA.

2. The set of solutions of PEP between the hull of codes includes the set of solutions of PEP between the codes, see Theorem 1.3.2 and Example 1.3.3. Accordingly the SSA algorithm might fail to find the correct permutation since it solves PEP for the hull of codes. Thus the solution that is found by SSA needs to be validated against the codes since it might be valid solution for the hulls but not for codes.

3. According to the previous property and section Properties of the Solution Set the same relation holds between the permutation groups, i.e. the permutation group of the hull includes the permutation group of the code.

4. The dimension of the hull for random linear code over $\mathbb{F}_q$ is small with very high probability and is not always 0 and tends to a constant when the size of the code goes to infinity [74, 75]. This property is of interest for our algorithm where we achieve the best initial linear system when the hull is trivial and as the hull increases we get fewer linear equations. Also as we have seen in Section Block Linearization as the size of the hull increases we need to go for further improvements.

   This property is also of interest for the SSA algorithm since the algorithm needs to compute a signature of the codes using the weight enumerator and as the size of the code gets larger computing the weight enumerator gets complex. Thus it is suitable to use the hull since it is usually has a small size.

5. Unlike random linear codes, almost all cyclic codes have hull of large dimension. The hull of cyclic codes of length n over $\mathbb{F}_q$ is 0 if $n|q^i+1$ for some $i \geqslant 1$ otherwise it is $O(n)$ [80]. Since almost all positive integers do not belong to $\{q^i+1, i \geqslant 1\}$ hence almost all cyclic codes are of large hull [80]. This property makes cyclic codes one of the hard instances for our algorithm, see section **??** as well for SSA algorithm.

Hull and PEP

As the hull is a source of complexity for our algorithm, it can also be used to improve the modeling by adding extra linear equations to the system. If the hull has dimension $k_{\mathcal{H}(\mathscr{C})}$ with $\binom{k_{\mathcal{H}}}{2} < n$, where n is the length of the code we can use the square code of the hull since it is also stable by permutation equivalence. This will add some linear equations to the system, see section **??**. The condition $\binom{k_{\mathcal{H}}}{2}$ $< n$ is achievable with high probability for random linear codes as we have mentioned in the previous section. For codes with large hull such as cyclic, selfdaul and weakly selfdaul codes we cannot use the square of the hull.

B.0.3 Hull Representation

In this section we introduce a new representation for the generator and parity check matrix of linear codes considering the hull. In this representation, which we call it *Hull Representation*, the first rows of the matrix are the basis of the hull. We have used partially this representation in Proposition 3.2.23 to find the rank of the linear system $\mathfrak{L}$.

Definition B.0.2 (Hull Representation). Let $\mathscr{C}$ be an $[n,k]$ linear code with a hull of dimension h and let $\boldsymbol{G}$ be a generator matrix of $\mathscr{C}$. We say $\boldsymbol{G}$ is in *Hull Representation* if it has the form

$$\begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \\ \boldsymbol{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N} \end{pmatrix}$$

Where the first h rows are the basis of the hull.

Remark B.0.3. Note that when the hull is trivial $(h=0)$ or the code is self-dual or weakly self-dual $(k-h=0)$ the hull representation is identical to the standard form.

The next theorem shows that any linear code has a generator matrix in the hull representation.

Theorem B.0.4. *Any linear code has a generator matrix in the hull representation.*

Proof. Let $\mathscr{C}$ be an $[n,k]$ linear code with a hull $\mathcal{H}(\mathscr{C})$ of dimension h. We construct $\boldsymbol{G}$ as follows:

1. Find a generator matrix of the hull in the standard form. This will construct the first h rows of $\boldsymbol{G}$ and they are in the required form, $\begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \end{pmatrix}$. Remember the column swap that is performed here.
2. Find any basis for the rest of the code $\mathscr{C} \setminus \mathcal{H}(\mathscr{C})$.
3. Perform column swap on this basis as the one that is done in Step 1. This basis will construct the remaining $k-h$ rows of $\boldsymbol{G}$.

4. Do row operations to get $\mathbf{0}$ matrix below $\boldsymbol{I}_h$.
5. Do row operations among the lower $k-h$ rows and columns swap among the last $n-h$ columns to obtain the lower $k-h$ rows in the required form.

Note that the last step will preserve $\boldsymbol{I}_h$ and the $\mathbf{0}$ matrix and transform the rest to the required form. ■

The following lemma provides an important relationships between the different components of $\boldsymbol{G}$ in the hull representation. These relationship will be needed later.

Lemma B.0.5. *Let*

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \\ \mathbf{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N} \end{pmatrix}$$

be a generator matrix in the hull representation then

$$-(\boldsymbol{D}\boldsymbol{D}^T + \boldsymbol{M}\boldsymbol{M}^T) = \boldsymbol{I}_h \quad \textit{and} \quad \boldsymbol{D} = -\boldsymbol{M}\boldsymbol{N}^T$$

Proof. Since $\begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \end{pmatrix}$ is a basis of the hull then

$$\boldsymbol{G} \begin{pmatrix} \boldsymbol{I}_h \\ \boldsymbol{D}^T \\ \boldsymbol{M}^T \end{pmatrix} = \mathbf{0}$$

and this gives directly the required result. ■

The next theorem provides a formula for the parity check matrix in the standard form associated to the generator matrix in the hull representation.

Theorem B.0.6. *Let $\mathscr{C}$ be an $[n,k]$ linear code with a generator matrix*

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \\ \mathbf{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N} \end{pmatrix}$$

then the parity check matrix $\boldsymbol{H}$ is given by

$$\begin{pmatrix} \boldsymbol{N}^T\boldsymbol{D}^T - \boldsymbol{M}^T & -\boldsymbol{N}^T & \boldsymbol{I}_{n-k} \end{pmatrix}$$

Proof. Let

$$\boldsymbol{G}_1 = \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M} \end{pmatrix} \quad \text{and} \quad \boldsymbol{G}_2 = \begin{pmatrix} \mathbf{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N} \end{pmatrix}.$$

Let $\mathscr{C}_1$ and $\mathscr{C}_2$ be the linear codes generated by $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, respectively. One can easily see that

$$\boldsymbol{G}_1^{\perp} = \begin{pmatrix} -\boldsymbol{D}^T & \boldsymbol{I}_{n-h} \\ -\boldsymbol{M}^T & \end{pmatrix} \quad \text{and} \quad \boldsymbol{G}_2^{\perp} = \begin{pmatrix} \boldsymbol{I}_h & \mathbf{0} & \\ \mathbf{0} & -\boldsymbol{N}^T & \boldsymbol{I}_{n-k} \end{pmatrix}$$

are the generator matrices of ${\mathscr{C}_1}^{\perp}$ and ${\mathscr{C}_2}^{\perp}$, respectively. We note that the code $\mathscr{C}^{\perp}$, is the intersection of ${\mathscr{C}_1}^{\perp}$ and ${\mathscr{C}_2}^{\perp}$. Next we find the elements in the intersection.

$$(u,v)\boldsymbol{G}_1^{\perp} = (-u\boldsymbol{D}^T - v\boldsymbol{M}^T, u, v)$$

$$(u',v')\boldsymbol{G}_2^{\perp} = (u', -v'\boldsymbol{N}^T, v')$$

where the length of u is $k-h, v$ is $n-k, u'$ is h and v' is $n-k$.

The vectors in the intersection satisfy the following:

$$v = v', u = -v'\boldsymbol{N}^T, \text{and } u' = -u\boldsymbol{D}^T - v\boldsymbol{M}^T$$

Thus

$$\begin{aligned}(u,v)\boldsymbol{G}_1^{\perp} &= (v\boldsymbol{N}^T\boldsymbol{D}^T - v\boldsymbol{M}^T, -v\boldsymbol{N}^T, v)\\ &= v\begin{pmatrix}\boldsymbol{N}^T\boldsymbol{D}^T - \boldsymbol{M}^T & -\boldsymbol{N}^T & \boldsymbol{I}_{n-k}\end{pmatrix}\end{aligned}$$

and hence the parity check matrix $\boldsymbol{H}$ is given by

$$\begin{pmatrix}\boldsymbol{N}^T\boldsymbol{D}^T - \boldsymbol{M}^T & -\boldsymbol{N}^T & \boldsymbol{I}_{n-k}\end{pmatrix}$$

■

Next we present another proof for Theorem B.0.6.

Proof. We know that the parity check matrix $\boldsymbol{H}$ satisfies $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$. Thus we have

$$\boldsymbol{G}\boldsymbol{H}^T = \begin{pmatrix}\boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M}\\ \boldsymbol{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N}\end{pmatrix}\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\\ \boldsymbol{Z}\end{pmatrix} = \boldsymbol{0}$$

$$\Rightarrow \boldsymbol{X} + \boldsymbol{D}\boldsymbol{Y} + \boldsymbol{M}\boldsymbol{Z} = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{Y} + \boldsymbol{N}\boldsymbol{Z} = \boldsymbol{0}$$

$$\Rightarrow \boldsymbol{Y} = -\boldsymbol{N}\boldsymbol{Z} \quad \text{and} \quad \boldsymbol{X} = (\boldsymbol{D}\boldsymbol{N} - \boldsymbol{M})\boldsymbol{Z}.$$

The matrix $\boldsymbol{Z}$ can be chosen freely any $n-k \times n-k$ square matrix so we chose $\boldsymbol{Z} = \boldsymbol{I}_{n-k}$. Thus we have

$$\boldsymbol{Y} = -\boldsymbol{N} \quad \text{and} \quad \boldsymbol{X} = \boldsymbol{D}\boldsymbol{N} - \boldsymbol{M}$$

and

$$\boldsymbol{H}^T = \begin{pmatrix}\boldsymbol{D}\boldsymbol{N} - \boldsymbol{M}\\ -\boldsymbol{N}\\ \boldsymbol{I}_{n-k}\end{pmatrix}$$

and this completes the proof.

■

In the previous theorem the parity check matrix is given in the standard form. Next we deduce the parity check matrix in the hull representation.

Theorem B.0.7. *Let*

$$\boldsymbol{G} = \begin{pmatrix}\boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{M}\\ \boldsymbol{0} & \boldsymbol{I}_{k-h} & \boldsymbol{N}\end{pmatrix}$$

be a generator matrix in the hull representation of an $[n,k]$ linear code, without loss of generality, we assume $k \leqslant n-k$. Then we can write

$$\boldsymbol{G} = \begin{pmatrix}\boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{X} & \boldsymbol{M}_1\\ \boldsymbol{0} & \boldsymbol{I}_{k-h} & \boldsymbol{Y} & \boldsymbol{N}_1\end{pmatrix},$$

where $\boldsymbol{M}_1$ is $h \times k$ and $\boldsymbol{N}_1$ is $k-h \times k$.

The parity check matrix in the hull representation is given by

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{X} & \boldsymbol{M}_1 \\ \boldsymbol{0} & \boldsymbol{I}_{n-k-h} & & \boldsymbol{Z} \end{pmatrix}$$

where $\boldsymbol{Z}$ is a solution for the system

$$\boldsymbol{B}\boldsymbol{Z}^T = -\boldsymbol{A}$$

with

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{D} & \boldsymbol{X} \\ \boldsymbol{I}_{k-h} & \boldsymbol{Y} \end{pmatrix} \quad \textit{and} \quad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{M}_1 \\ \boldsymbol{N}_1 \end{pmatrix}.$$

Proof. In this theorem we assume $k \leqslant n-k$ otherwise we can consider the dual code.

The first h rows in $\boldsymbol{G}$ are basis of the hull. These rows can be the first h rows in $\boldsymbol{H}$ also. The last $n-k-h$ rows in $\boldsymbol{H}$ will have the form $\begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_{n-k-h} & \boldsymbol{Z} \end{pmatrix}$ as suggested by the hull representation. Our target is to find $\boldsymbol{Z}$ in terms of components of $\boldsymbol{G}$. We know that $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$ thus we have:

$$\boldsymbol{G}\boldsymbol{H}^T = \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{D} & \boldsymbol{X} & \boldsymbol{M}_1 \\ \boldsymbol{0} & \boldsymbol{I}_{k-h} & \boldsymbol{Y} & \boldsymbol{N}_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_h & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}^T & \boldsymbol{I}_{k-h} & \boldsymbol{0} \\ \boldsymbol{X}^T & \boldsymbol{0} & \boldsymbol{I}_{n-2k} \\ \boldsymbol{M}_1^T & \boldsymbol{Z}_1^T & \boldsymbol{Z}_2^T \end{pmatrix} = \boldsymbol{0}$$

This yields the following equations in $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$:

$$\begin{cases} \boldsymbol{D} + \boldsymbol{M}_1\boldsymbol{Z}_1^T &= \boldsymbol{0} \\ \boldsymbol{X} + \boldsymbol{M}_1\boldsymbol{Z}_2^T &= \boldsymbol{0} \\ \boldsymbol{I}_{k-h} + \boldsymbol{N}_1\boldsymbol{Z}_1^T &= \boldsymbol{0} \\ \boldsymbol{Y} + \boldsymbol{N}_1\boldsymbol{Z}_2^T &= \boldsymbol{0}. \end{cases}$$

These equations can be written as follows:

$$\begin{pmatrix} \boldsymbol{M}_1 \\ \boldsymbol{N}_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{Z}_1^T & \boldsymbol{Z}_2^T \end{pmatrix} = - \begin{pmatrix} \boldsymbol{D} & \boldsymbol{X} \\ \boldsymbol{I}_{k-h} & \boldsymbol{Y} \end{pmatrix}$$

and this gives the required result. It is worth mentioning here, this system must have at least one solution since we proved in Theorem B.0.6 that any linear code has a generator matrix in the hull representation where the selection of the basis of the hull is optional. ■

Remark B.0.8. Consider a generator matrix $\boldsymbol{G}$ in the hull representation. Let $\boldsymbol{B}$ be the matrix constructed from the last k columns of $\boldsymbol{G}$, that is $\boldsymbol{B} = \boldsymbol{G}[n-k+1; n]$. Let $\boldsymbol{A}$ be the matrix constructed from the $n-k-h$ columns before the last k columns in $\boldsymbol{G}$, that is $\boldsymbol{A} = \boldsymbol{G}[h+1; n-k]$. Then regardless of the dimension of the code k, the submatrix $\boldsymbol{Z}$ satisfies the system $\boldsymbol{B}\boldsymbol{Z}^T = -\boldsymbol{A}$.

Remark B.0.9. Note that the hull representation for the dual code as presented in Theorem B.0.7 and Remark B.0.8 is valid when $h = 0$ and $h = k$.

Appendix C

ISD Algorithm

Throughout this section we consider $\boldsymbol{G}$ a generator matrix of an $[n, k]$ linear code, $\boldsymbol{P}$ is an $n \times n$ permutation matrix, $\boldsymbol{I}_k$ is $k \times k$ identity matrix and p and l are parameters chosen in a suitable way to improve the performance of the algorithms.

Lee-Brickell's Algorithm

This algorithms was first introduced in [52] and it works as follows:

1: **Input**: $\boldsymbol{G}$ and p.
2: **Output**: Codeword of weight w.
3: Compute $\boldsymbol{GP}$ for a random $\boldsymbol{P}$ such that the first k columns of $\boldsymbol{GP}$ are linearly independent.
4: Write $\boldsymbol{GP}$ in the standard form $\boldsymbol{G}' = [\boldsymbol{I}_k\ \boldsymbol{J}]$, where $\boldsymbol{J}$ is a $k \times n - k$ matrix.
5: **for** all u, weight p vectors of length k **do**
6: **if** the weight of $u\boldsymbol{G}'$ is w **then**
7: **return** $u\boldsymbol{G}'\boldsymbol{P}^{-1} = u\boldsymbol{G}$.
8: **end if**
9: **end for**
10: **if** no such a codeword **then**
11: go to line 3.
12: **end if**

Leon's Algorithm

This algorithm was introduced by Leon in [54] where the algorithm works as Lee-Brickell's algorithm except $\boldsymbol{G}'$ is composed of three parts where the middle part is assumed to be error-free.

1: **Input**: $\boldsymbol{G}, p, l$.
2: **Output**: Codeword of weight w.
3: Compute $\boldsymbol{GP}$ for a random $\boldsymbol{P}$ such that the first k columns of $\boldsymbol{GP}$ are linearly independent.
4: Write $\boldsymbol{GP}$ in the standard form $\boldsymbol{G}' = [\boldsymbol{I}_k\ \boldsymbol{L}\ \boldsymbol{J}]$, where $\boldsymbol{L}$ is a $k \times l$ matrix and $\boldsymbol{J}$ is a $k \times n - k - l$ matrix.
5: **for** all u vectors of weight $\leqslant p$ and length k **do**

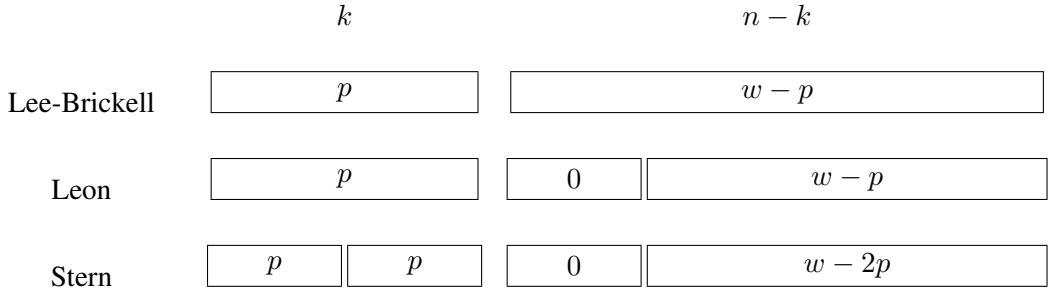
6: **if** the wight of $u[\boldsymbol{I}_k\ \boldsymbol{L}]$ is $\leqslant p$ and the weight of $u\boldsymbol{G}'$ is w **then**
7: **return** $u\boldsymbol{G}'\boldsymbol{P}^{-1} = u\boldsymbol{G}$.
8: **end if**
9: **end for**
10: **if** no such a codeword **then**
11: Go to line 3.
12: **end if**

Stern's Algorithm

This algorithm was first proposed by Stern in [82]. It has significant improvement over the previous algorithms in terms of time complexity. Below is the description of the algorithm.

1: **Input**: $\boldsymbol{G}, p, l$.
2: **Output**: Codeword of weight w.
3: Compute $\boldsymbol{GP}$ for a random $\boldsymbol{P}$ such that the first k columns of $\boldsymbol{GP}$ are linearly independent.
4: Write $\boldsymbol{GP}$ in the standard form $\boldsymbol{G}' = [\boldsymbol{I}_k\ \boldsymbol{L}\ \boldsymbol{J}]$, where $\boldsymbol{L}$ is a $k \times l$ matrix and $\boldsymbol{J}$ is $k \times n-k-l$ matrix.
5: **for** all u vectors of weight p and length $k/2$ **do**
6: Add to L_0 the codeword $x_0 = (u, 0)\boldsymbol{G}'$.
7: Add to L_1 the codeword $x_1 = (0, u)\boldsymbol{G}'$.
8: **end for**
9: Sort L_0 and L_1 according to $\phi^{\boldsymbol{L}}(x) = [x_{k+1}, ..., x_{k+l}]$, the values of x in the coordinates corresponding to $\boldsymbol{L}$
10: **for** all pairs $x_0 \in L_0$ and $x_1 \in L_1$ **do**
11: **if** $\phi^{\boldsymbol{L}}(x_0) = \phi^{\boldsymbol{L}}(x_1)$ **then**
12: Compute $x = x_0 + x_1$.
13: **if** x has weight w **then**
14: **return** $x\boldsymbol{P}^{-1}$
15: **end if**
16: **end if**
17: **end for**
18: **if** no such x **then**
19: Go to line 3.
20: **end if**

The following figure illustrates how the three algorithms work.



Complexity of ISD

A work factor of Stern's algorithm over binary field was provided in [29] as follows:

$$\frac{4^{p+1}lpk^2(n-k)(n-\frac{k+1}{2})\binom{k/2}{p}^3\binom{n-k}{l}\binom{n}{k}}{2^{l+1}\binom{2p}{p}\binom{w}{2p}\binom{k-w}{k-2p}\binom{n-k-w+2p}{l}}$$

A precise work factor over arbitrary field $\mathbb{F}_q$ was given in [20] as follows: Let Q be the cost of one iteration then:

$$\begin{aligned} Q &= (n-1)\big((k-1)(1-\tfrac{1}{q^r})+(q^r-r)\big)\frac{c}{r} \\ &+\big((\tfrac{k}{2}-p+1)+2\tbinom{k/2}{p}(q-1)^p\big)l \\ &+\tfrac{q}{q-1}(w-2p+1)2p(1+\tfrac{q-2}{q-1})\big(\tfrac{\binom{k/2}{p}^2(q-1)^{2p}}{q^l}\big) \end{aligned}$$

where c and r are parameters used to speed up the Gaussian elimination by reusing information and precomputation that are proposed in [13].

The number of iterations neglecting the dependency on the previous iteration is:

$$T=\frac{\binom{n}{w}}{\binom{k/2}{p}^2\binom{n-k-l}{w-2p}}$$

Thus the work factor of the algorithm over $\mathbb{F}_q$ is TQ.

R. C. Torres and N. Sendrier studied the asymptotic behavior of a generic ISD in [79]. Considering n as the code length, R is the code rate and ISD is looking for a codeword of weight w with $\lim_{n\to\infty} w/n = 0$ the work factor of ISD is: $2^{cw(1+O(1))}$ when n grows, and $c = -\log_2(1-R)$ is a constant.

Abstract

Code equivalence problem plays an important role in coding theory and code based cryptography. That is due to its significance in classification of codes and also construction and cryptanalysis of code based cryptosystems. It is also related to the long standing problem of graph isomorphism, a well-known problem in the world of complexity theory.

We introduce new method for solving code equivalence problem. We develop algebraic approaches to solve the problem in its permutation and diagonal versions. We build algebraic system by establishing relations between generator matrices and parity check matrices of the equivalent codes. We end up with system of multivariables of linear and quadratic equations which can be solved using algebraic tools such as Groebner basis and related techniques.

By using Groebner basis techniques we can solve the code equivalence but the computation becomes complex as the length of the code increases. We introduced several improvements such as block linearization and Frobenius action. Using these techniques we identify many cases where permutation equivalence problem can be solved efficiently. Our method for diagonal equivalence solves the problem efficiently in small fields, namely $\mathbb{F}_3$ and $\mathbb{F}_4$. The increase in the field size results in an increase in the number of variables in our algebraic system which makes it difficult to solve.

We introduce a new reduction from permutation code equivalence when the hull is trivial to graph isomorphism. This shows that this subclass of permutation equivalence is not harder than graph isomorphism. Using this reduction we obtain an algebraic system for graph isomorphism with interesting properties in terms of the rank of the linear part and the number of variables. We solve the graph isomorphism problem efficiently for random graphs with large number of vertices and also for some regular graphs such as Petersen, Cubical and Wagner Graphs.

Keywords

code equivalence, graph isomorphism, Groebner bases, polynomial system, code-based cryptography.

Résumé

Le problème d'équivalence de code joue un rôle important dans la théorie de code et la cryptographie basée sur le code. Cela est dû à son importance dans la classification des codes ainsi que dans la construction et la cryptanalyse des cryptosystèmes à base de codes. Il est également lié à un problème ouvert d'isomorphisme de graphes, un problème bien connu dans le domaine de la théorie de la complexité.

Nous prouvons pour les codes ayant un hull trivial qu'il existe une réduction polynomiale de l'équivalence par permutation de codes à l'isomorphisme de graphes. Cela montre que cette sous-classe d'équivalence de permutation n'est pas plus dure que l'isomorphisme de graphes.

Nous introduisons une nouvelle méthode pour résoudre le problème d'équivalence de code. Nous développons des approches algébriques pour résoudre le problème dans ses deux versions : en permutation et en diagonale. Nous construisons un système algébrique en établissant des relations entre les matrices génératrices et les matrices de parité des codes équivalents. Nous nous retrouvons avec un système plusieurs variables d'équations linéaires et quadratiques qui peut être résolu en utilisant des outils algébriques tels que les bases de Groebner et les techniques associées.

Il est possible en thorie de résoudre l'équivalence de code avec des techniques utilisant des bases de Groebner. Cependant, le calcul en pratique devient complexe à mesure que la longueur du code augmente. Nous avons introduit plusieurs améliorations telles que la linéarisation par bloc et l'action de Frobenius. En utilisant ces techniques, nous identifions de nombreux cas o le problème d'équivalence de permutation peut être résolu efficacement. Notre méthode d'équivalence diagonale résout efficacement le problème dans les corps de petites tailles, à savoir $\mathbb{F}_3$ et $\mathbb{F}_4$. L'augmentation de la taille du corps entrane une augmentation du nombre de variables dans notre système algébrique, ce qui le rend difficile à résoudre.

Nous nous intressons enfin au problme d'isomorphisme de graphes en considrant un système algébrique quadratique pour l'isomorphisme de graphes. Pour des instances tires alatoirement, le systme possdent des propriétés intéressantes en termes de rang de la partie linéaire et du nombre de variables. Nous résolvons efficacement le problème d'isomorphisme de graphes pour des graphes aléatoires avec un grand nombre de sommets, et également pour certains graphes réguliers tels que ceux de Petersen, Cubical et Wagner.

Mots clés

équivalence de code, isomorphisme de graphes, bases de Groebner, système polynomial cryptographie à base de codes.