ulm university universität
uulm

Decoding Evaluation Codes and their Interleaving

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS
(DR.-ING.)

der Fakultät für Ingenieurwissenschaften,
Informatik und Psychologie der Universität Ulm

von

Wenhui Li

aus Jinan, China

Betreuer:	Prof. Dr.-Ing. Martin Bossert
	Prof. Dr. Hans-Andrea Löliger
Amtierende Dekanin:	Prof. Dr. Tina Seufert

Ulm, 24.04.2015

Preface

THIS dissertation is an original intellectual product of the author, Wenhui Li, who has spent four years in the Institute of Communications Engineering at the University of Ulm as a research assistant. The purpose of this dissertation is to develop algorithms to correct errors and erasures for evaluation codes and their interleaving. This research was supported by the *German Research Council* (Deutsche Forschungsgemeischaft DFG) under project Bo 867/22-1. Part of the results were presented in a number of symposia and conferences and are published in proceedings of these conferences and in journal *Designs, Codes and Cryptography*.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Dr.-Ing. Martin Bossert for giving me opportunity to enter the research world of algebraic coding theory and work with his group, and for supporting me with my PhD study and related research. Thanks to him, that I was able to go to many conferences, and meet and discuss with great scientists.

I would like to thank the second reviewer Prof. Dr. Hans-Andrea Löliger from Signal and Information Processing Laboratory (Institut für Signal- und Informationsverarbeitung ISI), ETH, Zürich, and the rest of my thesis committee from University of Ulm: Prof. Dr.-Ing. Stefan Wesner, and Prof. Dr. Jian Xie, for their valuable comments and suggestions. Their questions widen my research from various perspectives.

My completion of this dissertation could not have been accomplished without the support of Dr. Vladimir Sidorenko, my closest friend. His lecture Channel Coding inspires my interest to coding theory. His guidance helped me throughout my research, thesis writing as well. I will forever be beholden to him for his patience, immense knowledge, selfless contribution, and all of our outdoor sports in spare time.

I truly appreciate former colleagues for our cooperative and progressive work: Dr. Alexander Zeh, who supervised my Master thesis, confirmed me to continue the coding direction for doctoral study, and recommended great ideas; Dr. Sabina Kampf, who disclosed the secret of varying-length MS-LFSR synthesis; Dr. Johan S.R. Nielsen, who gave tremendous help in discussing, and working together before deadlines for ICR codes.

Tons of thanks go to Sven Mülich, Mostafa Hosni Mohamed, Sven Puchinger, Vladimir Sidorenko, and my boss Martin Bossert for helping proofread my dissertation draft. I appreciate Susanne Sparrer for sharing her tea with me time after time. Four-year's work can not be done without computer, hence I thank Linux experts Werner Hack and Günther Haas for their technical support in several ways. Also I thank the rest of my colleagues in

the institute of Communications Engineering. In particular, I am very grateful to secretaries Ulrike Stier, Ilse Walter, and Fe Bauer, who endured my numerous annoying disturbance of paper work.

Last but not the least, I would like to thank my parents from my deep heart for supporting me spiritually through the PhD period and my life in general.

Ulm, September 2015, *Wenhui Li*

Contents

Abstract

Four classes of evaluation codes are considered in this dissertation. They are Reed–Solomon (RS) codes, Hermitian codes, Gabidulin codes, and Chinese remainder codes. We also consider the interleaving of these codes since this we can benefit from this special structure. Our goal is to construct good decoding algorithms which are able to correct more errors with less complexity and less failure probability if any.

The classical decoding approaches for RS codes allow correcting up to half the minimum distance number of errors. By the interleaving scheme, an RS code can be decoded beyond half the minimum distance. In the dissertation, for decoding interleaved RS codes, a version of a fast algorithm which is based on the extended Euclidean algorithm is proposed, reducing the complexity from quadratic to sub-quadratic in length.

Furthermore, a joint decoding algorithm for interleaved *extended* RS codes is proposed, having quadratic in length complexity. We apply this algorithm to decode Hermitian codes resulting in correcting up to $(N-K)/(q+1)$ burst errors with complexity $\mathcal{O}(N^{1\frac{2}{3}})$ operations in $\mathbb{F}_q$ where N is the length and K is the dimension of Hermitian codes. The low rate Hermitian codes can correct even more burst errors using "power" decoding.

The error and erasure decoding for Gabidulin codes and Chinese remainder codes are considered. For a Gabidulin code with minimum (rank) distance d, a transform domain algorithm is proposed with quadratic number of operations in $\mathbb{F}_{q^m}$ to correct ε full errors, μ_R row erasures and μ_c column erasures as long as $2\varepsilon + \mu_R + \mu_C \leq d-1$. For an (n,k) Chinese remainder code with minimum distance d, any pattern of ε errors and μ erasures will be corrected, provided that $(\log p_1 + \log p_n)/\log p_1 \varepsilon + \mu \leq d-1$ where the first and last coordinates of the codeword are over $\mathbb{Z}_{p_1}$ and $\mathbb{Z}_{p_n}$. A syndrome-based decoder for error correction for Chinese remainder codes is proposed.

Decoding interleaved Gabidulin codes and decoding interleaved Chinese remainder codes are also considered. The complexities of both decoders are analyzed and show that the proposed algorithms are efficient. The decoding radius and failure probability are also analyzed for both decoders.

1

Introduction

MODERN information technologies can not exist without using error correcting codes. Processors become more and more powerful which allow us to implement more and more sophisticated algorithms to correct more errors, which is necessary since volumes of data, transmission rate, and density of recording are permanently growing.

It was estimated that the most of applications with error correcting codes are based on **Reed-Solomon (RS) codes** named in honor of Irving S. Reed and Gustave Solomon [RS60] who invented this class of codes in 1960. They are widely used in data transmission (DSL, WiMAX etc.), space communication (Voyager program, Galileo spacecraft etc.), data storage (CDs, Blu-ray Discs etc.), cloud computation, and cloud storage. Furthermore, RS codes are found very frequently in two-dimension (2D) codes. As an upgrade of one dimension barcode, 2D codes are readable two dimensional patterns which are used for product identification. The information related to the product is encrypted in this pattern which is actually an RS codeword. In comparison with the first generation barcode, 2D code contains much more information and has higher reliability. Nowadays, 2D code is becoming the most frequently used type in transportation, cellphone scanning etc.

The RS codes are an outstanding class of codes. They are defined over the finite field $\mathbb{F}$ and have several splendid properties, and hence, the decoding of RS codes has always been one of the hot topics in research. We transmit the RS codewords over the noisy channel and receive words with errors. If every symbol in the received words is from $\mathbb{F}$, then the decoder in the next step uses *hard decision* decoding approaches. Otherwise, the received words contain additional information (reliabilities) from the channel and *soft decision* decoding schemes are applied for the decoder.

In the late 90s, Krachkovsky and Lee started to work on *interleaved Reed–Solomon* (IRS) codes to correct burst errors jointly [KL97, Kra03]. Among the hard decision decoding methods, the well-known Berlekamp–Massey algorithm [Ber68, Mas69] can be generalized to decode arbitrary IRS codes efficiently [SSB09]. The Euclidean algorithm which has an equivalence connection to the Berlekamp–Massey algorithm, however, is generalized by Feng and Tzeng [FT89] to decode only homogeneous IRS codes.

RS codes are also used in one of the structures of secret sharing (Shamir's secret

sharing [Sha79]). It refers to the methods by which the secret is distributed to a set of participants, each of whom has a share of the secret, and the secret can only be reconstructed by a subset of the participants.

Part of this dissertation is devoted to applying the generalized Euclidean algorithm to decode the heterogeneous IRS codes and accelerating the proposed algorithm as well.

There is another class of codes which can be used in secret sharing as well [GRS00]. The codes are called **Chinese remainder codes**, and they are named after the ancient Chinese remainder theorem. The Chinese remainder codes can be used not only in cryptography, but also in many other applications, for example, *cloud computation systems*. When we manipulate arithmetic operations (except division) with large integer numbers, it is time-consuming to run the task on one computer. Using the Chinese remainder theorem, running a big task is converted to running smaller tasks on a system of $n > 1$ computers, where the results (still small data) are collected for the final calculation. Using the Chinese remainder codes, although some computers from the system are crashed, we can still recover the data from the corrupted computer, and obtain the correct final result. The destroyed computers cause errors if the computer locations are not known. Otherwise, they create erasures.

The *interleaved Chinese remainder codes* allow for recovering data when a system of computers are used to run multiple big tasks. A corrupted computer from the system creates an error burst, since it could not handle any tasks. Using the structure of interleaving, more error bursts can be corrected in comparison with using a single Chinese remainder code. Developing efficient algorithms for decoding Chinese remainder codes and their interleaving with errors or with erasures is also the topic of the dissertation.

A hot topic in the last decades is *network coding* [ACLY00] which requires rank metric codes. A representative of rank metric codes is **Gabidulin codes** [Del78, Gab85, Rot91] which have drawn significant attention in network coding, or more specifically, in *random linear network coding* [HKM$^+$03]. In the latter model, a node in the network receives packets from different routes and forward their random linear combination as one packet. Hence, an erroneous packet affects the successive ones and then paralyzes the transmission. Gabidulin codes are used in this case such that they can provide close to optimal solution to the error control problem [KK08, SKK08]. Besides errors, erasures which can be defined in rank metric [SKK08, GP08, GPT91] can also occur in a network. Correcting errors and erasures in a lossy network using Gabidulin codes and using interleaved Gabidulin codes will be considered in this dissertation.

Shannon showed that the communication which is almost error free is possible using long good codes [Sha48]. Although the RS codes are maximum distance separable codes, a defect of using RS codes is that their length are restricted by the size of the field. From this point of view, **Hermitian codes** and RS codes have almost the same normalized minimum distance, but the length of Hermitian codes can be much longer than the size of the field. This fact makes Hermitian codes potentially very interesting for scientific research and applications. There are some publications about reducing decoding a Hermitian code to decoding interleaved RS codes [YB92, Ren04]. An efficient algorithm will be proposed in

the dissertation, competing with previous results in the respects of decoding radius and complexity.

The RS codes, Chinese remainder codes, Gabidulin codes, and Hermitian codes all belong to the family of **evaluation codes**. The dissertation is dedicated to decoding these four classes of evaluation codes and their interleaving with the algorithms which provide good performances with respect to decoding radius, complexity, and failure probability. The dissertation is structured as follows.

Structure of the work

In **Chapter 2**, we define the family of evaluation codes and show that four code classes considered in the dissertation belong to this family. Moreover, the code metrics and the considered channel model throughout the dissertation are specified. Some frequently used decoding paradigms for the evaluation codes with errors and/or erasures are introduced as well. The structure of interleaving is briefly described in the end.

In the next four chapters, each class of evaluation codes is considered and analyzed individually. For all the proposed algorithms, the complexity, failure probability, and decoding radius are analyzed.

We start with the most widely used codes — RS codes in **Chapter 3**. Basic terms and facts which are required for RS codes are briefly introduced as a preclude, including different definitions in terms of the polynomial, matrix, and discrete Fourier transform, the key equation based on the syndrome, and the well-known decoding algorithms: Berlekamp–Massey algorithm and Sugiyama et.al algorithm [SKHN75]. The generalization of these two algorithms is shown by Schmidt, Sidorenko, and Bossert in [SSB09] and by Feng and Tzeng in [FT89] to decode IRS codes. As a reference, we provide the pseudo codes of all these algorithms for decoding RS codes and their interleaving except the Feng–Tzeng algorithm. As we mentioned before, the drawback of Feng–Tzeng algorithm is that their algorithm can not be applied to decode heterogeneous IRS codes. Therefore, we modify the Feng–Tzeng algorithm such that it can be applied for decoding arbitrary IRS codes. The pseudo codes of the modified algorithm is given together with an example and the complexity analysis. Zeh and Wachter show that the divide and conquer strategy allows for the acceleration of the Feng–Tzeng algorithm [ZW11], based on which we fasten our modified algorithm. An example of decoding IRS codes using fast algorithm follows its pseudo codes. Detailed analysis of the complexity of the fast algorithm is shown.

Due to very strong relation to decoding IRS codes, decoding Hermitian codes is discussed immediately after RS codes in **Chapter 4**. After a simple definition of Hermitian codes without too much knowledge of algebraic geometry and the definition of phased bursts, reducing decoding Hermitian codes to decoding interleaved extended RS (IERS) codes is explained. Based on decoding a single extended RS code, a joint decoding algorithm for IERS codes is proposed. In addition, an upper bound of failure probability and decoding radius for decoding IERS codes are described. Equipped with the proposed algorithm for

IERS codes, we also show that "power " decoding and "mixed" decoding can improve the decoding performance for low-rate IERS codes. We return to the theme of this chapter eventually, sketching and analyzing the algorithm for decoding Hermitian codes.

Chapter 5 deals with decoding Gabidulin codes and interleaved Gabidulin codes. Firstly, some notations and the definition of Gabidulin codes in terms of parity check matrix are introduced. Then we show that, the Gabidulin codes can also be obtained by evaluating skew polynomials. The problem of error correction for Gabidulin codes is considered as a skew shift register synthesis by the Berlekamp–Massey type algorithm from which error positions are obtained. Then we define "erasures" for the rank metric, which is not trivial, propose an efficient decoding algorithm in transform domain, correcting both errors and erasures for Gabidulin codes. We also generalize this algorithm for interleaved Gabidulin codes, for which the decoding failure probability and decoding radius are given.

Chapter 6 is devoted to the Chinese remainder codes. After a short introduction of classical Chinese remainder codes and some relevant notations, a syndrome-based error correcting algorithm is proposed. Then Chinese remainder codes are generalized in order to create bounded minimum distance decoder for correcting both errors and erasures. The same idea as for the previous evaluation codes is applied to define interleaved Chinese remainder codes. Inspired by the lattice reduction, the corresponding decoding algorithm for interleaved Chinese remainder codes is proposed, together with an example and complexity analysis. The failure probability and the decoding radius are studies intensively to obtain theoretical bounds. We also give the simulation results to illustrate the tightness of the bounds. The proposed algorithm for interleaved Chinese remainder codes is extended to decode a single low-rate Chinese remainder code and low-rate interleaved Chinese remainder codes, which are sketched at the end of this chapter.

The dissertation is completed by some concluding discussion in **Chapter 7**.

2

Notations and Definitions

As an overview of the whole dissertation, this chapter can be regarded as a preliminary to evaluation codes. Some basic knowledge of these codes is introduced here, including the subclasses of evaluation codes, the channel model, and the algebraic decoders at the receiver side. Some of the decoders which we discuss through the dissertation can correct only errors, and some can correct both errors and erasures. By using interleaved codes, we can decode burst errors beyond half the minimum distance.

2.1 Evaluation Codes

Evaluation codes are a class of codes which can be obtained by evaluation of polynomials (or integers) at certain points. The number of points is the length of the codes. There are quite a lot of evaluation codes, such as Reed–Muller codes, Reed–Solomon codes, and BCH codes etc. In this dissertation, we will consider four types of codes which are defined over the polynomial ring, the algebraic geometric curve, the linearized (skew) polynomial ring, and the integer ring, respectively. These codes are Reed–Solomon codes, Hermitian codes, Gabidulin codes, and Chinese remainder codes. In the following chapters, we will explore them individually.

Reed–Solomon Codes

In 1960, Irving S. Reed and Gustave Solomon invented non-binary error correcting codes which were later called Reed–Solomon (RS) codes. Consider a ring $\mathbb{F}[x]$ of polynomials

$$\{f(x) = f_{k-1}x^{k-1} + \cdots + f_1 x + f_0 \mid f_i \in \mathbb{F}\}$$

where $\mathbb{F}$ indicates a finite field. Let $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ be n distinct elements over $\mathbb{F}$. For an integer $0 < k \leq n$, the RS code $\mathcal{RS}(n,k)$ is a set of n-tuples $(c_0, c_1, \ldots, c_{n-1})$, where every component c_i for $i = 0, \ldots, n-1$ is obtained by evaluating polynomials $f(x) \in \mathbb{F}[x]$ at α_i, i.e.,

$$\mathcal{RS}(n,k) = \{(f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{n-1})) \mid f(x) \in \mathbb{F}[x] \text{ and } \deg f(x) < k\}.$$

Hermitian Codes

Hermitian codes are one of the most studied families of algebraic geometry (AG) codes which are also known as Goppa codes. Consider an extension field $\mathbb{F}_Q$ where $Q = q^2$ and q is a power of prime. A Hermitian curve $\mathcal{H}(q)$ over $\mathbb{F}_Q$ is defined by

$$y^q + y = x^{q+1}.$$

Let $P = \{P_1, \ldots, P_n\}$ be a set of all points (x, y) on $\mathcal{H}(q)$ except the single point at infinity. Given bivariate polynomials of the form

$$h(x, y) = f^{(1)}(x) + yf^{(2)}(x) + \cdots + y^{q-1}f^{(q)}(x),$$

where $f^{(i)}(x) \in \mathbb{F}_Q[x]$ of degree at most $\lfloor (m - (i-1)(q+1))/q \rfloor$ and m is an integer at least $q^2 - 1$, the Hermitian code $\mathcal{H}_m$ of length n over $\mathbb{F}_Q$ is defined by evaluation of $h(x, y)$ at all points from P, i.e.,

$$\mathcal{H}_m = \left\{ (h(P_1), \ldots, h(P_n)) \mid P_j \in \mathcal{H}(q) \text{ and } \deg f^{(i)}(x) \leq \left\lfloor \frac{m - (i-1)(q+1)}{q} \right\rfloor \right\}.$$

Gabidulin Codes

Consider an extension field $\mathbb{F} = \mathbb{F}_{q^m}$ and the ring of linearized q-polynomials

$$\mathbb{F}_{(q)}[x] = \{ f_{(q)}(x) = f_n x^{[k-1]} + \cdots + f_1 x^{[1]} + f_0 x^{[0]} \mid f_i \in \mathbb{F} \text{ and } k \in \mathbb{N} \}$$

where $x^{[i]}$ is the Frobenius power with $x^{[i]} = x^{q^i}$. For an integer $n \leq m$, let us fix the vector

$$\boldsymbol{h} = (h_0, h_1, \ldots, h_{n-1}),$$

with elements $h_i \in \mathbb{F}_{q^m}$ linearly independent over $\mathbb{F}_q$, the Gabidulin code $\mathcal{G}(q^m; n, k)$ with $k \leq n$ is defined by the evaluation of linearized polynomials $f_{(q)}(x) \in \mathbb{F}_{(q)}[x]$ with restricted degree at points h_0, h_1, $\ldots$, h_{n-1}, i.e.,

$$\mathcal{G}(q^m; n, k) = \{ \big(f_{(q)}(h_0), \ldots, f_{(q)}(h_{n-1}) \big) \mid f_{(q)}(x) \in \mathbb{F}_{(q)}[x] \text{ and } \deg f_{(q)}(x) < k \}.$$

Since any vector of length n over $\mathbb{F}_{q^m}$ can be considered as a $(m \times n)$ matrix over the subfield $\mathbb{F}_q$, the Gabidulin code can have two equivalent forms: as the set of matrices C over the base field $\mathbb{F}_q$, or the set of vectors $\boldsymbol{c}$ over the extension field $\mathbb{F}_{q^m}$.

Chinese Remainder Codes

The evaluation of polynomials $f(x)$ over any field at some point a can be considered as the remainder when $f(x)$ is divided by $(x - a)$. Indeed, $f(x) = q(x)(x - a) + r$ where $q(x)$ is the quotient and r is the remainder, hence, $r = f(a)$. Let us denote the remainder r by $[f(x)]_{(x-a)}$, then

$$f(a) = [f(x)]_{x-a}.$$

Now consider two integers $A > B$ and denote by $[A]_B$ the remainder when we divide A by B, $[A]_B = A \mod B$. Thus, this modulo operation for integers is equivalent to evaluation for polynomials. There is a class of codes which are obtained by "evaluating" an integer, and they are called Chinese remainder codes.

Given a list of n prime numbers $\mathcal{P} = (p_1, p_2, \ldots, p_n)$ where $0 < p_0 < p_1 < \cdots < p_{n-1}$. For $0 < k < n$, let us define $K = \prod_{i=0}^{k-1} p_i$. A Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ having cardinality K and length n over the list $\mathcal{P}$ is defined as follows

$$\mathcal{CR}(\mathcal{P}; n, K) = \{([C]_{p_1}, \ldots, [C]_{p_n}) \mid C \in \mathbb{N} \text{ and } C < K\}.$$

Minimum Code Distance

Given a metric, i.e., for every two words $\boldsymbol{a}$ and $\boldsymbol{b}$, the distance $\mathrm{dist}(\boldsymbol{a}, \boldsymbol{b})$ between $\boldsymbol{a}$ and $\boldsymbol{b}$ where the function dist() satisfies the axioms of a metric, the minimum distance d of the code $\mathcal{C}$ is defined as $\min \mathrm{dist}(\boldsymbol{c}, \boldsymbol{c}')$ where $\boldsymbol{c} \neq \boldsymbol{c}'$ and $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$.

For RS codes, Hermitian codes, and Chinese remainder codes, we consider *Hamming* metric, i.e., $\mathrm{dist}(\boldsymbol{a}, \boldsymbol{b})$ is the number of positions in which the words $\boldsymbol{a}$ and $\boldsymbol{b}$ differ. In case of Gabidulin codes the words are matrices, and we consider *rank* metric, i.e., the distance between two words is defined as the rank of their difference.

RS codes, Gabidulin codes, and Chinese remainder codes satisfy the Singleton bound with equality, i.e., $d = n - k + 1$. The codes are called maximum distance separable (MDS) codes for Hamming metric and maximum rank distance (MRD) codes for rank metric.

2.2 Channel Model

Let $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{F}^n$ be a codeword, and $\boldsymbol{r} = (r_0, \ldots, r_{n-1}) \in \mathbb{F}^n$ the received word. An *additive* channel is illustrated in Figure 2.1. It means that $\boldsymbol{c}$ and $\boldsymbol{r}$ have the same alphabet and $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{e} = (e_0, \ldots, e_{n-1}) \in \mathbb{F}^n$ is an error word. The weight of the error word is the number of errors. We assume that the additive channel is *memoryless*,

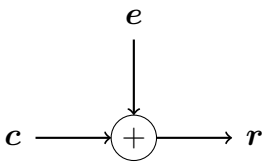


Figure 2.1: An additive noisy channel

i.e., the symbols are transmitted independently.

q-ary Symmetric Channel with Error and Erasures

When an error occurs, the component in the codeword is replaced by other element of the field. Assume an error symbol occurs with probability p.

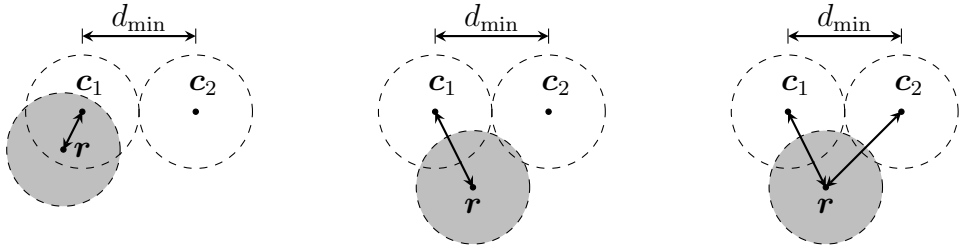
In some cases, some symbols are "scrambled" at the receiver, which means that the receiver knows the erased positions, but does not know the value of the symbols. Thus, these erased symbols are considered as erasures. Denote the symbol at an erased position by ϵ and an erasure occurs with probability p_ϵ.

By $p(r_i|c_i)$ for $i = 0, \dots n-1$ we denote the probability that the symbol r_i is received after we transmit the codeword symbol c_i. For a q-ary symmetric channel with error and erasures, the crossover probability is as follows:

$$p(r_i|c_i) = \begin{cases} 1-p-p_\epsilon & \text{if } r_i = c_i \\ p_\epsilon & \text{if } r_i = \epsilon \\ p/(q-1) & \text{otherwise} \end{cases} .$$

2.3 Decoding Evaluation Codes

In this dissertation, all the algebraic decoders we present are *bounded distance* decoders which means that they are able to decode up to a certain number of errors. With the decoder for Gabidulin codes, error matrices with up to a certain number of rank can be corrected. Consider a codeword $\boldsymbol{c}$ of an evaluation code $\mathcal{C}$ at the center, and a ball of radius t around the codeword, we have three paradigms to decode such codes [Moo05]. They are illustrated in Figure 2.2.



(a) BD decoding (b) ML decoding (c) List decoding

Figure 2.2: BD, ML and list decoding comparison

- Bounded distance (BD) decoding
  The decoder can correct up to a certain number of errors. Given a received word $\boldsymbol{r}$. If $\boldsymbol{r}$ is in the ball of some codeword with radius $t = \lfloor (d_{\min} - 1)/2 \rfloor$, then a unique codeword $\boldsymbol{c}$ can be always delivered. Such a decoder is also known as the *bounded minimum distance* (BMD) decoder. If $\boldsymbol{r}$ lies outside half the minimum distance of the code, it is also possible to decode with some efficient probabilistic algorithms. Nevertheless, in this case, the decoder might fail.

- Maximum Likelihood (ML) decoding
  Given a certain boundary between codewords, the decoder selects the codeword which is in the same area as the received word. The boundary is not necessarily a sphere,

and it can be of any shape. In other words, the ML decoder selects the nearest codeword to $\boldsymbol{r}$. The number of the errors can be larger than half the minimum distance, in which case, ML decoding has high complexity [BMvT78].

- List decoding
  All possible codewords $\boldsymbol{c} \in \mathcal{C}$ could be found within a given distance from the received word $\boldsymbol{r}$. The distance can be larger than half the minimum distance. The Sudan approach in [Sud97] and the one-step-further Guruswami–Sudan approach in [GS98] are representatives of list decoders.

Error Correction

When a word is received, at some positions the symbols can be either errors or erasures. Let us start with decoding when only errors occurred.

Let us take RS codes as an example. Denote the number of errors by t. When $t \leq \lfloor (d-1)/2 \rfloor$, the algebraic decoding of an RS code is as follows. The decoder first finds the error positions as roots of a polynomial known as *error locator* polynomial (ELP). There are some classical algorithms such as Belekamp–Massey algorithm (BMA) and extended Euclidean algorithm (EEA) to obtain the ELP involving *syndromes*. The second step is to calculate the error values. This can be done by solving linear equations with syndromes, or by some efficient algorithm such as Forney algorithm.

Erasure Correction

Since the positions of the erasures are known, calculating the erasure values is equivalent to the second step of error decoding.

Denote the number of erasures by ε. If we remove all erased positions, then the *punctured* RS code has minimum distance $\geq d - 1 - \varepsilon$. Since the punctured code has only errors, the decoder can correct up to $\lfloor (d-1-\varepsilon)/2 \rfloor$ errors, i.e., $t \leq \lfloor (d-1-\varepsilon)/2 \rfloor$.

Overall, to decode an RS code with t errors and ε erasures, the following inequality should be satisfied.

$$2t + \varepsilon \leq d - 1.$$

For other evaluation codes, we consider the following decoders in this dissertation.

- Decoding Hermitian codes will be reduced to decoding *interleaved* RS codes. We only consider the error-only case for Hermitian codes.

- For Gabidulin codes and Chinese remainder codes, due to the similar structure to RS codes, we would adapt the same technique to develop error-erasure decoders for Gabidulin codes and Chinese remainder codes. On the other hand, different algebraic structures of Gabidulin codes and Chinese remainder codes mean that the analysis varies a lot.

2.4 Interleaved Evaluation Codes

The codeword of an interleaved evaluation code can be considered as a matrix in which every row is a codeword from the same evaluation code. The number of rows is called *interleaving order* which we denote by ℓ through the whole dissertation. Speaking of the error word for the interleaving case, we consider *burst errors*. A whole column of an interleaved codeword is corrupted by one burst error. A typical example of correcting burst errors is reading CDs with scratches. The scratches are regarded as burst errors and they can be corrected using interleaved RS codes in the CD, c.f. [WB94].

One can decode each row of the interleaved evaluation codeword separately, which clearly increases the time complexity of such brute-force decoder with a factor of ℓ, compared with the complexity of decoding a single code. Furthermore, the number of errors should not exceed half the minimum distance for each row. Since the error positions are assumed to be the same for all rows, we can use this special structure to find these positions. Afterwards, the error values for each row will be calculated individually. Joint decoding is more efficient than the brute-force decoder: it allows decoding errors beyond half the minimum distance without increasing the complexity.

In this work, we consider error decoders for interleaved RS (IRS) codes, interleaved Gabidulin codes, and interleaved Chinese remainder codes. There are many research results about decoding IRS codes, based on which we will propose a fast algorithm.

3

Decoding Interleaved Reed–Solomon Codes

Reed–Solomon (RS) codes are named in honor of Irving S. Reed and Gustave Solomon [RS60] who invented this class of codes. The RS codes are evaluation codes, which means that every codeword is the evaluations of a polynomial $f(x) \in \mathbb{F}[x]$ with $\deg f(x) < k \leq n$ at points α_i, for $i = 1, \ldots, n$, where α_i are different elements over $\mathbb{F} = \mathbb{F}_q$. We denote the length, dimension and minimum distance of an RS code $\mathcal{C}_{RS}$ by n, k and d, respectively. RS codes are *maximum distance separable* (MDS) since they fulfill the Singleton bound with equality.

There exist three methods to combine RS codes: interleaving, concatenation and folding. In our work, we mainly consider the interleaved RS (IRS) codes. The advantages of applying interleaving scheme to RS codes are as follows.

- In addition to random errors, burst errors can also be corrected;
- The decoding radius is increased beyond half the minimum distance in comparison with a single code. If the interleaving factor ℓ goes to infinity, then up to $d - 2$ error bursts can be corrected.

Thus, IRS codes are applied in noisy channels with memory which produce errors. Therefore, as an outer code, the IRS code can be used in code concatenation designs [SSB09].

The rest of this chapter is organized as follows. We shortly recall RS codes in Section 3.1. Some decoding approaches of RS codes including the classical Berlekamp–Massey algorithm and Sugiyama et.al algorithm (based on the Euclidean algorithm) are introduced in Section 3.2. We introduce the generalized Berlekamp–Massey algorithm for decoding IRS codes in Section 3.3. In Section 3.4, we modify Feng–Tzeng's algorithm [FT89] based on generalized Euclidean algorithm such that it can be applied for decoding not only homogeneous IRS codes but also heterogeneous IRS codes. The modified algorithm has quadratic polynomial time complexity in length. At the end of this chapter (Section 3.5), we propose a fast algorithm based on the generalized Euclidean algorithm to decode IRS codes, and reduce the complexity to sub-quadratic operations in length in $\mathbb{F}$.

3.1 Basics of RS Codes

A block code is denoted by $\mathcal{C}(n,k)$, or simply $\mathcal{C}$, where n is the (block) length of the code, and $k \leq n$ is number of information symbols. In other words, k information symbols are *encoded* as a *codeword* of n symbols $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$. Therefore, the code *rate* is $R = k/n \leq 1$.

In our work, most of the evaluation codes are defined over some field. Let $\mathbb{F}$ be an arbitrary commutative field.

Definition 1 (Linear code). *A code is called linear, if any linear combination of two codewords over $\mathbb{F}$ is again a codeword, i.e.,*

$$a\boldsymbol{c}_1 + b\boldsymbol{c}_2 \in \mathcal{C} \quad \text{for} \quad \boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C},\ a, b \in \mathbb{F}.$$

If $\mathcal{C}$ is a linear (n,k) code, then k is the *dimension* of the code.

Any word can be described not only by an n-tuple $(c_0, \ldots, c_{n-1}) \in \mathbb{F}^n$, but also by the polynomial: $c(x) = \sum_{i=0}^{n-1} c_i x^i \in \mathbb{F}[x]$ where $\mathbb{F}[x]$ is the polynomial ring over the field $\mathbb{F}$.

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field of order q. If q is a prime, we call such a finite field *prime field*, or *extension of a prime field* if q is a power of a prime. Let α denote a primitive element in $\mathbb{F}_q$. Any nonzero element in $\mathbb{F}_q$ can be represented as a power of the primitive element, i.e., α^i for $i = 0, 1, \ldots, q-1$.

We use Hamming metric to measure the *distance* $\operatorname{dist}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ between two words $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, i.e., $\operatorname{dist}(\boldsymbol{c}_1, \boldsymbol{c}_2)$ is the number of positions where they differ. The *Hamming weight* $\operatorname{wt}(\boldsymbol{c})$ is the number of nonzero symbols in $\boldsymbol{c}$.

Definition 2 (Minimum distance). *The minimum distance d, or simply distance d, of a code is the smallest (Hamming) distance of any two codewords in the code, i.e.,*

$$d = \min \operatorname{dist}(\boldsymbol{c}_1, \boldsymbol{c}_2) = \min \operatorname{wt}(\boldsymbol{c}_1 - \boldsymbol{c}_2),\ \text{for } \boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C} \text{ and } \boldsymbol{c}_1 \neq \boldsymbol{c}_2.$$

If $\mathcal{C}$ is linear, the distance

$$d = \min \operatorname{wt}(\boldsymbol{c}) \text{ for } \boldsymbol{c} \in \mathcal{C} \text{ and } \boldsymbol{c} \neq \boldsymbol{0}.$$

The distance of any block code is upper bounded by the Singleton bound [Sin64].

Theorem 1 (Singleton bound). *If $\mathcal{C}$ is a block code of length n and minimum distance d over $\mathbb{F}_q$, then*

$$|\mathcal{C}| \leq q^{n-d+1}.$$

If $\mathcal{C}$ is linear, then

$$k \leq n - d + 1.$$

The codes which fulfill the Singleton bound with equality are called *maximum distance separable* (MDS) codes.

Definition 3 (Reed–Solomon code). *Given n distinct elements (called locators) α_i, $i = 0, 1, \ldots, n-1$ of the field $\mathbb{F}_q$, a Reed–Solomon (RS) code $\mathcal{RS}(n, k)$ of length n and dimension k over $\mathbb{F}_q$ consists of the following codewords*

$$\mathcal{C}_{RS} = \{(m(\alpha_0), m(\alpha_1), \ldots, m(\alpha_{n-1})) \ : \ m(x) \in \mathbb{F}_q[x] \text{ and } \deg m(x) < k\}. \tag{3.1}$$

Remarks. The classical definition of RS codes usually assumes $\alpha_i \neq 0, \forall i$. A classical RS code is called *primitive* if the set of locators consists of all nonzero elements of $\mathbb{F}_q$, hence the length of the primitive RS code is $n = q - 1$. Denote by α the primitive element in $\mathbb{F}_q$, a primitive RS code with locator $\alpha_i = \alpha^i$, $i = 0, 1, \ldots, n-1$ is a cyclic code. If zero element is included in the list of locators, then the RS code is called *extended* RS code. The most interesting part for practice is the extension of primitive RS code, having all q elements of $\mathbb{F}_q$ as locators and hence $n = q$. In this work, we will consider both primitive RS codes and the RS codes of length q which for short is called extended RS codes.

The k information symbols are the coefficients m_i of the message polynomial $m(x) = \sum_{i=0}^{k-1} m_i x^i$ where $m_i \in \mathbb{F}_q$. The RS codes are a subclass of the evaluation codes. The RS codes are MDS, since the Hamming distance of an RS code is $d = n - k + 1$, which fulfills the Singleton bound with equality.

The RS codes can be encoded using Definition 3 which is one of the non-systematic encoding methods, we will use this definition for RS codes through our work. One can also encode the RS codes systematically, which means after encoding the information symbols are directly some elements in the codeword. Different encoding methods can be employed for the same code, resulting in different mappings from information symbols to codewords. We refer [Bos99] for other encoding methods.

From Definition 3, the evaluation of $m(x)$ at n points gives the RS codeword $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$, which can be seen as the message vector $\boldsymbol{m} = (m_0, m_1, \ldots, m_{k-1})$ multiplied with a $k \times n$ Vandermonde matrix

$$G_{RS} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{n-1} \\ \alpha_0^2 & \alpha_1^2 & \ldots & \alpha_{n-1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^{k-1} & \alpha_1^{k-1} & \ldots & \alpha_{n-1}^{k-1} \end{pmatrix}. \tag{3.2}$$

The codeword is, therefore, obtained by

$$\boldsymbol{c} = \boldsymbol{m} G_{RS}. \tag{3.3}$$

We call G_{RS} a *generator matrix* of the RS code.

We denote by H_{RS} a *parity check matrix* of the code, where H_{RS} has full rank and $H_{RS}\boldsymbol{c}^T = 0$, $\forall \boldsymbol{c} \in \mathcal{C}$. To obtain a parity check matrix H_{RS} of an RS code, replace $\boldsymbol{c}$ by $\boldsymbol{m}G_{RS}$ from (3.3), we have $H_{RS}G_{RS}^T\boldsymbol{m}^T = 0$ and hence $H_{RS}G_{RS}^T = 0$ for nonzero information words. The following Lemma 2 shows a parity check matrix of the primitive RS code.

Lemma 2. *Consider a primitive RS code $\mathcal{RS}(n,k)$ from Definition 3, having all nonzero elements α_i, $i=0,\ldots,n-1$ of $\mathbb{F}_q$, as code locators. Then the following matrix is a parity check matrix of the code*

$$H_{RS} = \begin{pmatrix} \alpha_0^{d-1} & \alpha_1^{d-1} & \ldots & \alpha_{n-1}^{d-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_0^2 & \alpha_1^2 & \ldots & \alpha_{n-1}^2 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_{n-1} \end{pmatrix}. \tag{3.4}$$

where $d = n-k+1$.

Proof: Notice that the matrices G_{RS} in (3.2) and H_{RS} in (3.4) are Vandermonde matrices, hence both have full rank. To prove that H_{RS} is a parity check matrix of the RS code, we need to show $H_{RS}G_{RS}^T = 0$, i.e., each row $\boldsymbol{h}_s = (\alpha_0^s, \alpha_1^s, \ldots, \alpha_{n-1}^s)$, $s = 1,\ldots,d-1$ of H_{RS} is orthogonal to each row $\boldsymbol{g}_t = (\alpha_0^t, \alpha_1^t, \ldots, \alpha_{n-1}^t)$, $t = 0,\ldots,k-1$ of G_{RS}. The inner product of $\boldsymbol{h}_s$ and $\boldsymbol{g}_t$ is

$$< \boldsymbol{h}_s, \boldsymbol{g}_t >= \sum_{i=0}^{n-1} \alpha_i^{s+t}, \text{ for } s = 1,\ldots,d-1;\ t = 0,\ldots,k-1. \tag{3.5}$$

Denote by α the primitive element of the field $\mathbb{F}_q$. For a primitive RS code, all nonzero locators can be written as $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$. Then (3.5) becomes

$$< \boldsymbol{h}_s, \boldsymbol{g}_t >= \sum_{i=0}^{n-1} (\alpha^{s+t})^i = \frac{\alpha^{(s+t)n}-1}{\alpha^{s+t}-1}. \tag{3.6}$$

Since $\alpha^n = 1$, the numerator in (3.6) is 0. From (3.5), $1 \le s+t \le (d-1)+(k-1) = n-1$, hence the denominator $\alpha^{s+t}-1 \neq 0$ in (3.6). We have $< \boldsymbol{h}_s, \boldsymbol{g}_t >= 0$. The statement of the lemma follows, since H_{RS} has full rank $n-k$ and $H_{RS}G_{RS}^T = 0$. □

Note that the order of rows in the parity check matrix can be arbitrary. We selected the parity check matrix H_{RS} in (3.4), which later allows accelerating decoding of extended RS codes.

We already discussed that encoding using Definition 3 is the same as using (3.3). Actually, evaluation of a polynomial in Definition 3 is also equivalent to the *discrete Fourier transform*.

Definition 4 (Discrete Fourier transform). *Let α be a primitive element of the field $\mathbb{F}_q$. Consider two polynomials $a(x) = \sum_{i=0}^{n-1} a_i x^i$, $A(x) = \sum_{i=0}^{n-1} A_i x^i \in \mathbb{F}_n[x]$. The discrete Fourier transform (DFT) is defined by*

$$a_i = A(\alpha^i),\ i = 0,\ldots,n-1.$$

Then, the inverse discrete Fourier transform (IDFT) is

$$A_j = \frac{1}{n} a(\alpha^{-j}),\ j = 0,\ldots,n-1.$$

We say that the polynomial $a(x)$ is in time domain and $A(x)$ in frequency domain. The transform is denoted by

$$a(x) \circ\!\!-\!\!\bullet\ A(x).$$

The *convolution property* of the Fourier transform [Bos99, Theorem 3.6] says that the multiplication of two polynomials in one domain corresponds to element-wise multiplication in the transform domain.

When we discuss Fourier transform, by default, we use the lower case letters to represent the expressions used in the time domain, and upper case letter in frequency domain. For instance, for RS codes, the transmitted codeword $c(x)$ or $\boldsymbol{c}$ is in time domain. The transformed polynomial $C(x)$ is in frequency domain, and is the same as $m(x)$ in Definition 3.

3.2 Decoding RS Codes

We assume the following *channel model*. The codeword vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1})$ or polynomial $c(x) = \sum_{i=0}^{n-1} c_i x^i$ is transmitted over a q-ary channel and the word $\boldsymbol{r} = (r_0, \ldots, r_{n-1}) \in \mathbb{F}_q^n$ or $r(x) = \sum_{r=0}^{n-1} r_i x^i \in \mathbb{F}[x]$ is received. The difference of the received word and the codeword is the error word $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{c} \in \mathbb{F}_q$ or $e(x) = \sum_{i=0}^{n-1} e_i x^i \in \mathbb{F}[x]$. The number of nonzero elements in the vector $\boldsymbol{e}$ is the error weight. The codeword, the error word, and the received word in transform domain are denoted by $C(x)$, $E(x)$, and $R(x)$, and $R(x) = C(x) + E(x)$ holds.

Assume that t errors are at positions $\mathcal{E} = \{i_1, \ldots, i_t\}$ and define the *error locator polynomial* (ELP) as follows

$$\Lambda(x) = \Lambda_0 + \Lambda_1 x + \cdots + \Lambda_t x^t = \prod_{i \in \mathcal{E}} (x - \alpha_i) \tag{3.7}$$

where $\Lambda_t = 1$. For decoding, we consider non-extended RS codes with $\alpha_i \neq 0$, hence $\Lambda_0 = (-1)^t \prod_{i \in \mathcal{E}} \alpha_i \neq 0$. An equivalent error locator polynomial is $\Lambda'(x) = \Lambda(x)/\Lambda_0$ where the constant term is 1. We can use $\Lambda(x)$ and $\Lambda'(x)$ equivalently later for decoding, since they have the same roots. The IDFT $\lambda(x)$ of the ELP directly shows the error positions as the power of each term $\lambda_i x^i$, for $\lambda_i = 0$. Therefore, we have $\lambda_i e_i = 0, i = 0, \ldots, n-1$ in time domain, and according to the convolution property,

$$\Lambda(x)E(x) \equiv 0 \mod x^n - 1 \tag{3.8}$$

in frequency domain.

Given a received vector $\boldsymbol{r}$ and the parity check matrix H_{RS} from (3.4), the syndrome is defined as follows

$$\boldsymbol{S} = (S_0, S_1, \ldots, S_{d-2}) = \boldsymbol{r} H_{RS}^T. \tag{3.9}$$

and $S(x) = \sum_{i=0}^{d-2} S_i x^i$.

Despite many of our results hold for arbitrary RS codes, to simplify the description, from now on, we will consider the primitive RS codes with locators $\alpha_i = \alpha^i$ for $i = 0, \ldots, n-1$ where α is a primitive element of the field.

Lemma 3. *Assume that the primitive RS code locators $\alpha_i = \alpha^i,\ i = 0, \ldots, n-1$ where α is a primitive element of $\mathbb{F}$. Given a received word $R(x) = \sum_{i=0}^{n-1} R_i x^i$ in frequency domain, the syndrome sequence $S_0, \ldots, S_{d-2}$ is proportional to the sequence $R_k, R_{k+1}, \ldots, R_{n-1}$, i.e., $S_j = \beta R_{j+k}$ with $\beta \in \mathbb{F}$ and $\beta \neq 0$ for $j = 0, \ldots, d-2$.*

Proof: By multiplying the received vector $\boldsymbol{r} = (r_0, r_1, \ldots, r_{n-1})$ and the parity check matrix H_{RS}^T, each element in syndrome $\boldsymbol{S}$ is

$$S_j = \sum_{i=0}^{n-1} r_i \alpha_i^{d-1-j},\ j = 0, \ldots, d-2.$$

Since α_i can be written as a power of the primitive element, the syndrome can be rewritten as

$$S_j = \sum_{i=0}^{n-1} r_i (\alpha^{d-1-j})^i = r(\alpha^{d-1-j}),\ j = 0, \ldots, d-2. \tag{3.10}$$

The polynomial $R(x)$ is obtained from $r(x)$ by DFT:

$$R_i = \frac{1}{n} r(\alpha^{-i}),\ i = 0, \ldots, n-1. \tag{3.11}$$

Compare (3.10) and (3.11), R_i is proportional to S_j when $\alpha^{-i} = \alpha^{d-1-j}$, i.e., $i = j+1-d+n$, because we need i to be positive. Since j runs from 0 to $d-2$, i runs from $n-d+1$ to $n-1$ which is from k to $n-1$ for RS codes. Then the statement follows. □

In the dissertation, the syndrome S_j and received word in transform domain R_{j+k} are equivalent since they are proportional. One can obtain the syndrome by multiplying the received word with the parity check matrix, or by calculating the last $(n-k)$ coefficients in the DFT of received word.

As we mentioned before, $R(x) = C(x) + E(x)$. Since $C(x)$ has at most degree $k-1$, we have $S_i \equiv R_{j+k} = E_{j+k}$ for $j = 0, \ldots, d-2$. From (3.8), one can write n homogeneous equations in a matrix form. We take only the syndrome-dependent part and obtain the following $n-k-t$ equations

$$\begin{array}{rc} x^{n-1}: & S_{d-2}\Lambda_0 + S_{d-3}\Lambda_1 + \cdots + S_{d-t-2}\Lambda_t = 0 \\ x^{n-2}: & S_{d-3}\Lambda_0 + S_{d-4}\Lambda_1 + \cdots + S_{d-t-3}\Lambda_t = 0 \\ \vdots & \vdots \\ x^{t+k+1}: & S_{t+1}\Lambda_0 + S_t\Lambda_1 + \cdots + S_1\Lambda_t = 0 \\ x^{t+k}: & S_t\Lambda_0 + S_{t-1}\Lambda_1 + \cdots + S_0\Lambda_t = 0, \end{array} \tag{3.12}$$

or

$$\begin{pmatrix} S_t & S_{t-1} & \ldots & S_0 \\ S_{t+1} & S_t & \ldots & S_1 \\ \vdots & \vdots & & \vdots \\ S_{d-2} & S_{d-3} & \ldots & S_{d-t-2} \end{pmatrix} \begin{pmatrix} \Lambda_0 \\ \Lambda_1 \\ \vdots \\ \Lambda_t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

These equations are also known as *generalized Newton's identities* [MS77]. Note that there are t unknowns in $n-k-t$ equations. To have a unique solution, the matrix consisting of syndromes should have full rank, and the number of equations should not be less than the number of unknowns, i.e., $n-k-t \geq t$, or

$$t \leq \left\lfloor \frac{n-k}{2} \right\rfloor = \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{3.13}$$

Indeed, [PW72] has shown that if 3.13 is satisfied, the syndrome matrix has full rank. An iterative form of (3.12) is written as follows:

$$S_i = -\sum_{j=1}^{t} S_{i-j}\Lambda_j, \ i = t, \ldots, d-2.$$

which is described as a linear feedback shift register (LFSR). As it is indicated in Figure 3.1, each symbol of the syndrome should be generated by the linear combination of the next t syndrome symbols. The coefficients of the t syndrome symbols are the coefficients of the error locator polynomial $\Lambda(x)$. Therefore, $\Lambda(x)$ is also called the *connection polynomial* in LFSR synthesis.

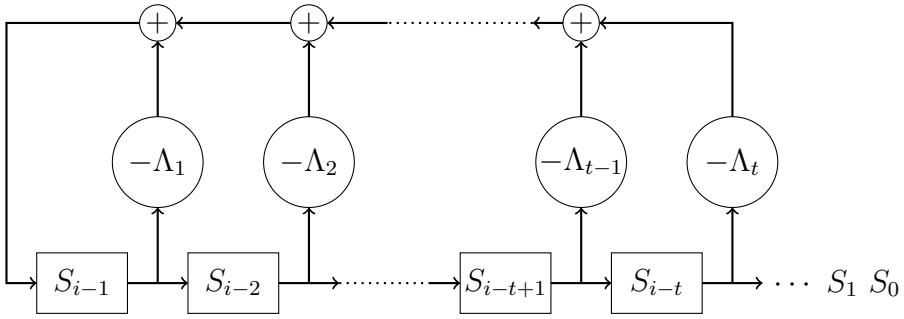


Figure 3.1: LFSR for generating a sequence $\boldsymbol{S} = \{S_i\}_{i=0}^{d-2}$.

It is also shown in [PW72] that one can write (3.12) in a polynomial form:

$$S(x)\Lambda(x) \equiv \Omega(x) \mod x^{d-1} \tag{3.14}$$

with $\deg \Omega(x) < \deg \Lambda(x)$. We call (3.14) *key equation* and $\Omega(x)$ *error evaluator polynomial* (EEP).

For decades, solving the linear feedback shift register synthesis or the key equation has always been a hot topic in decoding RS codes. The classical syndrome-based decoder usually contains two steps: firstly finding the error positions and secondly calculating error values. Algorithm 3 depicts the procedure of decoding RS codes. Representatives for the first step are the Peterson algorithm [Pet60], Sugiyama et.al algorithm [SKHN75], Belerkamp–Massey algorithm [Ber68, Mas69] etc. They aim at obtaining the ELP $\Lambda(x)$. The erroneous positions can be found by finding roots of $\Lambda(x)$, e.g., Chien search [Chi64].

Algorithm 1: Berlekamp–Massey algorithm

Input: $\boldsymbol{S} = \{S_i\}_{i=0}^{d-2}$

begin

- $t \leftarrow 0, \Lambda(x) \leftarrow 1$
- $i_a \leftarrow -1$
- $\Lambda_a(x) \leftarrow 1, t_a \leftarrow 0, \Delta_a \leftarrow 1$
- **for** *each* i *from* 0 *to* $n-k-1$ **do**
  - $\Delta \leftarrow S_i + \sum_{j=1}^{t} \Lambda_j S_{i-j}$
  - **if** $\Delta \neq 0$ **then**
    - **if** $i - i_a \leq t - t_a$ **then**
      - $\Lambda(x) \leftarrow \Lambda(x) - \frac{\Delta}{\Delta_a}\Lambda_a(x)x^{i-i_a}$
    - **else**
      - $\tilde{t} \leftarrow t, \widetilde{\Lambda}(x) \leftarrow \Lambda(x)$
      - $\Lambda(x) \leftarrow \Lambda(x) - \frac{\Delta}{\Delta_a}\Lambda_a(x)x^{i-i_a}$
      - $t \leftarrow i - i_a + t_a$
      - $t_a \leftarrow \tilde{t}, \Lambda_a(x) \leftarrow \widetilde{\Lambda}(x)$
      - $\Delta_a \leftarrow \Delta, i_a \leftarrow i$

Output: Linear feedback shift–register $(\Lambda(x), t)$

After $\Lambda(x)$ is found in the first step, the second step can be implemented straightforwardly by recursively solving system of linear equations (3.8) and then transforming $E(x)$ to time domain [Gor73]. Another efficient way to find the error values is proposed by the Forney algorithm [For65]. The decoders which are mentioned above are all bounded minimum distance (BMD) decoders, since they have unique solutions and are capable to correct up to half the minimum distance errors.

There are decoders which can decode an RS code beyond half the minimum distance, such as list decoder which is based on the interpolation and factorization techniques. Welch–Berlekamp algorithm [WB86], Sudan algorithm [Sud97], and Guruswami–Sudan algorithm [GS99] can decode the RS code by listing all codewords inside the decoding radius around the received word. A list of error locator polynomials is given by Wu algorithm [Wu08].

Recently, a unified view of the above algebraic decoding algorithms is proposed by Bossert and Bezzateev in [BB13].

In our work, we consider only bounded distance decoders. List decoders are not considered.

Among those methods based on the syndrome, Berlekamp–Massey algorithm and Sugiyama et.al algorithm outperform others in their simplicity and efficiency. Both algorithms have complexity $\mathcal{O}(n^2)$ operations in $\mathbb{F}$. These algorithms are shown in Algorithm 1 and Algorithm 2. In Line 6 of Algorithm 2, the `Quotient` function computes the quotient $q_i(x)$ of $r_{i-1}/r_r(x)$. In practice, it is not necessary to compute $f_i(x)$ through Algorithm 2.

Algorithm 2: Sugiyama et.al algorithm

Input: $S(x) = \sum_{i=0}^{d-2} S_i x^i \in \mathbb{F}[x]$

begin

$\begin{pmatrix} r_0(x) & f_0(x) & g_0(x) \\ r_1(x) & f_1(x) & g_1(x) \end{pmatrix} \leftarrow \begin{pmatrix} x^{d-1} & 1 & 0 \\ S(x) & 0 & 1 \end{pmatrix}$

$i \leftarrow 1$

while $\deg r_i(x) > \deg g_i(x)$ **do**

$q_i(x) \leftarrow \texttt{Quotient}(r_{i-1}(x), r_i(x))$

$\begin{pmatrix} r_i(x) & f_i(x) & g_i(x) \\ r_{i+1}(x) & f_{i+1}(x) & g_{i+1}(x) \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 1 \\ 1 & -q_i(x) \end{pmatrix} \begin{pmatrix} r_{i-1}(x) & f_{i-1}(x) & g_{i-1}(x) \\ r_i(x) & f_i(x) & g_i(x) \end{pmatrix}$

$i \leftarrow i + 1$

$k \leftarrow i$

Output: $(\Lambda(x), \Omega(x), t) = (g_k(x), f_k(x), \deg g_k(x))$

We will discuss this algorithm later in Section 3.5.1. As a result one can decode RS code by Algorithm 3.

Algorithm 3: Decoding an RS code

Input: Received word $\boldsymbol{r}$

begin

Compute syndrome $\boldsymbol{S} = \boldsymbol{r} H_{RS}^T$

Run Algorithm 1 or Algorithm 2 for $\boldsymbol{S}$, and get t and $\Lambda(x)$

Find roots $\alpha^{i_1}, \ldots, \alpha^{i_t}$ of $\Lambda(x)$ in $\mathbb{F}$

Compute $\boldsymbol{c}$ by correcting t erasures in positions $i_1, \ldots, i_t$ of $\boldsymbol{r}$

Output: Codeword $\boldsymbol{c}$

3.3 Decoding IRS Codes with Berlekamp–Massey Algorithm

In the last decades, interleaved Reed–Solomon (IRS) codes were one of the popular research topics in algebraic coding theory. Related to this topic, the decoding approaches underwent several phases. A trivial thinking is to treat the IRS code as ℓ individual RS codes and decode them separately, which leads the correctable error number is only within half the smallest minimum distance of these ℓ codes. In case of burst errors, all the error positions for each RS code are the same, thus, collaborative decoding can correct more errors than the

trivial method with the same complexity. In our work, to deal with burst errors, we consider the collaborative decoding approach, which means to find the common error locations for all ℓ RS codes at first and then evaluate the error values separately. As mentioned in the previous chapter, the first step is more complicated than the second one, so we mainly focus on finding the error locations, meanwhile we keep the complexity low in collaborative decoding of IRS codes.

Given ℓ Reed–Solomon codes $\mathcal{RS}(n, k^{(l)})$, $l = 1, 2, \ldots, \ell$, over $\mathbb{F}_q$ of length n and dimensions $k^{(l)}$ defined by generator matrices $G_{RS}^{(l)}$ of the form (3.2) or by parity check matrices $H_{RS}^{(l)}$ of the form (3.4), the interleaved RS (IRS) code $\mathcal{IRS}(n, k^{(1)}, \ldots, k^{(\ell)})$ consists of all $\ell \times n$ matrices C_{IRS}

$$C_{IRS} = \begin{pmatrix} \boldsymbol{c}^{(1)} \\ \boldsymbol{c}^{(2)} \\ \vdots \\ \boldsymbol{c}^{(\ell)} \end{pmatrix} = \begin{pmatrix} c_0^{(1)} & c_1^{(1)} & \ldots & c_{n-1}^{(1)} \\ c_0^{(2)} & c_1^{(2)} & \ldots & c_{n-1}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ c_0^{(\ell)} & c_1^{(\ell)} & \ldots & c_{n-1}^{(\ell)} \end{pmatrix},$$

where $\boldsymbol{c}^{(l)} \in \mathcal{RS}(n, k^{(l)})$. If $k^{(l)}$ are all the same, then the IRS codes are called *homogeneous*, otherwise, they are *heterogeneous*. If every row $\boldsymbol{c}^{(l)}$ is a codeword of the extended RS code, then such construction forms interleaved extended RS (IERS) codes. We will focus on decoding IRS codes in this section, and discuss IERS codes later in Section 4.2 and Section 4.4.

We assume the following *interleaving scheme* and *channel model*. We transmit a code matrix C and receive a $\ell \times n$ matrix R over $\mathbb{F}_q$, we say that the error matrix is $E = R - C$ over $\mathbb{F}_q$. A burst error which occurred in the channel destroys all components in a column of the received word. Figure 3.2 depicts the burst error model for a heterogeneous IRS codeword. The burst error weight is the number of columns which are corrupted, i.e., the number of nonzero columns in the matrix E. We assume that the channel is q^ℓ-ary symmetric.

The IRS code can be efficiently decoded as follows. By default, the index l always runs from 1 to ℓ. Given a received matrix R, denote by $\boldsymbol{r}^{(l)}$ rows of R and compute the syndrome vectors $\boldsymbol{S}^{(l)}$ and polynomials $S^{(l)}(x)$ for every component RS code as follows

$$\boldsymbol{S}^{(l)} = (S_0^{(l)}, S_1^{(l)}, \ldots, S_{d-2}^{(l)}) = \boldsymbol{r}^{(l)} H_{RS}^{(l)T} \tag{3.15}$$

and

$$S^{(l)}(x) = \sum_{i=0}^{d-2} S_i^{(l)} x^i. \tag{3.16}$$

Assume that t erroneous columns are at positions $\mathcal{E} = \{i_1, \ldots, i_t\}$ and define the *error locator* polynomial by (3.7). The syndromes and the ELP for the IRS code satisfy the following system of key equations

$$S^{(l)}(x)\Lambda(x) \equiv \Omega^{(l)}(x) \mod x^{d^{(l)}-1} \text{ for } l = 1, \ldots, \ell, \tag{3.17}$$

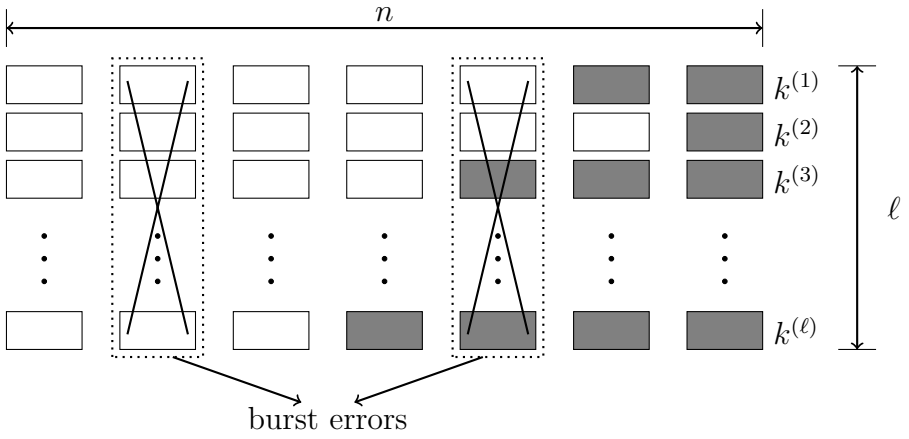


Figure 3.2: Burst error model for interleaved codes.

where $\Omega^{(l)}(x)$ is a polynomial with $\deg \Omega^{(l)}(x) < \deg \Lambda(x)$. Similar to the single sequence case in LFSR synthesis in Figure 3.1, solving the system of key equations (3.17) can be regarded as one LFSR generating ℓ sequences, which is so-called multi-sequence linear feedback shift register (MS-LFSR) synthesis (see Figure 3.3). Note that $\Lambda(x)$ in (3.17) may not be unique to generate the same syndromes. We are interested in finding a $\Lambda(x)$ with smallest degree which is the ELP. Regarding to MS-LFSR synthesis, decoding IRS codes are considered as the following problem.

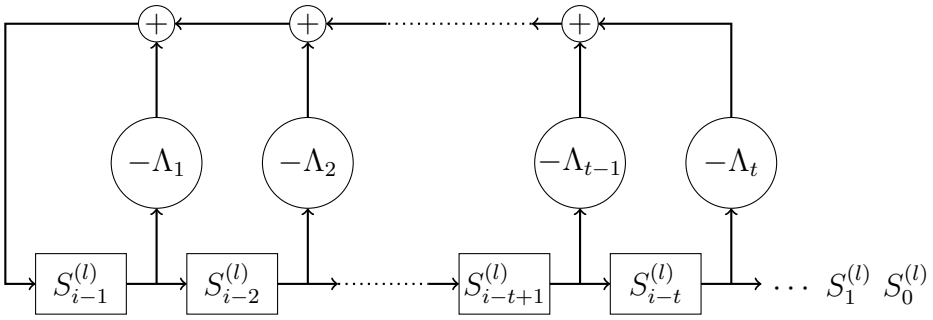


Figure 3.3: MS-LFSR synthesis.

Problem 1. *Let $\boldsymbol{S}^{(1)}, \boldsymbol{S}^{(2)}, \ldots, \boldsymbol{S}^{(\ell)}$ of length $N^{(1)}, N^{(2)}, \ldots, N^{(\ell)}$, respectively, be sequences over a field $\mathbb{F}$. Find the smallest nonnegative integer t for which there is a vector of coefficients $\boldsymbol{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_t)$ over $\mathbb{F}$ such that for $l = 1, 2, \ldots, \ell$ and for $i = t, \ldots, N^{(l)}-1$*

$$S_i^{(l)} = -\Lambda_1 S_{i-1}^{(l)} - \Lambda_2 S_{i-2}^{(l)} - \cdots - \Lambda_t S_{i-t}^{(l)}. \tag{3.18}$$

Moreover, find a vector of coefficients $\boldsymbol{\Lambda}$ which fulfills (3.18).

Known results. As mentioned before, there are two notable algorithms which solve LFSR synthesis efficiently: Berlekamp–Massey algorithm (BMA), and Sugiyama et.al algorithm

which is based on the Euclidean algorithm (EA). These two algorithms are to solve single-sequence LFSR synthesis. Due to their efficiency and simplicity, both algorithms have been generalized to the multi-sequence LFSR synthesis. There are approaches such as Feng and Tzeng's fundamental iterative algorithm (FIA) [FT85] and the generalized iterative algorithm (GIA) [FT91] which are considered as a generalization of BMA. In 1989, Feng and Tzeng proposed the generalization of extended Euclidean algorithm (GEEA) [FT89] for MS-LFSR synthesis. However, the same drawback of FIA, GIA and GEEA is that they require input sequences to have the same length. Regarding to sequences of different lengths, efficient modifications based on GIA and GEEA have been done in [SS11] and [KL13], respectively, aiming to solve the multi-sequence varying length problem — Problem 1. Details of the latter one will be discussed in Section 3.4. More recently, Nielsen proposed an easy-understandable "module minimization" approach [Nie13a] which converts Problem 1 to finding leading position at the first column in the weak Popov form of a certain-structured polynomial matrix over $\mathbb{F}[x]$. The modified GIA, the modified GEA and module minimization are all looking for an error locator polynomial, although the equivalence of these three methods are not proved, some similar intermediate behaviors/results can be found during their calculations.

An efficient solution based on GIA, or say based on the BMA, in [SS11] of Problem 1 is given by Algorithm 4.

Theorem 4 ([SS11]). *If the output of Algorithm 4 is $\Lambda(x)$, N, t; and $\Lambda^{(l)}(x)$, $n^{(l)}$, $t^{(l)}$ for $l = 1, \ldots, \ell$, then t is the length of a shortest shift-register that generates ℓ sequences $\boldsymbol{S}^{(l)}$. The connection polynomial $\Lambda(x)$ is unique if and only if $\varepsilon = 0$, where*

$$\varepsilon = \sum_{l=1}^{\ell} \varepsilon^{(l)},$$

$$\varepsilon^{(l)} = \max_{l}\{0, n^{(l)} - t^{(l)} - z^{(l)} - (N - t)\},$$

$$z^{(l)} = \max_{l}\{0, t - N^{(l)}\}.$$

Time complexity of Algorithm 4 is $\mathcal{O}(\ell N^2)$ operations in $\mathbb{F}$.

After the unique error locator polynomial is found, position of errors can be obtained by computing roots of the polynomial. When the error positions are known, the errors are transformed to erasures and can be corrected independently in every component code. As a result the IRS code can be decoded by Algorithm 5.

Theorem 5 ([SSB07]). *For an IRS code $\mathcal{IRS}(n, k^{(1)}, \ldots, k^{(\ell)})$, Algorithm 5 corrects errors of weight t up to decoding radius $t_{\max}$ with failure probability $P_f(t)$ if*

$$t \leq t_{\max} = \min\left\{ \frac{\ell}{\ell + 1} \left(n - \overline{k}\right), n - k_{\max} \right\}, \tag{3.19}$$

Algorithm 4: MS-LFSR synthesis based on BMA (Problem 1)

Input: ℓ; $\boldsymbol{S}^{(l)} = S_0^{(l)}, \ldots, S_{N^{(l)}-1}^{(l)}$ and $N^{(l)}$ for $l = 1, \ldots, \ell$

begin

  $t \leftarrow 0$, $\Lambda(x) \leftarrow 1$, $N \leftarrow \max_l\{N^{(l)}\}$

  $\delta^{(l)} \leftarrow N - N^{(l)}$, $n^{(l)} \leftarrow \delta^{(l)}$, $t^{(l)} \leftarrow 0$, $\Delta^{(l)} \leftarrow 1$, $\Lambda^{(l)}(x) \leftarrow 0$ for $l = 1, \ldots, \ell$

  for *each* n *from* 0 *to* $N-1$ **do**

    for *each* l *from* 1 *to* ℓ **do**

      if $n - t \geq \delta^{(l)}$ **then**

        $\Delta \leftarrow \sum_{j=0}^{t} \Lambda_j S_{n-\delta^{(l)}-j}^{(l)}$

        if $\Delta \neq 0$ **then**

          if $n - n^{(l)} \leq t - t^{(l)}$ **then**

            $\Lambda(x) \leftarrow \Lambda(x) - \frac{\Delta}{\Delta^{(l)}} \Lambda^{(l)}(x) x^{n-n^{(l)}}$

          else

            $\tilde{t} \leftarrow t$, $\widetilde{\Lambda}(x) \leftarrow \Lambda(x)$

            $\Lambda(x) \leftarrow \Lambda(x) - \frac{\Delta}{\Delta^{(l)}} \Lambda^{(l)}(x) x^{n-n^{(l)}}$

            $t \leftarrow n - n^{(l)} + t^{(l)}$

            $t^{(l)} \leftarrow \tilde{t}$, $\Lambda^{(l)}(x) \leftarrow \widetilde{\Lambda}(x)$, $\Delta^{(l)} \leftarrow \Delta$, $n^{(l)} \leftarrow n$

Output: $\Lambda(x)$, N, t; and $\Lambda^{(l)}(x)$, $n^{(l)}$, $t^{(l)}$ for $l = 1, \ldots, \ell$

where

$$\overline{k} = \frac{1}{\ell} \sum_{l=1}^{\ell} k^{(l)}, \quad k_{\max} = \max_l\{k^{(l)}\}$$

are the average and the maximum dimension of the ℓ Reed–Solomon codes respectively, and

$$P_f(t) \leq \widehat{P}_f(t) = \gamma q^{-(l+1)(t_{\max}-t)-1}\,, \tag{3.20}$$

$$\gamma = \left(\frac{q^l - \frac{1}{q}}{q^l - 1}\right)^t \frac{q}{q-1} \approx 1.$$

The bound (3.20) can be also written as

$$P_f(t) \leq \widehat{P}_f(t) = \gamma q^{-(\#\text{ of equations}-t)-1}\,, \tag{3.21}$$

using the numbers of equations in the system (3.18).

Algorithm 5: Decoding an IRS code

Input: Received words $\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(\ell)}$

begin

  Compute syndromes $\boldsymbol{S}^{(l)} = \boldsymbol{r}^{(l)} H_{RS}^{(l)T}$ for $l = 1, \ldots, \ell$

  Run Algorithm 4 for ℓ sequences $\boldsymbol{S}^{(l)}$, $l = 1, \ldots, \ell$ of length $N^{(l)} = n - k^{(l)}$, and get $\Lambda(x)$ and t

  if $\varepsilon \neq 0$ **then**

    output decoding failure and stop

  Find roots $\alpha^{i_1}, \ldots, \alpha^{i_\tau}$ of $\Lambda(x)$ in $\mathbb{F}$

  if *number of roots not equal* t **then**

    output decoding failure and stop

  for $l = 1, \ldots, \ell$ **do**

    Compute $\boldsymbol{c}^{(l)}$ by correcting t erasures in positions $i_1, \ldots, i_t$ of $\boldsymbol{c}^{(l)}$

Output: Codewords $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(\ell)}$ or decoding failure

3.4 Decoding IRS Codes with Euclidean Algorithm

Thr Euclidean algorithm is often used to find the greatest common divisor between two integers or polynomials. In 1975, Sugiyama et.al applied extended EA for decoding RS codes to find the error locator polynomial given $S(x)$, c.f. (3.14) and Algorithm 2. In 1989, Feng and Tzeng [FT89] generalized the Euclidean algorithm to solve Problem 1, given $S^{(l)}(x), l = 1, \ldots, \ell$. However, Feng–Tzeng's algorithm only concentrates on the homogeneous case ($N^{(1)} = N^{(2)} = \cdots = N^{(\ell)}$), without any hints or ideas of how to adapt it for the heterogeneous case. In order to fit Feng–Tzeng's algorithm to decode any IRS codes, one idea is presented in [ZL10a] where the syndrome sequences of different lengths are converted into several sequences of the same length. Assume the shortest sequence has length $N_{\min}$, one can write any long sequence as some short sequences of length $N_{\min}$. If lengths of the syndrome sequences differ very much, then the rewriting significantly increases the number of interleaved words and hence the decoding complexity. In order to the complexity low, in this section, we present a modification of [FT89] that does not need this transformation of syndromes.

Problem 1 can be equivalently rewritten as follows.

Problem 2. *Let all prerequisites be the same as in Problem 1. Find a polynomial* $\Lambda(x) \in \mathbb{F}[x]$ *with smallest degree such that for* $l = 1, 2, \ldots, \ell$

$$S^{(l)}(x)\Lambda(x) = \Omega^{(l)}(x) + P^{(l)}(x)x^{N^{(l)}} \tag{3.22}$$

holds for some polynomials $P^{(l)}(x)$ *with*

$$\deg \Omega^{(l)}(x) < \deg \Lambda(x). \tag{3.23}$$

In this case, we say $\Lambda(x)$ *generates* $S^{(l)}(x)$ *for* $l = 1, \ldots, \ell$.

The term $P^{(l)}(x)x^{N^{(l)}}$ is just an alternative representation to the modulo operation in (3.17). We replace the indeterminate x by x^ℓ, i.e.,

$$S^{(l)}(x^\ell)\Lambda(x^\ell) = \Omega^{(l)}(x^\ell) + P^{(l)}(x^\ell)x^{N^{(l)}\ell}.$$

Then, we shift the coefficients of $S^{(l)}(x^\ell)$ in a way such that there is no overlapping when adding all the ℓ equations together. This can be done by multiplying the l-th equation by x^{l-1}, and we obtain the *concatenated key equation* as follows

$$\widetilde{S}(x)\Lambda(x^\ell) = \widetilde{\Omega}(x) + \sum_{l=1}^{\ell} P^{(l)}(x^\ell)x^{N^{(l)}\ell+l-1} \tag{3.24}$$

where

$$\widetilde{S}(x) = \sum_{l=1}^{\ell} S^{(l)}(x^\ell)x^{l-1}, \tag{3.25}$$

$$\widetilde{\Omega}(x) = \sum_{l=1}^{\ell} \Omega^{(l)}(x^\ell)x^{l-1}. \tag{3.26}$$

The cross breeding of ℓ syndromes produces one "long" syndrome which is a vector of coefficients in $\widetilde{S}(x)$. Figure 3.4 depicts an alternative of MS-LFSR. For every sequence shift, only symbols from the same syndrome are coped with the connection polynomial $\Lambda(x)$. Therefore, it is straightforward to see that (3.24) is equivalent to (3.22).

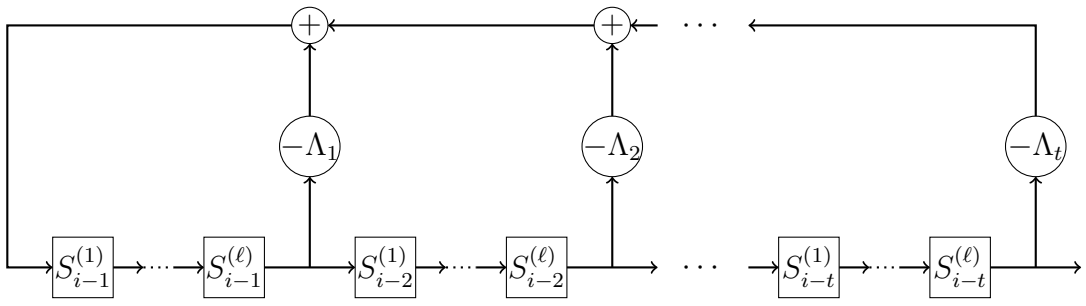


Figure 3.4: An equivalence of MS-LFSR synthesis based on (3.24).

The following lemma states how the degree conditions in Problem 2 change correspondingly for (3.24).

Lemma 6. *$\Lambda(x)$ generates sequences $S^{(l)}(x)$ for $l = 1, \ldots, \ell$, if and only if*

$$\deg \widetilde{\Omega}(x) < \ell \deg \Lambda(x). \tag{3.27}$$

Proof: Let (3.27) holds, first we prove that (3.27) is a sufficient condition to obtain (3.23). Let us assume that $\Omega^{(l^*)}(x)$ has the maximum degree among $\Omega^{(l)}(x)$ for all l. Then

$$\deg \widetilde{\Omega}(x) = \ell \deg \Omega^{(l^*)}(x) + l^* - 1 < \ell \deg \Lambda(x).$$

It follows immediately that,

$$\deg \Lambda(x) - \deg \Omega^{(l^*)}(x) > \frac{l^* - 1}{\ell} \geq 0.$$

Hence (3.23) is satisfied.

Conversely, if $\forall l$, $\deg \Omega^{(l)}(x) < \deg \Lambda(x)$, then

$$\ell \deg \Omega^{(l)}(x) \leq \ell(\deg \Lambda(x) - 1). \tag{3.28}$$

Since the summation in (3.26) can not have degree higher than $\ell \deg \Omega^{(l^*)}(x) + l^* - 1$, together with (3.28) we have

$$\deg \widetilde{\Omega}(x) \leq \ell \deg \Omega^{(l^*)}(x) + l^* - 1 \leq \ell(\deg \Lambda(x) - 1) + \ell - 1 < \ell \deg \Lambda(x).$$

□

Hence, the following problem is equivalent to Problem 2.

Problem 3. *Let all prerequisites be the same as in Problem 1. Find a polynomial $\Lambda(x) \in \mathbb{F}[x]$ with smallest degree such that* (3.24) *is satisfied with restriction* (3.27).

3.4.1 Generalized Extended Euclidean Algorithm (GEEA)

As we mentioned before, Feng–Tzeng's generalized extended Euclidean algorithm (GEEA) [FT89] solves the LFSR problem with sequences of the same length. In particular, the generalized division `GenDiv` plays the main role in the GEEA.

Definition 5 (Congruence relation). *Given $a(x)$, $b(x) \in F[x]$ and a nonzero integer ℓ, we say that $a(x)$ is congruent to $b(x)$, denoted by $a(x) \sim b(x)$, if and only if*

$$\deg a(x) \equiv \deg b(x) \mod \ell,$$

If the polynomials are not congruent, we denote it by $a(x) \nsim b(x)$.

Thus, the congruence relation splits $\mathbb{F}[x]$ into ℓ *congruence classes* $[x^v]$, $v = 0, \ldots, \ell$ where $[x^v] = \{a(x) | a(x) \in \mathbb{F}[x] \text{ and } \deg a(x) \equiv v \mod \ell\}$.

Definition 6 (Congruence set). *Given polynomials $b^{(l)}(x)$ for $l = 1, \ldots, \ell$ over $\mathbb{F}[x]$. If every polynomial belongs to a distinct congruence class, e.g., $b^{(l)}(x) \in [x^l]$, then $\forall l$, $b^{(l)}(x)$ form a (full) congruence set.*

To run the GEEA, we need a modified division of two polynomials. Feng and Tzeng considered this operation only for polynomials which are congruent to each other. They modified the usual polynomial division such that the quotient contains powers which are only multiple of ℓ. Let $a(x)$, $b(x)$ be two polynomials over $\mathbb{F}[x]$. If $a(x) \sim b(x)$ and $\deg a(x) \geq b(x)$, then the *modified division* (`ModDiv`, Algorithm 6) finds uniquely a quotient $Q(x^\ell) \neq 0$ and a remainder $r(x)$ such that

$$a(x) = Q(x^\ell)b(x) + r(x),$$

where $\deg r(x) < \deg b(x)$ if $r(x) \sim b(x)$. Note that the difference from the usual polynomial division is that, if $r(x) \nsim b(x)$ then $\deg r(x)$ might be greater than $\deg b(x)$ when the algorithm stops.

Algorithm 6: Modified division function `ModDiv`

Input: ℓ; $a(x), b(x) \neq 0$ in $\mathbb{F}[x]$, and $\deg a(x) \equiv \deg b(x) \mod \ell$
begin
  if $\deg a(x) < \deg b(x)$ **then**
    exchange $a(x)$ and $b(x)$
  $\begin{pmatrix} Q(x^\ell) & r(x) \end{pmatrix} \leftarrow \begin{pmatrix} 0 & a(x) \end{pmatrix}$
  while $r(x) \sim b(x)$ *and* $\deg r(x) \geq \deg b(x)$ **do**
    $\delta = \deg r(x) - \deg b(x)$
    $\begin{pmatrix} Q(x^\ell) & r(x) \end{pmatrix} \leftarrow \begin{pmatrix} 1 & x^\delta \end{pmatrix} \begin{pmatrix} Q(x^\ell) & r(x) \\ 1 & -b(x) \end{pmatrix}$
Output: $Q(x^\ell), r(x)$

Now let us introduce *generalized division* for $\ell + 1$ polynomials. Given ℓ polynomials $b^{(l)}(x) \in \mathbb{F}[x]$ for $l = 1, \ldots, \ell$ which form a congruence set, and a polynomial $a(x) \in \mathbb{F}[x]$ with $\deg a(x) \geq \deg b^{(l)}(x)$ $\forall l$, we can apply `ModDiv` iteratively and obtain the following form

$$a(x) = \sum_{l=1}^{\ell} Q^{(l)}(x^\ell)b^{(l)}(x) + r(x) \tag{3.29}$$

where $r(x) \sim b^{(v)}(x)$ for some v and $\deg r(x) < \deg b^{(v)}(x)$.

From now on, we will use another expression which is equivalent to (3.29). Assume $a(x) \sim b^{(u)}(x)$ for some $u \in [1, \ldots, \ell]$. Let us rewrite (3.29) as

$$r(x) = -Q^{(u)}(x^\ell)b^{(u)}(x) + a(x) + \sum_{l \neq u} -Q^{(l)}(x^\ell)b^{(l)}(x). \tag{3.30}$$

Let $A(x) = b^{(u)}(x)$, $B^{(l)}(x) = b^{(l)}(x)$ for all $l \neq u$, and $B^{(u)}(x) = a(x)$. Let $p(x^\ell) = -Q^{(u)}(x^\ell)$, $q^{(l)}(x^\ell) = -Q^{(l)}(x^\ell)$ for all $l \neq u$, and $q^{(u)}(x^\ell) = 1$. Then (3.30) becomes

$$r(x) = p(x^\ell)A(x) + \sum_{l=1}^{\ell} q^{(l)}(x^\ell)B^{(l)}(x), \tag{3.31}$$

where $r(x) \sim A(x)$ and $\deg r(x) < \deg A(x)$, or $r(x) \sim B^{(v)}(x)$ for some $v \neq u$ and $\deg r(x) < \deg B^{(v)}(x)$. With the new notations, the problem is as follows. Given $\ell + 1$ polynomials $A(x)$ and $B^{(l)}(x)$ ($A(x) \sim B^{(u)}(x)$ for some u and $\deg A(x) \leq \deg B^{(u)}$), we would like to express the remainder $r(x)$ as a combination of these $\ell + 1$ polynomials by (3.31) like in extended Euclidean algorithm.

Algorithm 7 `GenDiv` gives the solution of the problem. It finds uniquely polynomials $r(x)$, $p(x^\ell)$, and $q^{(l)}(x^\ell)$ for all l such that (3.31) is satisfied. Line 6 and the `while` loop form the generalized division which is as shown in (3.29). If $\ell = 1$, `GenDiv` is simplified to `ModDiv`.

Algorithm 7: Generalized division function `GenDiv`

1 **Input**: ℓ; $A(x)$, $B^{(l)}(x) \in \mathbb{F}[x]$ and $B^{(l)}(x) \in [x^{l-1}]$ for $l = 1, \ldots, \ell$.
2 **begin**
3 $u \leftarrow (\deg A(x) \mod \ell) + 1$
4 $a(x) \leftarrow B^{(u)}(x)$; $b^{(u)}(x) \leftarrow A(x)$, and $b^{(l)}(x) \leftarrow B^{(l)}(x)$ for all $l \neq u$
5 $Q^{(l)}(x^\ell) \leftarrow 0$ for all l
6 $r(x) \leftarrow a(x)$
7 **while** $\deg r(x) \geq \deg b^{(u)}(x)$ **do**
8 $\widetilde{Q}^{(u)}(x^\ell), r(x) \leftarrow \texttt{ModDiv}(r(x), b^{(u)}(x))$
9 $Q^{(u)}(x^\ell) \leftarrow Q^{(u)}(x^\ell) + \widetilde{Q}^{(u)}(x^\ell)$
10 $Q^{(l)}(x^\ell) \leftarrow Q^{(l)}(x^\ell)$ for all $l \neq u$
11 $u \leftarrow (\deg r(x) \mod \ell) + 1$
12 $p(x^\ell) \leftarrow -Q^{(u)}(x^\ell)$
13 $q^{(u)}(x^\ell) \leftarrow 1$
14 $q^{(l)}(x^\ell) \leftarrow -Q^{(l)}(x^\ell)$ for all $l \neq u$
15 **Output**: $r(x), p(x^\ell)$, $q^{(l)}(x^\ell)$, for $l = 1, \ldots, \ell$

In the original Feng–Tzeng's GEEA, all the input sequences are assumed to have the same length N, and $b_0^{(l)}(x)$ is initialized as $x^{N\ell+l-1}$ $\forall l$. To solve MS-LFSR synthesis problem, all sequences may have different lengths. Hence, GEEA needs to be modified. According to (3.24), we use $b_0^{(l)}(x) = x^{N^{(l)}\ell+l-1}$ in Feng–Tzeng's GEEA where $N^{(l)}$ is the length of each sequence. Without changing `ModDiv` and `GenDiv`, the modified algorithm (Algorithm 8) can

Algorithm 8: Modified FengTzeng's algorithm for MS-LFSR synthesis

1 **Input**: ℓ; sequences $S^{(l)}(x)$ with length $N^{(l)}$, for $l = 1, \ldots, \ell$
2 **begin**
3 $r_0(x) \leftarrow \widetilde{S}(x)$ according to (3.25), $\underline{b_0^{(l)}(x) \leftarrow x^{N^{(l)}\ell+l-1}}$ $\forall l$
4 $U_0(x) \leftarrow 1$, $V_0^{(l)}(x) \leftarrow 0$ $\forall l$
5 $j \leftarrow 0$
6 **while** $\deg r_j(x) \geq \deg U_j(x^\ell)$ **do**
7 $r_{j+1}(x), p_{j+1}(x^\ell), q_{j+1}^{(l)}(x^\ell) \leftarrow \texttt{GenDiv}(r_j(x), b_j^{(l)}(x))$
8 $v_{j+1} \leftarrow \deg r_j(x) \mod \ell + 1$
9 $b_{j+1}^{(v_{j+1})}(x) \leftarrow r_j(x)$
10 $U_{j+1}(x) \leftarrow p_{j+1}(x)U_j(x) + \sum_{l=1}^{\ell} q_{j+1}^{(l)}(x)V_j^{(l)}(x)$
11 $V_{j+1}^{(v_{j+1})}(x) \leftarrow U_j(x)$
12 $j \leftarrow j + 1$
13 $k \leftarrow j$
14 **Output**: Monic polynomial $\delta U_k(x)$

be immediately applied to the LFSR synthesis with sequences of different lengths. And the complexity is the same as the original GEEA's since only the longest length is considered.

The underlined part in Line 3 is the only difference from the original Feng–Tzeng's GEEA. In Line 7 of every `while` loop, given $\ell + 1$ polynomials, by running `GenDiv` repeatedly, the remainder $r_{j+1}(x)$ and quotients $p_{j+1}(x^\ell)$, $q_{j+1}^{(l)}(x^\ell)$ are obtained satisfying

$$r_{j+1}(x) = p_{j+1}(x^\ell)r_j(x) + \sum_{l=1}^{\ell} q_{j+1}^{(l)}(x^\ell)b_j^{(l)}(x) \text{ for } j = 0, 1, \ldots. \tag{3.32}$$

In Line 8 and Line 9, we find out the polynomial $b_j^{(v_{j+1})}(x) \sim r_j(x)$ in $b_j^{(l)}(x)$ and replace this polynomial by $r_j(x)$ for the next iteration. For all $l \neq v_{j+1}$, $b_{j+1}^{(l)}(x) = b_j^{(l)}(x)$.

Algorithm 8 finds the error locator polynomial $\delta U_k(x)$. In fact, this is also a minimal solution.

Theorem 7. *To solve the MS-LFSR synthesis problem, the polynomials $\delta U_k(x)$ which Algorithm 8 outputs is a $\Lambda(x)$ with smallest degree.*

Proof: For multi-sequence varying lengths case, the proof is the same as for multi-sequence same length case. In [FT89], it is proved by contradiction that, there exist no other connection polynomial with degree less than $\delta U_k(x)$. In our case, input sequences have different lengths, however, the changes of degree of $U_j(x^\ell)$ and $r_j(x)$ are the same as that in the case of the same length, i.e., the degree conditions don not alter in terms of length of the input sequence. □

The decoding radius and failure probability of using Algorithm 8 to decode IRS codes are the same as to decode by using Algorithm 4 based on BMA, which is already stated in Theorem 5.

Example 1. *Let us transmit an IRS code where each row is a primitive RS codeword, i.e.,* $\boldsymbol{c}_1 \in \mathcal{RS}_1(10,3)$ *and* $\boldsymbol{c}_2 \in \mathcal{RS}_2(10,5)$ *codes. The syndromes of the received words*

$$\begin{aligned} \boldsymbol{r}^{(1)} &= (5,5,1,0,8,0,2,1,1,9), \\ \boldsymbol{r}^{(2)} &= (0,4,3,0,10,0,10,6,10,3) \end{aligned}$$

are calculated as follows according to (3.9)

$$\begin{aligned} \boldsymbol{S}^{(1)} &= (1,10,5,4,9,2,8), \\ \boldsymbol{S}^{(2)} &= (3,10,6,2,0). \end{aligned}$$

To proceed with the decoding, we apply these syndromes to the modified GEEA (Algorithm 8). The intermediate results are listed in Table 3.1.

j	$r_j(x)$	$p_j(x)$	$q_j^{(1)}(x)$	$q_j^{(2)}(x)$	$U_j(x)$
0	$8x^{12}+2x^{10}+9x^8+2x^7+4x^6+6x^5+5x^4+10x^3+10x^2+3x+1$	–	–	–	1
1	$x^{10}+8x^9+7x^8+5x^6+x^5+2x^4+2x^3+5x^2+8x+10$	$4x+10$	1	0	$4x+10$
2	$3x^9+6x^8+5x^7+5x^6+7x^4+10x^3+2x^2+6x+2$	$3x+10$	1	9	x^2+4x+2
3	$2x^8+3x^7+7x^6+6x^5+9x^4+8x^3+x^2+2x+8$	$7x+5$	2	1	$7x^3+9x+8$
4	$5x^7+9x^5+8x^4+10x^3+6x^2+10x+7$	$5x+2$	1	7	$2x^4+3x^3+8x^2+2x+7$

Table 3.1: Decoding IRS codes using Algorithm 8.

Since $\deg U_4(x^4) > \deg r_4(x)$*, we have* $k=4$ *and the monic error locator polynomial is* $\delta U_4(x) = x^4+7x^3+4x^2+x+9$*. It can be easily checked that* $\alpha^0, \alpha^1, \alpha^2$*, and* α^3 *are roots of* $\delta U_4(x)$ *which give the positions of errors.*

3.4.2 Complexity of Modified GEEA

Because of the GEEA, the degree of remainder $r_{j+1}(x)$ is monotonically decreasing with the increasing of the iterations. If inputs of Algorithm 8 are $r_0(x) \in \mathbb{F}[x]$ and $b_0^{(l)}(x) \in \mathbb{F}[x]$, and Algorithm 8 does not stop when $\deg r_k(x) < \deg U_k(x^\ell)$ until the remainder $r_j = 0$. We say such an algorithm is the *complete* GEEA.

Theorem 8 (Complexity of complete GEEA). *Given $\ell+1$ polynomials $r_0(x) \in \mathbb{F}[x]$, $b_0^{(l)}(x) \in \mathbb{F}[x]$ for $l = 1, \ldots, \ell$ where all $b^{(l)}(x)$ form a congruence set. Furthermore, for some $u \in [1, \ell]$, $\deg r_0(x) < \deg b^{(u)}(x)$ where $r_0(x) \sim b_0^{(u)}(x)$. Let an integer I be at least the maximum degree of these $\ell+1$ polynomials. Then the complete GEEA computes a greatest common divisor (GCD) of $r_0(x), b_0^{(l)}(x)$ in polynomial time $\mathcal{O}(I^2)$.*

Proof: Using generalized division `GenDiv`, the time T_j to perform this division at the j-th iteration in Algorithm 8 is determined by the product of $p_{j+1}(x^\ell)r_j(x)$ and $q_{j+1}^{(l)}(x^\ell)b_j^{(l)}(x)$ for $l = 1, \ldots, \ell$. Note that only every ℓ-th coefficient in $p_{j+1}(x^\ell)$ and $q_{j+1}^{(l)}(x^\ell)$ is considered. Thus,

$$T_j \leq \frac{c}{\ell} \max_l \left\{ \deg p_{j+1}(x^\ell) \deg r_j(x), \deg q_{j+1}^{(l)}(x^\ell) \deg b_j^{(l)}(x) \right\} \tag{3.33}$$

for some constant c. Since the complete GEEA requires at most $I+1 = \mathcal{O}(I)$ iterations, the total time T is bounded by

$$\begin{aligned} T = \sum_{j=0}^{I} T_j &\leq \frac{c}{\ell} \sum_{j=1}^{I} \max_l \left\{ \deg p_{j+1}(x^\ell) \deg r_j(x), \deg q_{j+1}^{(l)}(x^\ell) \deg b_j^{(l)}(x) \right\} \\ &\leq \frac{cI}{\ell} \sum_{j=0}^{I} \max_l \left\{ \deg p_{j+1}(x^\ell), \deg q_{j+1}^{(l)}(x^\ell) \right\} \end{aligned}$$

We know from (13.1) and (13.2) in [FT89] that,

$$\begin{aligned} &\deg p_{j+1}(x^\ell) = \deg b_j^{(v_j)}(x) - \deg r_j(x), \\ &\deg q_{j+1}^{(l)}(x^\ell) < \deg b_j^{(v)}(x) - \deg b^{(l)}(x) \text{ for } l \neq v_j. \end{aligned} \tag{3.34}$$

Due to our updating rule in Line 8 and Line 9 in Algorithm 8, at most the first ℓ iterations, the minuend in (3.34) can be one of $b_0^{(l)}(x)$. From the $(\ell+1)$st iteration on, all minuends are the previous remainders which can be canceled by previous subtrahends. Hence,

$$\begin{aligned} T &= \frac{cI}{\ell} \sum_{j=0}^{I} \max_l \left\{ \deg b_j^{(v_j)}(x) - \deg r_j(x), \deg b_j^{(v)}(x) - \deg b^{(l)}(x) \right\} \\ &\leq \frac{cI}{\ell} \sum_{l=1}^{\ell} \deg b_0^{(l)}(x) \leq \frac{cI}{\ell} \ell I = cI^2. \end{aligned}$$

Line 10 is used for calculating the coefficient polynomials of $r_0(x)$. However, this will not increase the burden to the overall complexity as the time of Line 10 is less than T_j. Therefore, the complete GEEA algorithm still has complexity $\mathcal{O}(I^2)$ in $\mathbb{F}$. □

Given ℓ sequences $S^{(l)}(x) \in \mathbb{F}[x]$ of length $N^{(l)}$. If $r_0(x), b_0^{(l)}(x)$ are given according to Line 3 in Algorithm 8 and the maximum length of the sequences is N, then

$$I \geq \ell N + l - 1.$$

Therefore, the overall complexity of *complete* GEEA is $\mathcal{O}(l^2N^2)$.

Since we use GEEA to decode (heterogeneous) IRS codes, the algorithm stops when $\deg r_j(x) < \deg U_j(x^\ell)$ for the first time, which means the iteration times $\widetilde{I} = I - \deg U_k(x^\ell) = I - \ell t$ where t is the number of errors. Assume every sequence has length N, by combining (3.19) with $\widetilde{I}$, we obtain

$$\begin{aligned}\widetilde{I} &= \ell N + l - 1 - \ell\frac{\ell}{\ell+1}N \\ &< \frac{\ell}{\ell+1}N + \ell.\end{aligned}$$

which has order $\mathcal{O}(N)$. However, in each iteration, we still need I operations. Therefore, we have the following theorem.

Corollary 9 (Complexity of GEEA solving MS-LFSR synthesis). *Given multiple sequences $S^{(l)}(x)$ of length $N^{(l)}$ for $l = 1, \ldots, \ell$ over $\mathbb{F}[x]$. Denote the maximum length of the sequence by N. Solving Problem 1, 2 or 3 with Algorithm 8 has time complexity $\mathcal{O}(\ell N^2)$ operations in $\mathbb{F}$.*

From Theorem 4 and Theorem 9, BMA-based Algorithm 4 and EA-based Algorithm 8 have the same order of time complexity.

3.5 Fast Decoding of IRS Codes

First let us revisit the strategy to accelerate the Euclidean algorithm (EA) for only two polynomials over $\mathbb{F}[x]$. Recall Algorithm 2, if we replace the degree comparison by $r_i(x) \neq 0$ for the `while` condition, then Algorithm 2 can find GCD of $r_0(x)$ and $r_1(x)$.

3.5.1 Extended Euclidean Algorithm

In fact, Algorithm 2 is an extended Euclidean algorithm (EEA). Given two polynomials $r_0(x), r_1(x) \in \mathbb{F}[x]$, for every iteration step, it finds not only the remainder $r_i(x)$, but also the coefficient polynomials $f_i(x)$ and $g_i(x)$ in each iteration such that

$$f_i(x)r_0(x) + g_i(x)r_1(x) = r_i(x) \text{ for } i = 2, 3, \ldots.$$

Denote the $\left(\begin{smallmatrix} 0 & 1 \\ 1 & -q_i(x) \end{smallmatrix}\right)$ in Line 7 by Q_i. Then

$$\begin{pmatrix} r_i(x) \\ r_{i+1}(x) \end{pmatrix} = Q_i \begin{pmatrix} r_{i-1}(x) \\ r_i(x) \end{pmatrix} = Q_iQ_{i-1}\cdots Q_1 \begin{pmatrix} r_0(x) \\ r_1(x). \end{pmatrix} = R_i \begin{pmatrix} r_0(x) \\ r_1(x). \end{pmatrix}$$

where

$$R_i = Q_i \cdots Q_1 = \begin{pmatrix} f_i(x) & g_i(x) \\ f_{i+1}(x) & g_{i+1}(x) \end{pmatrix} \text{ for } i = 1, \ldots k-1 \tag{3.35}$$

is called *Euclidean matrix*.

The main complexity issue concerning the Euclidean algorithm is the time to perform each division and the number of iterations that are required to compute the GCD of two polynomials $r_0(x)$ and $r_1(x)$.

Theorem 10 (Complexity of EA [Lip81] Theorem 3, VII.3.1). *Given two polynomials $r_0(x)$, $r_1(x) \in \mathbb{F}[x]$ and let $N \geq \deg r_0(x) \geq \deg r_1(x)$ be an integer, the cost of carrying out a GCD of $r_0(x)$ and $r_1(x)$ by the Euclidean algorithm is in quadratic time, i.e., $\mathcal{O}(N^2)$, assuming the coefficients operations take constant $\mathcal{O}(1)$ time.*

It is easy to see that, each iteration of EEA needs to handle the multiplication of $f_i(x)$ and $q_i(x)$, $g_i(x)$ and $q_i(x)$ in addition to the division. However, this does not affect the complexity of Theorem 10.

3.5.2 Fast Extended Euclidean Algorithm

The so-called *divide and conquer* is a strategy to accelerate solving a problem. It recursively divides the problem into smaller parts, compute the solution of each parts, and combine the solutions for the whole problem. This strategy can be applied to speed up the EEA since the first quotient q_i only depends on the highest coefficients of $r_i(x)$ and $r_{i-1}(x)$ in each iteration. The fast EA has been described by Aho, Hopcroft and Ulman [AHU74]. Blahut [Bla85] used the same idea to accelerate the EEA.

Theorem 11 ([Bla85] Theorem 10.7.1). *Given two polynomials $r_0(x)$, $r_1(x) \in \mathbb{F}[x]$ with $\deg r_0(x) \geq \deg r_1(x)$, let*

$$\begin{aligned} r_0(x) &= \tilde{r}_0(x)x^{\kappa} + \hat{r}_0(x), \\ r_1(x) &= \tilde{r}_1(x)x^{\kappa} + \hat{r}_1(x) \end{aligned}$$

where $\deg \hat{r}_0(x) < \kappa$ and $\deg \hat{r}_1(x) < \kappa$ for some κ satisfying

$$\kappa \leq 2\deg r_1(x) - \deg r_0(x), \tag{3.36}$$

i.e., $\deg \tilde{r}_1(x) \geq \deg \tilde{r}_0(x)$. Let $r_0(x) = q(x)r_1(x) + r_2(x)$ and $\tilde{r}_0(x) = \tilde{q}(x)\tilde{r}_1(x) + \tilde{r}_2(x)$ each satisfy the division algorithm, then

$$\begin{aligned} q(x) &= \tilde{q}(x), \\ r_2(x) &= \tilde{r}_2(x)x^{\kappa} + \hat{r}_2(x) \end{aligned}$$

where $\deg \hat{r}_2(x) < \deg r_0(x) - \deg r_1(x) + \kappa$. In other words, $r_2(x)$ and $\tilde{r}_2(x)x^{\kappa}$ agree in all terms of degree $\deg r_0(x) - \deg r_1(x) + \kappa$ or higher.

If the division algorithm are proceeded recursively in EA or EEA, then the following theorem states how the polynomial should be truncated such that the correctness of the quotients is not influenced.

Algorithm 9: Fast (complete) extended Euclidean algorithm `FEEA`

1 **Input**: $r_0(x), r_1(x) \in \mathbb{F}[x]$, with $\deg r_0(x) \geq \deg r_1(x)$
2 **begin**
3 $\quad \kappa = \lfloor \deg r_0(x)/2 \rfloor$
4 $\quad$ **if** $\deg r_1(x) \leq \kappa$ **then**
5 $\quad\quad q(x) \leftarrow \texttt{Quotient}(r_0(x), r_1(x))$
6 $\quad\quad R \leftarrow \begin{pmatrix} 0 & 1 \\ 1 & -q(x) \end{pmatrix}$
7 $\quad$ **else**
8 $\quad\quad R \leftarrow \texttt{HEEA}(r_0(x), r_1(x))$
9 $\quad \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix} \leftarrow R \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix}$
10 $\quad$ **if** $r_1(x) \neq 0$ **then**
11 $\quad\quad R' \leftarrow \texttt{FEEA}(r_0(x), r_1(x))$
12 $\quad\quad R \leftarrow R'R$
13 **Output**: R

Theorem 12 ([Bla85] Theorem 10.7.3). *Let R_i and $\widetilde{R}_i$ be the Euclidean matrices of $r_0(x), r_1(x)$ and truncated polynomials $\tilde{r}_0(x), \tilde{r}_1(x)$ as in Theorem 11 in each i, respectively. Then*

$$R_i = \widetilde{R}_i$$

provided that $\deg \tilde{r}_{i+1}(x) \geq (\deg r_0(x) - \kappa)/2$ *where $\tilde{r}_{i+1}(x)$ is the remainder of division for $\tilde{r}_{i-1}(x)$ and $\tilde{r}_i(x)$.*

From Theorem 11, the smallest value that κ can get is 0. In this case, we see from (3.36) that

$$\deg r_1(x) \geq \frac{\deg r_0(x)}{2} \tag{3.37}$$

should be satisfied if the fast algorithm is applied.

Towards formulating the the fast EEA, the dividend and divisor input polynomials are truncated recursively till (3.37) is not fulfilled. The quotients are calculated by polynomials of smallest degree in this recursive truncation. Afterwards, the algorithm moves one step further in the EA by calculating the next remainder. This remainder, together with the previous divisor are then used as input in the next recursive call.

The complete fast extended Euclidean algorithm consists of two algorithms, the fast EEA (`FEEA`), Algorithm 9, as the main one and the half EEA (`HEEA`), Algorithm 10, as a calling function [Bla85].

In Line 4 of Algorithm 9, it decides whether to perform one normal iteration of the EA or to truncate the polynomials. In the normal iteration, The `Quotient` function computes

Algorithm 10: Half extended Euclidean algorithm `HEEA`

1 **Input**: $r_0(x), r_1(x) \in \mathbb{F}[x]$, with $\deg r_0(x) \geq \deg r_1(x)$
2 **begin**
3 $\quad \kappa = \lfloor \deg r_0(x)/2 \rfloor$
4 $\quad$ **if** $\deg r_1(x) \leq \kappa$ **then**
5 $\quad\quad R \leftarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
6 $\quad$ **else**
7 $\quad\quad \tilde{r}_0(x) \leftarrow \texttt{Quotient}(r_0(x), x^{\kappa})$
8 $\quad\quad \tilde{r}_1(x) \leftarrow \texttt{Quotient}(r_1(x), x^{\kappa})$
9 $\quad\quad R \leftarrow \texttt{HEEA}(\tilde{r}_0(x), \tilde{r}_1(x))$
10 $\quad\quad \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix} \leftarrow R \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix}$
11 $\quad\quad q(x) \leftarrow \texttt{Quotient}(r_0(x), r_1(x))$
12 $\quad\quad R \leftarrow \begin{pmatrix} 0 & 1 \\ 1 & -q(x) \end{pmatrix} R$
13 $\quad\quad \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix} \leftarrow \begin{pmatrix} 0 & 1 \\ 1 & -q(x) \end{pmatrix} \begin{pmatrix} r_0(x) \\ r_1(x) \end{pmatrix}$
14 $\quad\quad \kappa' \leftarrow \lfloor \kappa/2 \rfloor$
15 $\quad\quad \tilde{r}_0(x) \leftarrow \texttt{Quotient}(r_0(x), x^{\kappa'})$
16 $\quad\quad \tilde{r}_1(x) \leftarrow \texttt{Quotient}(r_1(x), x^{\kappa'})$
17 $\quad\quad R' \leftarrow \texttt{HEEA}(\tilde{r}_0(x), \tilde{r}_1(x))$
18 $\quad\quad R \leftarrow R'R$
19 **Output**: R

the quotient of $r_0(x)/r_1(x)$ which is a usual division. The degree of the new $r_0(x)$ and $r_1(x)$ in Line 9 are not larger than κ. Since these two new polynomials are the input for the next call of `FEEA` in Line 11, one normal iteration halves the degree the input polynomials.

Divide and conquer strategy is applied in Algorithm 10. The algorithm contains two recursive calls with truncated polynomials in Line 9 and Line 17. In Line 10 and Line 13, the new $r_1(x)$ are successive remainders, of degree at most 3/4 the degree of $r_0(x)$. The input polynomials in Line 17 are of degree at most 1/4 the degree of $r_0(x)$.

Theorem 13 (Complexity of fast EEA [AHU74] Theorem 8.19). *Given two polynomials $r_0(x)$, $r_1(x) \in \mathbb{F}[x]$ and let $N \geq \deg r_0(x) \geq \deg r_1(x)$ be an integer, the Algorithm 9 finds a GCD with time complexity $\mathcal{O}(M(N) \log N)$, where $M(N)$ is the time needed for multiplying two polynomials of degree N.*

Since we recursively divide the problem into nearly equal parts, we assume that N is actually a power of 2. In our case, if $\deg r_0(x)$ is not a power of 2, we select N as the smallest power of 2 which is equal to or larger than $\deg r_0(x)$. The complexity $M(N)$ is usually considered as $\mathcal{O}(N \log N)$ [Lip81, Theorem 1, IX.2.1]. Thus, the time complexity of Algorithm 9 is given by $\mathcal{O}(N \log^2 N)$.

3.5.3 Fast GEEA

The divide and conquer strategy for two polynomials can be generalized for multiple polynomials. This fast algorithm has been explored in [ZW11]. However, their algorithm is proposed to solve MS-LFSR problem for sequences of the same length. In this subsection, we propose a version of fast GEEA for sequences of different lengths, and also give the complexity analysis.

Let us shortly revisit Feng–Tzeng's GEEA. Given $\ell + 1$ polynomials $r_0(x), b_0^{(l)}(x)$ for $l = 1, \ldots, \ell$, where $r_0(x) \in [x^{v_0}]$ for some $1 \leq v_0 \leq \ell$ and $b_0^{(l)}(x) \in [x^{l-1}]$ with $\deg b_0^{(v_0)}(x) > r_0(x)$. By repeatedly running `GenDiv`, (3.32) can be represented as

$$\begin{pmatrix} r_{j+1}(x) \\ b_{j+1}^{(1)}(x) \\ \vdots \\ b_{j+1}^{(v_j)}(x) \\ \vdots \\ b_{j+1}^{(\ell)}(x) \end{pmatrix} = Q_{j+1} \begin{pmatrix} r_j(x) \\ b_j^{(1)}(x) \\ \vdots \\ b_j^{(v_j)}(x) \\ \vdots \\ b_j^{(\ell)}(x) \end{pmatrix} \tag{3.38}$$

where the matrix $Q_{j+1} \in \mathbb{F}^{(m+1)\times(m+1)}[x]$, $j = 0, 1, \ldots$ is given by

$$\begin{pmatrix} p_{j+1}(x^\ell) & q_{j+1}^{(1)}(x^\ell) & \ldots & q_{j+1}^{(v_j-1)}(x^\ell) & q_{j+1}^{(v_j)}(x^\ell) & q_{j+1}^{(v_j+1)}(x^\ell) & \ldots & q_{j+1}^{(\ell)}(x^\ell) \\ 0 & 1 & & 0 & 0 & 0 & \ldots & 0 \\ \vdots & & \ddots & & \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 & 0 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ 0 & 0 & \ldots & 0 & 0 & 0 & & 1 \end{pmatrix}. \tag{3.39}$$

Similarly, let $U_0(x) = 1$, and $V_0^{(l)}(x) = 0$ for $l = 1, \ldots, \ell$, define polynomials $U_j(x), V_j^{(l)}(x)$

for $j \geq 0$ such that

$$\begin{pmatrix} U_{j+1}(x^\ell) \\ V_{j+1}^{(1)}(x^\ell) \\ \vdots \\ V_{j+1}^{(v_j)}(x^\ell) \\ \vdots \\ V_j^{(\ell)}(x^\ell) \end{pmatrix} = Q_{j+1} \begin{pmatrix} U_j(x^\ell) \\ V_j^{(1)}(x^\ell) \\ \vdots \\ V_j^{(v_j)}(x^\ell) \\ \vdots \\ V_j^{(\ell)}(x^\ell) \end{pmatrix}. \tag{3.40}$$

Let

$$R_j = Q_j Q_{j-1} \dots Q_1, \ j = 1, 2, \dots, \tag{3.41}$$

from (3.38) and (3.39) we obtain

$$\begin{pmatrix} r_j(x) \\ b_j^{(1)}(x) \\ \vdots \\ b_j^{(v_{j-1})}(x) \\ \vdots \\ b_j^{(\ell)}(x) \end{pmatrix} = R_j \begin{pmatrix} r_0(x) \\ b_0^{(1)}(x) \\ \vdots \\ b_0^{(v_0)}(x) \\ \vdots \\ b_0^{(\ell)}(x) \end{pmatrix}. \tag{3.42}$$

and from (3.39)

$$\begin{pmatrix} U_j(x^\ell) \\ V_j^{(1)}(x^\ell) \\ \vdots \\ V_j^{(v_{j-1})}(x^\ell) \\ \vdots \\ V_j^{(\ell)}(x^\ell) \end{pmatrix} = R_j \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{3.43}$$

Analogously to Theorem 11, the quotients of the original and the truncated polynomials for $r_j(x)$ and $b^{(l)}(x)$ are also the same.

Theorem 14. *Given $\ell + 1$ polynomials $r(x), \{b^{(l)}(x)\}_{l=1}^{\ell} \in \mathbb{F}[x]$ with $b^{(l)}(x) \in [x^{l-1}]$. For some $v \in [1, \ell]$, $r(x) \sim b^{(v)}(x)$ and $\deg r(x) < \deg b^{(v)}(x)$. Let*

$$\begin{aligned} r(x) &= \tilde{r}(x)x^\kappa + \hat{r}(x), \\ b^{(l)}(x) &= \tilde{b}^{(l)}(x)x^\kappa + \hat{b}^{(l)}(x) \end{aligned} \tag{3.44}$$

where $\{\deg \tilde{b}^{(l)}(x)\}_{l=1,l\neq v}^{\ell} > 0$, and $\deg \hat{r}(x) < \kappa$, $\{\deg \hat{b}^{(l)}(x)\}_{l=1}^{\ell} < \kappa$ for some κ satisfying

$$\kappa \leq 2 \deg r(x) - \deg b^{(v)}(x).$$

Let $r'(x) = p(x^\ell)r(x) + \sum_{l=1}^{\ell} q(x^\ell)b^{(l)}(x)$ *and* $\tilde{r}'(x) = \tilde{p}(x^\ell)\tilde{r}(x) + \sum_{l=1}^{\ell} \tilde{q}(x^\ell)\tilde{b}^{(l)}(x)$ *satisfy the generalized division algorithm* **GenDiv** *separately, then*

$$\begin{aligned} p(x^\ell) &= \tilde{p}(x^\ell), \\ q(x^\ell) &= \tilde{q}(x^\ell), \\ r'(x) &= \tilde{r}'(x)x^\kappa + \hat{r}'(x) \end{aligned}$$

where $\deg \hat{r}'(x) < \deg b^{(v)}(x) - \deg r(x) + \kappa$.

Proof: For $l \neq v$, since $\deg \tilde{b}^{(l)}(x) > 0$ and $\deg \hat{b}^{(l)}(x) < \kappa$, it follows from (3.44) that $\deg b^{(l)}(x) > \kappa$. In fact, all degrees of $b^{(l)}(x)$ are greater than κ, since $\kappa \leq 2\deg r(x) - \deg b^{(v)}(x) < \deg r(x) < \deg b^{(v)}(x)$.

Now let us recall `GenDiv`, i.e., Algorithm 7. The algorithm starts from $r(x)$ and $b^{(v)}(x)$ by calculating the quotient $q^{(v)}(x^\ell)$ and remainder $r_1(x)$ by using modified division algorithm `ModDiv` such that $b^{(v)}(x) = q^{(v)}(x^\ell)r(x) + r_1(x)$. Same procedure are also applied for truncated polynomials $\tilde{r}(x)$ and $\tilde{b}^{(v)}(x)$, i.e., $\tilde{b}^{(v)}(x) = \tilde{q}^{(v)}(x^\ell)\tilde{r}(x) + \tilde{r}_1(x)$. Since the modified division algorithm doesn't stop later than usual division algorithm and according to Theorem 11, we have $q^{(v)}(x^\ell) = \tilde{q}^{(v)}(x^\ell)$ and $r_1'(x) = \tilde{r}_1'(x)x^\kappa + \hat{r}_1'(x)$ where $\deg \hat{r}_1'(x) < \deg b^{(v)}(x) - \deg r(x) + \kappa$.

There are two cases in which the `GenDiv` algorithm stops after running once `ModDiv`. One is when $r_1(x) \sim r(x)$, and the other one is when $r_1(x) \sim b^{(u)}(x)$ for some $u \neq v$ and $\deg r_1(x) < \deg b^{(u)}(x)$. Clearly, if `GenDiv` stops after first time `ModDiv`, then the statement of the theorem follows.

If `GenDiv` continues running, then it must be when $r_1(x) \sim b^{(u)}(x)$ for some $u \neq v$ and $\deg r_1(x) \geq \deg b^{(u)}(x)$. By using `ModDiv` again, if we need to have $q^{(u)}(x^\ell) = \tilde{q}^{(u)}(x^\ell)$ with the same κ truncation, then $\kappa \leq 2\deg b^{(u)}(x) - \deg r_1(x) \leq \deg b^{(u)}(x)$. For the same reason, the modified division in the following steps will generate the same quotients for two original and truncated polynomials as long as κ is smaller than the degree of $b^{(l)}(x)$, for $l \neq v$ or degree of $r(x)$. Regarding to the remainders $r'(x)$ and $\tilde{r}'(x)$, since their degrees are smaller than those of $r_1(x)$ and $\tilde{r}_1'(x)$, we have $\deg \hat{r}'(x) < \deg \hat{r}_1'(x) < \deg b^{(v)}(x) - \deg r(x) + \kappa$. □

If `GenDiv` is proceeded repeatedly, then we can apply the following theorem which is a generalization of Theorem 12.

Theorem 15. *Let all prerequisites be the same as in Theorem 14. Let the polynomials be initialized with* $r_0(x) = r(x)$, $b_0^{(l)}(x) = b^{(l)}(x)$ *and with* $\tilde{r}_0(x) = \tilde{r}_0$, $\tilde{b}_0^{(l)}(x) = \tilde{b}^{(l)}(x)$. *Let* R_j *and* $\widetilde{R}_j$ *be the Euclidean matrices defined by* (3.41) *computed from* $r_0(x), b_0^{(l)}(x)$ *and the truncated polynomials* $\tilde{r}_0(x), \tilde{b}_0^{(l)}(x)$, *respectively. Let* $v_{j-1} = (\deg r_j(x) \mod \ell) + 1$. *Then for each* j,

$$R_j = \widetilde{R}_j$$

provided that $\deg \tilde{r}_j(x) \geq (\deg b^{(v)} - \kappa)/2$ *and* $\{\deg \tilde{b}_j^{(l)}(x)\}_{l=1, l\neq v_{j-1}}^{\ell} \geq (\deg b^{(v)} - \kappa)/2$.

Proof: Theorem 14 assures that the quotients $p_j(x^\ell), q_j^{(l)}(x^\ell)$ in the matrix Q_j in (3.39) agree in each iteration, i.e., after proper truncation, the quotients will not change. Theorem 14 also assures that in the corresponding sequences $r_j(x), b_j^{(1)}(x), \ldots, b_j^{(l)}(x)$ a sufficient number of the high-order terms agree. □

Equipped with Theorem 15, the fast version of our modified EEA (Algorithm 9) for MS-LFSR synthesis is illustrated in Algorithm 11. The algorithm together with its calling function `HGEEA` are an alternative version of those in [ZW11] with slight modification.

Algorithm 11: Fast generalized extended Euclidean algorithm for MS-LFSR synthesis `FGEEA`

Input: ℓ; $r(x), \{b^{(l)}(x)\}_{l=1}^{\ell} \in \mathbb{F}[x]$, $b^{(l)}(x) \in [x^{l-1}]$.

begin

- $U(x) \leftarrow 1$, $\{V^{(l)}(x)\}_{l=1}^{\ell} \leftarrow 0$
- Find $v \in [1, \ell]$ such that $r(x) \sim b^{(v)}(x)$ and $\deg r(x) < \deg b^{(v)}(x)$
- $\kappa \leftarrow \lfloor \deg b^{(v)}(x)/2 \rfloor$
- **if** $\deg r(x) \le \kappa$ *or* $\{\deg b^{(l)}(x)\}_{l=1, l\neq v}^{\ell} \le \kappa$ **then**
  - $r(x), p(x^\ell), q(x^\ell) \leftarrow \texttt{GenDiv}(r(x), \{b^{(l)}(x)\}_{l=1}^{\ell})$
  - Construct matrix R according to (3.39) with v
- **else**
  - $R \leftarrow \texttt{HGEEA}(r(x), \{b^{(l)}(x)\}_{l=1}^{\ell})$
- $$\begin{pmatrix} r(x) & U(x^\ell) \\ b^{(1)}(x) & V^{(1)}(x^\ell) \\ \vdots & \vdots \\ b^{(\ell)}(x) & V^{(\ell)}(x^\ell) \end{pmatrix} \leftarrow R \begin{pmatrix} r(x) & U(x^\ell) \\ b^{(1)}(x) & V^{(1)}(x^\ell) \\ \vdots & \vdots \\ b^{(\ell)}(x) & V^{(\ell)}(x^\ell) \end{pmatrix}$$
- **if** $\deg U(x^\ell) \le \deg r(x)$ **then**
  - $R' \leftarrow \texttt{FGEEA}(r(x), \{b^{(l)}(x)\}_{l=1}^{\ell})$
  - $R \leftarrow R'R$

Output: R, $U(x)$

Given ℓ sequences $S^{(l)}(x) \in \mathbb{F}[x]$ of length $N^{(l)}$, the initial assignment of the inputs in Algorithm 11 is given by

$$r(x) = r_0(x) = \sum_{l=1}^{\ell} S^{(l)}(x^\ell) x^{l-1}, \tag{3.45}$$

and

$$b^{(l)}(x) = b_0^{(l)}(x) = x^{N^{(l)}\ell + l - 1}, \text{ for } l = 1, \ldots, \ell. \tag{3.46}$$

Denote the length of longest sequence of $S^{(l)}$ by $N_{\max}$, the degree of $r_0(x)$ in (3.45) depends only on $N_{\max}$, since $\deg r_0(x) = \max_l\{(N_l - 1)\ell + l - 1\}$ and $l - 1 < \ell$. For $r_0(x)$ with degree

$(N_{\max}-1)\ell+l-1$, it is easy to see that, $r_0(x) \sim x^{N_{\max}\ell+l-1}$ and $\deg r_0(x) < N_{\max}\ell+l-1$ where $N_{\max}\ell+l-1$ is the highest degree of $b_0^{(l)}(x)$ in (3.46). Therefore, with (3.45) and (3.46), one can always find some v, such that $r_0(x) \sim b_0^{(v)}(x)$ and $\deg r_0(x) < \deg b_0^{(v)}(x)$. Since Algorithm 11 is recursive, Line 4 appears in each recursion. The property of generalized Euclidean division guarantees that such v always exists.

According to Theorem 15, the truncation is proceeded only when degrees of all polynomials are larger than half of that of $b^{(v)}(x)$. Otherwise, generalized Euclidean division is carried. If $r(x)$ has a degree which is less than or equal to κ, or other input polynomials $b^{(l)}(x)$, other than $r(x)$ and $b^{(v)}(x)$ have degrees that are less than or equal to κ, then one normal iteration of the GEEA cuts the size of the problem in half. Therefore, nothing is lost if only half of the coefficients of $r(x)$ or $b^{(l)}(x)$ is used in the calculation.

Nothing has changed in Line 11, in comparison with the updating part in the modified GEEA. If R is obtained from normal generalized division, then the matrix multiplication can be reduced to only using the first row of R to multiply with the matrix containing $r(x), b^{(l)}(x), U(x^\ell)$ and $V^{(l)}(x^\ell)$. If R is obtained by truncated polynomials in Line 10, then it is an $(\ell+1)\times(\ell+1)$ matrix multiplication.

Algorithm 12 describes the recursive procedure of fast half GEEA. There are two times of self call (Line 11 and Line 22). Each one directly follows the half truncation. Polynomials $\tilde{r}(x), \tilde{b}^{(l)}(x)$ are the truncated polynomials which are the quotient of $r(x)/x^\kappa$ by usual division. In Line 12 and Line 14, the two new $r(x)$ are successive remainders, of degree at most 3/4 the degree of $r(x)$ which is the input of Algorithm 12. The input polynomials in Line 22 are of degree at most 1/4 the degree of the input $r(x)$. The algorithm does not stop until one of the input polynomials, other than $b^{(v)}(x)$, has a degree less than κ, i.e., half of that of $b^{(v)}(x)$.

Now we illustrate fast algorithm `FGEEA` with an example.

Example 2. *Consider an $\mathcal{IRS}(10,[4,5])$ code. The syndromes of the received word are*

$$S^{(1)}(x) = 3x^5+9x^4+4x^3+4x^2+x+5$$
$$S^{(2)}(x) = 0x^4++3x^3+x^2+4x+1$$

of length $N^{(1)}=6$ and $N^{(2)}=5$, respectively.

Algorithm 12: Half generalized extended Euclidean algorithm `HGEEA`

Input: ℓ; $r(x), \{b^{(l)}(x)\}_{l=1}^{\ell} \in \mathbb{F}[x]$, $b^{(l)}(x) \in [x^{l-1}]$. $\exists v \in [1, \ell]$ such that $r(x) \sim b^{(v)}(x)$ and $\deg r(x) < \deg b^{(v)}(x)$.

begin

  Find $v \in [1, \ell]$ such that $r(x) \sim b^{(v)}(x)$ and $\deg r(x) < \deg b^{(v)}(x)$

  $\kappa \leftarrow \lfloor \deg b^{(v)}(x)/2 \rfloor$

  if $\deg r(x) \leq \kappa$ *or* $\{\deg b^{(l)}(x)\}_{l=1, l\neq v}^{\ell} \leq \kappa$ **then**

    $R \leftarrow (\ell+1) \times (\ell+1)$ identity matrix

  else

    $\tilde{r}(x) \leftarrow \texttt{Quotient}(r(x), x^{\kappa})$

    for $l = 1$ *to* ℓ **do**

      $\tilde{b}^{(l)}(x) \leftarrow \texttt{Quotient}(b^{(l)}(x), x^{\kappa})$

    $R \leftarrow \texttt{HGEEA}(\tilde{r}(x), \{\tilde{b}^{(l)}(x)\}_{l=1}^{\ell})$

    $$\begin{pmatrix} r(x) \\ b^{(1)}(x) \\ \vdots \\ b^{(\ell)}(x) \end{pmatrix} \leftarrow R \begin{pmatrix} r(x) \\ b^{(1)}(x) \\ \vdots \\ b^{(\ell)}(x) \end{pmatrix}$$

    Find $v' \in [1, \ell]$ such that $r(x) \sim b^{(v')}(x)$ and $\deg r(x) < \deg b^{(v')}(x)$

    $r(x), p(x^{\ell}), q(x^{\ell}) \leftarrow \texttt{GenDiv}(r(x), \{b^{(l)}(x)\}_{l=1}^{\ell})$

    Construct matrix R' according to (3.39) with v'

    $R \leftarrow R'R$

    $$\begin{pmatrix} r(x) \\ b^{(1)}(x) \\ \vdots \\ b^{(\ell)}(x) \end{pmatrix} \leftarrow R' \begin{pmatrix} r(x) \\ b^{(1)}(x) \\ \vdots \\ b^{(\ell)}(x) \end{pmatrix}$$

    $\kappa' \leftarrow \lfloor \kappa/2 \rfloor$

    $\tilde{r}(x) \leftarrow \texttt{Quotient}(r(x), x^{\kappa'})$

    for $l = 1$ *to* ℓ **do**

      $\tilde{b}^{(l)}(x) \leftarrow \texttt{Quotient}(b^{(l)}(x), x^{\kappa'})$

    $R'' \leftarrow \texttt{HGEEA}(\tilde{r}(x), \{\tilde{b}^{(l)}(x)_{l=l}^{\ell}\})$

    $R \leftarrow R''R$

Output: R

According to (3.45) *and* (3.46),

$$r_0(x) = 3x^{10} + 9x^8 + 3x^7 + 4x^6 + x^5 + 4x^4 + 4x^3 + x^2 + x + 5, \tag{3.47}$$

$$b_0^{(1)}(x) = x^{12}, \quad b_0^{(2)}(x) = x^{11}. \tag{3.48}$$

Obviously, $r_0(x) \sim b^{(1)}(x)$. *If* $r(x) = r_0(x)$ *and* $b^{(l)}(x) = b_0^{(l)}(x)$, *then in Lines 8-10 in Algorithm 12 we have* $v = 1$ *and* $\kappa = 6$, *and*

$$\tilde{r}(x) = 3x^4 + 9x^2 + 3x + 4, \tag{3.49}$$

$$\tilde{b}^{(1)}(x) = x^6, \quad \tilde{b}^{(2)}(x) = x^5. \tag{3.50}$$

Then **HGEEA** *is called in Line 11.*

In this first recursion, $\kappa = 3$. *It is not difficult to see that, after* κ *is cut to half, Line 5 is true and* R *is a* 3×3 *diagonal matrix in Line 11. The index* $v' = 1$. *Then we use the polynomials in* (3.49) *and* (3.50) *for generalized division in Line 14 and obtain*

$$R' = \begin{pmatrix} 7x^2 + 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In Line 15. The polynomials are updated in Line 17 and with value

$$r(x) = 10x^3 + 4x^2 + 3x + 4,$$
$$b^{(1)}(x) = 3x^4 + 9x^2 + 3x + 4, \quad b^{(2)}(x) = x^5.$$

Since it is still in the first iteration, $\kappa' = 1$, *and in Line 19-21*

$$\tilde{r}(x) = 10x^2 + 4x + 3,$$
$$\tilde{b}^{(1)}(x) = 3x^3 + 9x + 3, \quad \tilde{b}^{(2)}(x) = x^4.$$

Since $\deg \tilde{r} \leq \deg \tilde{b}^{(2)}$, R'' *is returned as a diagonal matrix in Line 22. Line 23 together with Line 16 indicates the Euclidean matrix*

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 7x^2 + 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 7x^2 + 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.51}$$

for the first recursion.

We proceed on Line 12, with multiplication of R *in* (3.51) *and the* $r_0(x)$, $b_0^{(0)}(x)$, $b_0^{(1)}(x)$ *in* (3.47) *and* (3.48), *to give*

$$r(x) = 10x^9 + 4x^8 + 10x^7 + 10x^6 + 7x^5 + 3x^2 + x + 5,$$
$$b^{(1)}(x) = 3x^{10} + 9x^8 + 3x^7 + 4x^6 + x^5 + 4x^4 + 4x^3 + x^2 + x + 5,$$
$$b^{(2)}(x) = x^{11}$$

which lead to R' in Line 15 to be

$$R' = \begin{pmatrix} x^2+10 & 6 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}. \tag{3.52}$$

Line 17 yields

$$\begin{aligned} r(x) &= 5x^8 + 4x^7 + 3x^6 + 10x^5 + 5x^4 + 3x^3 + 8x^2 + 5x + 3, \\ b^{(1)}(x) &= 3x^{10} + 9x^8 + 3x^7 + 4x^6 + x^5 + 4x^4 + 4x^3 + x^2 + x + 5, \\ b^{(2)}(x) &= 10x^9 + 4x^8 + 10x^7 + 10x^6 + 7x^5 + 3x^2 + x + 5. \end{aligned}$$

With $\kappa' = 3$, the truncated polynomials are

$$\begin{aligned} \tilde{r}(x) &= 5x^5 + 4x^4 + 3x^3 + 10x^2 + 5x + 3, \\ \tilde{b}^{(1)}(x) &= 3x^7 + 9x^5 + 3x^4 + 4x^3 + x^2 + 4x + 4, \\ \tilde{b}^{(2)}(x) &= 10x^6 + 4x^5 + 10x^4 + 10x^3 + 7x^2. \end{aligned}$$

At Line 22, we find

$$R'' = \begin{pmatrix} 6x^2+4 & 1 & 2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.53}$$

So in Line 23 we multiply (3.51), (3.52) *and* (3.53), *and have the result*

$$\begin{aligned} R &= \begin{pmatrix} 6x^2+4 & 1 & 2 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x^2+10 & 6 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 7x^2+1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 9x^6+3x^4+9x^2+1 & 6x^4+9x^2+9 & 6x^2+4 \\ 7x^4+5x^2+5 & x^2+10 & 1 \\ 7x^2+1 & 1 & 0 \end{pmatrix} \end{aligned}$$

which is the returned value in Line 10 in the main algorithm, Algorithm 11. Now let us continue the main one and compute $r(x)$ and $U(x^\ell)$ in Line 11 and obtain

$$\begin{aligned} r(x) &= 7x^5 + 6x^4 + 2x^3 + 2x^2 + x + 5, \\ U(x^\ell) &= 9x^6 + 3x^4 + 9x^2 + 1. \end{aligned}$$

It can be seen that $\deg U(x^\ell) > \deg r(x)$ and thus the algorithm stops. Making $U(x)$ monic yields $x^3 + 4x^2 + x + 5 = (x+2)(x+3)(x+10)$ with roots at $\alpha^0, \alpha^3, \alpha^6$ which give the positions of errors.

3.5.4 Complexity of Fast GEEA for MS-LFSR Synthesis

Given $\ell+1$ input polynomials $r(x), b^{(l)}(x)$ as shown in the input of Algorithm 11. Let $N \geq \deg b^{(v)}(x)$ where $r(x) \sim b^{(v)}(x)$. Let $M(N)$ be the time to multiply two polynomials of degree at most N. Let us start by analyzing the operations of Algorithm 12.

The polynomial truncation in Lines 8-10 and 19-21 requires negligible $\mathcal{O}(N)$ number of operations. In Line 12, it involves multiplication of an $(\ell+1)\times(\ell+1)$ matrix R with an $(\ell+1)\times 1$ matrix containing polynomials of degree at most N. This operation requires $(\ell+1)^2$ times of multiplication of two polynomials of degree at most N and a fewer number of additions. Since the polynomials in R is *sparse*, i.e., they only have exponents at multiples of ℓ, the multiplication needs only $\mathcal{O}(M(N)/\ell)$ in time. Hence this step costs $c_1\ell M(N)$ number of operations, for some constant c_1.

Line 14 calls on `GenDiv`, to compute the quotients and remainder for each single iteration in the generalized Euclidean algorithm. As we have analyzed in (3.33), the time to perform `GenDiv` is $c_2 M(N)/\ell$ for some constant c_2.

Line 16 computes product of two $(\ell+1)\times(\ell+1)$ square matrices. The multiplication of the two square matrices contains at most $(\ell+1)^3$ polynomial multiplications and a slightly lower number of additions. However, each of the element in the matrix R is a sparse polynomial. As a result, the matrix multiplication needs $c_3\ell^2 M(N)$ number of operations, where c_3 is some constant.

The updating in Line 17 has the same time complexity as that in Line 12, i.e., $\mathcal{O}(\ell M(N))$.

Line 23 is again a matrix multiplication with sparse polynomials. Therefore, the resultant cost is $\mathcal{O}(\ell^2 M(N))$.

All other lines except Lines 11 and 22 contain linear operations $\mathcal{O}(N)$ which can be neglected.

Considering the complete `HGEEA`, Algorithm 12, the total cost $T_H(N)$ with input of degree at most N, is bounded by

$$T_H(N) \leq 2T_H(N/2) + c\ell^2 M(N), \tag{3.54}$$

for some constant c. The term $T_H(N/2)$ comes from Lines 11 and 22 when `HGEEA` calls itself recursively with half sized arguments. Substituting $N/2$ for N in (3.54) gives

$$T_H(N/2) \leq 2T_H(N/4) + c\ell^2 M(N/2).$$

Subsequently substituting the halved term in (3.54) yields

$$\begin{aligned}
T_H(N) &\leq 2T_H(N/2) + c\ell^2 M(N) \\
&\leq 4T_H(N/4) + 2c\ell^2 M(N/2) + c\ell^2 M(N) \\
&\leq 8T_H(N/8) + 4c\ell^2 M(N/4) + 2c\ell^2 M(N/2) + c\ell^2 M(N) \\
&\cdots \\
&\leq 2^{\log_2 N} T_H(N/2^{\log_2 N}) + c\ell^2 \sum_{i=0}^{(\log_2 N)-1} 2^i M(N/2^i) \\
&= NT_H(1) + c\ell^2 M(N)\log N \\
&= \mathcal{O}(\ell^2 M(N)\log N).
\end{aligned}$$

Theorem 16 (Complexity of HGEEA). *Algorithm 12 requires $\mathcal{O}(\ell^2 M(N)\log N)$ operations in $\mathbb{F}$ if its arguments are of degree at most N, where $M(N)$ is the time required to multiply two polynomials of degree N.*

Now we proceed to analyze the complexity of the fast generalized extended Euclidean algorithm — FGEEA. Based on the complexity analysis of HGEEA, in Algorithm 11, the most expensive steps are a call to HGEEA in Line 10 and a self call in Line 13. Hence, the total cost $T_F(N)$ of FGEEA is bounded by

$$T_F(N) = T_F(N/2) + c\ell^2 M(N)\log N \tag{3.55}$$

for some constant c. We apply same idea as that in (3.54) to (3.55), to calculate $T_F(N)$. Therefore,

$$\begin{aligned} T_F(N) &\leq T_F(N/2^{\log_2 N}) + c\ell^2 \sum_{i=0}^{(\log_2 N)-1} M(N/2^i)\log(N/2^i) \\ &\leq T_F(1) + c\ell^2 \sum_{i=0}^{(\log_2 N)-1} M(N/2^i)\log(N/2^i) \\ &\leq T_F(1) + c\ell^2 M(N)\log N + c\ell^2 \sum_{i=1}^{(\log_2 N)-1} M(N/2^i)\log(N/2^i). \end{aligned} \tag{3.56}$$

The last term in (3.56) converges to $c\ell^2 M(N)\log N$, hence

$$T_F(N) = \mathcal{O}(\ell^2 M(N)\log N).$$

Theorem 17 (Complexity of FGEEA). *Algorithm 11 requires $\mathcal{O}(\ell^2 M(N)\log N)$ time if its arguments are of degree at most N, where $M(N)$ is the time required to multiply two polynomials of degree N.*

Multiplication of two polynomials of degree N can be carried out in time $\mathcal{O}(N\log N)$ by fast Fourier transform. Regarding to the MS-LFSR synthesis, $N \geq \ell N_{\max} + \ell$ where $N_{\max}$ is the maximum length of the syndromes.

Corollary 18 (Complexity for fast GEEA solving MS-LFSR synthesis). *Given ℓ sequences $S^{(l)}(x)$ of length $N^{(l)}$ for $l = 1, \ldots, \ell$ over $\mathbb{F}[x]$. Denote the maximum length of the sequence by $N_{\max}$, Solving Problem 1, 2 or 3 with Algorithm 11 has time complexity $\mathcal{O}(\ell^3 N_{\max}\log^2(\ell N_{\max}))$.*

3.6 Discussion

To solve the multi-sequence shift register synthesis problem for the IRS codes, we proposed a modified algorithm based on the Feng–Tzeng's generalized extended Euclidean algorithm such that the input sequences for the shift register can have different lengths. Our algorithm requires quadratic numbers of field operations in the maximum length of sequences. We also accelerated our algorithm with the "divide and conquer" strategy, which leads to sub-quadratic number of field operations in the maximum sequence length.

4

Decoding Hermitian Codes

HERMITIAN codes are one of the most studied families of algebraic geometry codes [Gop77]. The Hamming distance d of an (N, K) Hermitian code over the field $\mathbb{F}_Q = \mathbb{F}_{q^2}$ is not far from the Singleton bound $d \leq N - K + 1$, but the code length N can be much longer than the order Q of the field, in contrast to very popular (RS) codes. This fact makes Hermitian codes potentially very interesting for applications and a number of efficient unique decoding algorithms were suggested, correcting up to $d/2$ independent errors [FR93, SJM$^+$95] and list decoders, correcting more than $d/2$ errors [GS99].

In this chapter, we consider Hermitian codes having rate $R = K/N \geq 1/2q$. For these codes we propose an efficient algorithm, correcting phased bursts of errors of length q, later called simply bursts. This means that we decode Hermitian codes in the burst metric, which is equivalent to decoding q-folded Hermitian codes in Hamming metric. Our algorithm is based on the fact, that decoding a Hermitian code can be reduced to decoding interleaved extended Reed–Solomon (IERS) codes [YB92, Ren04].

Denote by d_B the code distance of a Hermitian code in the burst metric. We consider unique decoding algorithms correcting more than $d_B/2$ bursts. These algorithms may fail in some cases. By $P_f(t)$ we denote the failure probability of a given decoding algorithm in presence of t bursts in the channel. The function $P_f(t)$ describes the performance of the decoding algorithm. Usually $P_f(t) = 0$ if $t < d_B/2$, then $P_f(t) \ll 1$ for $d_B/2 \leq t \leq t_{\max}$ for some integer $t_{\max}$, which is called the decoding radius, and then $P_f(t) \approx 1$ for $t > t_{\max}$. This definition of $t_{\max}$ is not precise, and it depends on what $P_f(t_{\max}) \ll 1$ means. For the proposed algorithm we will give the function $P_f(t)$. Performance of a decoding algorithm can be approximately described by $t_{\max}$ and $P_f(t_{\max})$.

Known results. It was shown by Yaghoobian and Blake [YB92] and by Ren [Ren04] that every Hermitian code can be represented as concatenation of an IERS outer code and trivial (q, q) inner code. Hence decoding of a Hermitian code can be reduced to decoding IERS codes [Ren04]. To decode an IERS code, Ren [Ren04] suggested to decode RS codes individually, where every next decoder erases positions, corrected by previous RS decoders. It was shown by examples, that this decoder can correct more than $d/2$ errors, if the errors occur in bursts. However, burst error correcting radius and decoding-failure probability were not obtained. The time complexity of the algorithm is $\mathcal{O}(N^{5/3})$ operations in $\mathbb{F}_Q$.

It is known that, for decoding interleaved RS (IRS) codes, *joint* decoding [Kra03, SSB09] is more effective than Ren's [Ren04]. Özbudak and Yayla [OY14] suggested an algorithm for joint decoding IERS codes having cubic complexity in length, and applied it to decode Hermitian codes resulting in correcting up to $t_{\max} = (N-K)/(q+1)$ bursts with failure probability $P_f(t_{\max}) < (1-R)q/(q+1)$. The algorithm for decoding Hermitian codes has complexity $\mathcal{O}(N^3)$ field operations.

Our contribution. We propose a joint decoding algorithm for interleaved *extended* RS codes. The algorithm has quadratic complexity in length. This result has interest itself.

Then we apply this algorithm to decode Hermitian codes. Our algorithm has less failure probability than the one of Ren [Ren04], and lower complexity and a better bound on decoding failure probability in comparison with Özbudak and Yayla [OY14].

We also show that low rate Hermitian codes can correct even more bursts of errors using "power" and "mixed" decoding [SSB10, WZZB12].

The rest of this chapter is organized as follows. In Section 4.1 we give a simple definition of Hermitian codes and phased bursts. In Section 4.2 we reduce decoding Hermitian codes to decoding IERS codes. Section 4.3 reminds of an idea of decoding a single *extended* RS code. This idea is generalized in Section 4.4 for *extended interleaved* RS codes. The proposed algorithm is based on decoding interleaved non extended codes described in the previous Sections 3.3 and 3.4. In Section 4.5, we shortly explain how power and mixed decoding can increase the decoding radius and/or decrease the failure probability for low rate IERS codes. Finally, in Section 4.6 we describe and analyze the decoding algorithm for Hermitian codes.

4.1 Hermitian Codes and Burst-Errors

Consider an extension field $\mathbb{F}_Q$ where $Q = q^2$ and q is a power of prime. A Hermitian curve $\mathcal{H}(q)$ over $\mathbb{F}_Q$ is defined by (c.f. [Sti88, Tie87, YB92])

$$y^q + y = x^{q+1}. \tag{4.1}$$

There are q^3 points $p = (x, y) \in \mathbb{F}_Q^2$ that satisfy (4.1) and hence lie on the Hermitian curve $\mathcal{H}(q)$. In order to list these points, denote a primitive element of $\mathbb{F}_Q$ by α and define the elements β_i as follows:

$$(\beta_1, \beta_2, \ldots, \beta_q) = \left(0, 1, \alpha^{(q+1)}, \alpha^{2(q+1)}, \ldots, \alpha^{(q-2)(q+1)}\right). \tag{4.2}$$

Let $(1, \gamma)$ be a solution to (4.1), then q^3 points $p_{i,j} = (x_j, y_{i,j})$ on the Hermitian curve $\mathcal{H}(q)$ can be written as elements of a $q \times q^2$ matrix $P = (p_{i,j})$, $i = 1, \ldots, q, j = 1, \ldots, Q$ as follows [YB92, Ren04]

$$\begin{array}{ll} p_{i,j} = (\alpha^{j-1}, \gamma\alpha^{j-1} + \beta_i), & i = 1, \ldots, q, j = 1, \ldots, Q-1, \\ p_{i,q^2} = (0, \beta_i), & i = 1, \ldots, q. \end{array} \tag{4.3}$$

We pay attention that the x coordinate of the point $p_{i,j}$ does not depend on i, and hence is denoted by x_j, and x_j runs through all Q elements of $\mathbb{F}_Q$ when $j = 1, \ldots, Q$.

To define Hermitian codes we consider bivariate (information) polynomials of the form

$$h(x,y) = f^{(1)}(x) + yf^{(2)}(x) + \cdots + y^{q-1}f^{(q)}(x), \tag{4.4}$$

where $f^{(i)}(x)$ is a polynomial over $\mathbb{F}_Q$ of restricted degree. Given a point $p = (x, y)$, by $h(p)$ we mean the evaluation $h(p) = h(x, y)$, and for the matrix $P = (p_{i,j})$ we agree that $h(P) = (h(p_{i,j}))$. For an integer $m \geq q^2 - 1$ we define the Hermitian code $\mathcal{H}_m$ as follows.

Definition 7 (Hermitian code). *For an integer $m \geq q^2 - 1$ the Hermitian code $\mathcal{H}_m$ of length $N = q^3$ over the field $\mathbb{F}_{q^2}$ is the set of $q \times q^2$ matrices W*

$$\mathcal{H}_m = \{\ W = h(P)\ \}, \tag{4.5}$$

where the matrix P is defined by (4.3), and h are all possible polynomials defined by (4.4) with degree constraints for $i = 1, \ldots, q$

$$\deg f^{(i)}(x) < k^{(i)} = \left\lfloor \frac{m - (i-1)(q+1)}{q} \right\rfloor + 1. \tag{4.6}$$

Hermitian code $\mathcal{H}_m$ is a linear (N, K) code over $\mathbb{F}_Q$. The dimension of the code is

$$K = \sum_{i=1}^{q} k^{(i)} = m - g + 1, \tag{4.7}$$

where

$$g = (q^2 - q)/2 \tag{4.8}$$

is the genus of the Hermitian curve (4.1). The code distance $\mathrm{dist}(\mathcal{H}_m)$ in the Hamming metric is lower bounded by the designed distance [FR93, SJM+95]

$$\mathrm{dist}(\mathcal{H}_m) \geq d_H = q^3 - m = N - K + 1 - g \tag{4.9}$$

and is upper bounded as $\mathrm{dist}(\mathcal{H}_m) \leq q^3 - q\lfloor m/q \rfloor$ [Ren04]. So a Hermitian code almost reaches the Singleton upper bound $N - K + 1$ on the code distance, but it is longer by factor q in comparison with RS codes, which is an advantage in many applications.

There are effective algebraic decoding algorithms, which allow correcting up to $(d_H - 1)/2$ independent errors [FR93, SJM+95]. However in this work we are focused on the correction of phased bursts of errors of length q, i.e., q-bursts. One burst-error can corrupt one column in the code matrix W, i.e., at least one element of the column is wrong.

Definition 8 (q-burst metric). *The* burst weight $w_B(V)$ *of a $q \times q^2$ matrix V is the minimum number of columns that contain all non zero components of V. The* burst distance $d_B(V, W)$ *between matrices V and W of the same size is the weight of their difference $d_B(V, W) = w_B(V - W)$.*

It follows from (4.9) and from [BS96] that the code distance d_B of an (n, k) Hermitian code in the q-burst metric is bounded by

$$(N - K + 1 - g)/q \leq d_B \leq (N - K)/q + 1 = \widehat{d}_B. \tag{4.10}$$

The code $\mathcal{H}_m$ with burst distance d_B guarantees the correction of $(d_B - 1)/2$ bursts of errors. In Section VIII we propose an algorithm correcting with high probability $\frac{q}{q+1}(\widehat{d}_B - 1)$ bursts of errors, which is almost twice the guaranteed correcting radius.

4.2 From Hermitian to Reed–Solomon Codes

Let the received matrix V be obtained from a code matrix W of a Hermitian code $\mathcal{H}_m$ by changing some elements in t columns of W. We can write $V = W + E$, where E has t nonzero columns and we assume that every nonzero column in E is equiprobable. For the error free case, $V = W$, from the definition of the code we have for all i, j

$$v_{i,j} = f^{(1)}(x_j) + y_{i,j} f^{(2)}(x_j) + \cdots + y_{i,j}^{q-1} f^{(q)}(x_j). \tag{4.11}$$

For the jth column r_j, $j = 1, \ldots, Q$, of the received matrix V we get the system of equations for unknowns $f^{(i)}(x_j)$

$$\begin{cases} f^{(1)}(x_j) + y_{1,j} f^{(2)}(x_j) + \cdots + y_{1,j}^{q-1} f^{(q)}(x_j) &= v_{1,j} \\ f^{(1)}(x_j) + y_{2,j} f^{(2)}(x_j) + \cdots + y_{2,j}^{q-1} f^{(q)}(x_j) &= v_{2,j} \\ \cdots & \\ f^{(1)}(x_j) + y_{q,j} f^{(2)}(x_j) + \cdots + y_{q,j}^{q-1} f^{(q)}(x_j) &= v_{q,j} \end{cases}. \tag{4.12}$$

It follows from (4.3) that for every fixed j, the elements $y_{i,j}$ are different. Hence the Vandermonde matrix of the system (4.12) is non-singular and we can uniquely compute the unknowns $f^{(i)}(x_j)$. By solving the system (4.12) for each of the Q columns of V we will find $f^{(i)}(x_j)$ for all i and j. Solving the system (4.12) with q unknowns by Gaussian elimination requires $\mathcal{O}(q^3)$ filed operations. Solving this system $Q = q^2$ times requires $\mathcal{O}(Qq^3) = \mathcal{O}(N^{5/3})$ operations in $\mathbb{F}_Q$, since $N = q^3$.

Let us recall the extended Reed–Solomon codes.

Definition 9 (Extended Reed–Solomon code). *If $\alpha_i = 0$ for some i, i.e., if zero locator is used, then the code is called extended Reed–Solomon (ERS) code $\mathcal{ERS}(n, k)$. An extended primitive RS code has all q field elements as locators, the length of this code is $n = q$.*

Definition 10 (Interleaved extended Reed–Solomon codes). *Given ℓ extended RS codes $\mathcal{ERS}(n+1, k^{(l)})$, $l = 1, 2, \ldots, \ell$, over $\mathbb{F}_q$ of length $n+1$ and dimensions $k^{(l)}$, the interleaved ERS (IERS) code $\mathcal{IERS}(n+1, k^{(1)}, \ldots, k^{(\ell)})$ consists of all $q \times (n+1)$ matrices C*

$$C = \begin{pmatrix} \boldsymbol{c}^{(1)} \\ \boldsymbol{c}^{(2)} \\ \vdots \\ \boldsymbol{c}^{(\ell)} \end{pmatrix} = \begin{pmatrix} f^{(1)}(x_1) & \ldots & f^{(1)}(x_{n+1}) \\ f^{(2)}(x_1) & \ldots & f^{(2)}(x_{n+1}) \\ \vdots & \vdots & \vdots \\ f^{(\ell)}(x_1) & \ldots & f^{(\ell)}(x_{n+1}) \end{pmatrix}, \tag{4.13}$$

where $\boldsymbol{c}^{(l)} \in \mathcal{ERS}(n+1, k^{(l)})$ and $x_1, \ldots, x_{n+1}$ are the code locators.

Now let us return to the decoding of Hermitian codes. By solving Q systems (4.12) we will compute $f^{(i)}(x_j)$ for all i, j and obtain for $n + 1 = Q$ a code matrix C (4.13) of the interleaved extended RS codes in the error free case.

If there is at least one error in the jth column v_j of the Hermitian codeword, then the solution $f^{(1)}(x_j), \ldots, f^{(q)}(x_j)$ of the system (4.12) will be wrong, i.e., components of the jth column in the matrix C will be replaced by other field elements, and we will get from C the received matrix R of the IERS code with the jth column wrong. If t columns of the Hermitian matrix W were corrupted, then the same t columns of received matrix R of the IERS code will be corrupted.

If we know how to decode IERS codes, then we get the following algorithm correcting burst errors in the Hermitian code. Given a received matrix V of the Hermitian code with t erroneous columns, i.e., with t bursts of errors, we first compute the matrix R of the IERS code with t erroneous columns. Second, by decoding the IERS code we find a codeword C nearest to R in the burst metric and obtain information polynomials $f^{(i)}(x)$ for all i. Since information polynomials for IERS and Hermitian codes coincide, we can transform decoding Hermitian codes with burst errors to decoding IERS codes.

In the next sections we show how to decode IERS codes.

4.3 Decoding Extended Reed–Solomon Codes

Lemma 19. *Consider an extended primitive $(n+1, k, d+1)$ RS code $\mathcal{ERS}(n+1, k)$ from Definition 9 of length $n+1 = q$ over $\mathbb{F}_q$, having all nonzero elements α_i, $i = 1, \ldots, n$, of $\mathbb{F}_q$ as nonzero code locators and the last zero locator $\alpha_{n+1} = 0$. Then the following matrix H is a parity check matrix of the code*

$$H = \begin{pmatrix} \alpha_1^{d-1} & \alpha_2^{d-1} & \ldots & \alpha_n^{d-1} & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 & 0 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{4.14}$$

where $d = n - k + 1$.

Proof: The proof is similar to Lemma 2. Since $\alpha_{n+1}^i = 0$ for $i \neq 0$ and $\alpha_{n+1}^0 = 1$, it follows from Definition 9 that the code can be generated by the following matrix of full rank k

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_n^{k-1} & 0 \end{pmatrix}. \tag{4.15}$$

The statement of the lemma follows, since H has full rank $n - k + 1$ and each row of H is orthogonal to each row of G. □

By $\widehat{\mathcal{C}}$, denote the $(n, k-1, d+1)$ RS code obtained by *shortening* the code $\mathcal{C}$ in the last position. The shortened code $\hat{\mathcal{C}}$ is defined by the parity check matrix

$$\widehat{H} = \begin{pmatrix} \alpha_1^{d-1} & \alpha_2^{d-1} & \dots & \alpha_n^{d-1} \\ \vdots & \vdots & \dots & \vdots \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ 1 & 1 & 1 & 1 \end{pmatrix}, \tag{4.16}$$

obtained from H by deleting the last column. The code $\widehat{\mathcal{C}}$ of length $Q-1$ has nonzero locators only and hence is a primitive RS code.

By $\widetilde{\mathcal{C}}$, denote the primitive (n, k, d) RS code obtained by *puncturing* the code $\mathcal{C}$ in the last position. The punctured code $\widetilde{\mathcal{C}}$ is defined by the parity check matrix

$$\widetilde{H} = \begin{pmatrix} \alpha_1^{d-1} & \alpha_2^{d-1} & \dots & \alpha_n^{d-1} \\ \vdots & \vdots & \dots & \vdots \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \end{pmatrix}, \tag{4.17}$$

obtained from H by deleting both the last column and the last row.

The extended RS code $\mathcal{C}$ can correct up to $t = \lfloor d/2 \rfloor$ errors as follows (c.f. [Bla83]). Assume a codeword $\boldsymbol{c} \in \mathcal{C}$ was transmitted and a word $\boldsymbol{r} = (r_1, r_2, \dots, r_{n+1})$ over $\mathbb{F}_Q$ containing up to t errors was received.

If the distance $d+1$ of the extended code is even, then the punctured code $\tilde{\mathcal{C}}$ has odd distance d and can correct t errors. Hence, we can correct up to t errors in the punctured word $\boldsymbol{r} = (r_1, r_2, \dots, r_n)$ using a bounded minimum distance (BMD) decoder of the classical RS code $\widetilde{\mathcal{C}}$. After decoding, the last symbol r_{n+1} can be corrected using the last parity check $c_1 + c_2 + \dots + c_{n+1} = 0$ from (4.14). Let us consider the nontrivial case of odd $d+1$.

First we compute the syndrome

$$\boldsymbol{S} = (S_1, S_2, \dots, S_{d-1}, S_d) = \boldsymbol{r}H^T = \boldsymbol{e}H^T, \tag{4.18}$$

where $\boldsymbol{e}$ is the error vector. We distinguish two cases: the last received symbol r_{n+1} is wrong or correct.

1. Assume that r_{n+1} is wrong and consider the received word $\boldsymbol{r}$ punctured in the last position, $\boldsymbol{r}' = (r_1, \dots, r_n)$ as a received word of the punctured code $\widetilde{\mathcal{C}}$. The syndrome $\widetilde{S} = (S_1, S_2, \dots, S_{d-1}) = \boldsymbol{r}'\widetilde{H}$ of the punctured code $\widetilde{\mathcal{C}}$ can be obtained directly from the known syndrome $\boldsymbol{S}$. By assumption, $\boldsymbol{r}'$ contains at most $t-1$ errors. The punctured code $\widetilde{\mathcal{C}}$ has even distance $\tilde{d} = d$. It can correct up to $t-1$ errors using a BMD decoder of a classical RS code and can detect if there were t errors. If t errors were detected, then we go to the next step, otherwise all errors in $\boldsymbol{r}'$ were corrected and the last symbol can be obtained using the last parity check $c_1 + c_2 + \dots + c_{n+1} = 0$ from (4.14).

2. Since t errors were detected in $\boldsymbol{r}'$, the last symbol r_{n+1} in word $\boldsymbol{r}$ is error free by assumption, $e_{n+1} = 0$, and we should correct t errors in the word $\boldsymbol{r}'$. To do this we consider the word $\boldsymbol{r}'$ as a received word of the shortened code $\widehat{\mathcal{C}}$ having distance $d+1$ and hence correcting t errors. It follows from (4.14) and (4.16) that the syndrome of $\boldsymbol{r}'$ in the shortened code $\widehat{\mathcal{C}}$ is $\widehat{\boldsymbol{S}} = \boldsymbol{r}'\widehat{H}^T = (e_1, e_2, \ldots, e_n)\widehat{H}^T = \boldsymbol{e}H^T = \boldsymbol{S}$ since $e_{n+1} = 0$. Using a BMD decoder of classical RS code $\widehat{\mathcal{C}}$ we will correct t errors in the word $\boldsymbol{r}'$ and obtain the correct codeword, since r_{n+1} is error free.

The drawback of the algorithm is that we use BMD decoders of RS codes twice. However, it was shown in [Bla83, Section 9.3] how to use results of the first decoding step in the second step. This method gives an algorithm for the extended code having the same complexity as a BMD decoder of a classical RS code. In the next sections we show how to extend these ideas for interleaved extended RS codes.

4.4 Interleaving of Extended Reed–Solomon Codes

Consider ℓ extended Reed–Solomon codes $\mathcal{ERS}(n+1, k^{(l)})$, $l = 1, 2, \ldots, \ell$, over $\mathbb{F} = \mathbb{F}_Q$ of length $n+1$ and dimensions $k^{(l)}$ defined by the parity check matrices $H^{(l)}$ obtained from (4.14) by replacing d with $d^{(l)} = n - k^{(l)} + 1$. The IERS code $\mathcal{IERS}(n+1, k^{(1)}, \ldots, k^{(\ell)})$ consists of all $\ell \times (n+1)$ matrices C

$$C = \begin{pmatrix} \boldsymbol{c}^{(1)} \\ \boldsymbol{c}^{(2)} \\ \vdots \\ \boldsymbol{c}^{(\ell)} \end{pmatrix} = \begin{pmatrix} c_1^{(1)} & c_2^{(1)} & \ldots & c_{n+1}^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \ldots & c_{n+1}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ c_1^{(\ell)} & c_2^{(\ell)} & \ldots & c_{n+1}^{(\ell)} \end{pmatrix},$$

where $\boldsymbol{c}^{(l)} \in \mathcal{ERS}(n+1, k^{(l)})$. Denote the received matrix R as follows

$$R = (R_1 \ldots R_{n+1}) = \left(\boldsymbol{r}^{(1)^T} \ldots \boldsymbol{r}^{(\ell)^T}\right)^T$$

and compute ℓ syndrome vectors $\boldsymbol{S}^{(l)} = \boldsymbol{r}^{(l)}H^{(l)T}$. Our goal will be to correct up to $t^+_{\max}$ columns in R, where decoding radius $t^+_{\max}$ is obtained by (3.19) using parameters of the IERS code. Let us reduce the decoding of IERS codes to the decoding of IRS codes. If we delete the last column R_{n+1} from the received matrix R we obtain interleaving of classical codes $\mathcal{RS}(n, k^{(l)}, d)$, which can be efficiently decoded. Let us consider again two cases: the last column R_{n+1} is wrong or correct.

1. Assume that R_{n+1} is wrong. Denote the received matrix punctured in the last position by $R' = (R_1, \ldots, R_n)$, and denote rows of the matrix by $\boldsymbol{r}^{(l)'}$. By assumption, there are $t-1$ erroneous columns in R'. Syndromes $\boldsymbol{S}^{(l)'}$ of punctured codes can be obtained by puncturing the syndromes $\boldsymbol{S}^{(l)}$ in the last position. We will decode the matrix R' using interleaving of ℓ punctured codes $\widetilde{\mathcal{C}}^{(l)}$, which are classical RS codes defined by

parity check matrices $\widetilde{H}^{(l)}$ (4.17), where d should be replaced by $d^{(l)} = n - k^{(l)} + 1$ to get $\widetilde{H}^{(l)}$ from $\widetilde{H}$. By decoding interleaving of ℓ codes $\widetilde{\mathcal{C}}^{(l)}$ we are able to correct up to $t_{\max}$ columns in R' according to (3.19), where from (4.21)

$$t_{\max} = t^{+}_{\max} - \ell/(\ell+1) \geq t^{+}_{\max} - 1.$$

Hence, from (3.20) decoding failure probability when correcting $t-1$ columns is

$$P_f(t) \leq \gamma Q^{-(l+1)(t_{\max}-(t-1))-1} \leq \gamma Q^{-(l+1)(t^{+}_{\max}-t)-1} \tag{4.19}$$

After this, the last column in R can be corrected using the last parity check $C_1 + \cdots + C_{n+1} = 0$ in (4.14). Since we are correcting errors beyond half the code distance, we can not detect the case when $t^{+}_{\max}$ columns of $\widetilde{R}$ are in error and hence we should go to the next step.

2. Assume that R_{n+1} is correct. By decoding R' using an interleaving of ℓ shortened codes $\widehat{\mathcal{C}}^{(l)}$ and full syndromes $\boldsymbol{S}^{(l)}$, we are able to correct up to $t^{+}_{\max}$ columns in R' according to Theorem 5, and get a correct code matrix C since the last column R_{n+1} was error free by assumption. By Theorem 5, the decoding failure probability when correcting t columns is
$$P_f(t) \leq \gamma Q^{-(l+1)(t^{+}_{\max}-t)-1}, \tag{4.20}$$
hence (4.20) gives the decoding failure probability of the complete decoder, since (4.20) coincides with the upper bound (4.19).

The proposed solution calls twice the decoder of IRS codes and hence extension of IRS codes by one symbol is twice more complex in comparison with classical RS codes. Fortunately, we are able to localize erroneous columns executing the following modification of Algorithm 4 only once.

The syndrome $\boldsymbol{S}^{(l)}$ can be obtained from $\boldsymbol{S}^{(l)'}$ by adding one more $d^{(l)}$th symbol. The Berlekamp–Massey (BM) type algorithm processes elements of a syndrome sequence sequentially to compute the polynomial $\Lambda(x)$. As a result, for the case of a single sequence, when the sequence $\boldsymbol{S}^{(l)'}$ was processed, to process the sequence $\boldsymbol{S}^{(l)}$ one should just continue the BM algorithm and make one step for the last element of $\boldsymbol{S}^{(l)}$.

In case of multiple sequences of different lengths, the application of this idea depends on the order of processing the elements of the sequences. Fortunately, the processing order in Aglorithm 4 allows applying this idea. To explain the processing order in Algorithm 4 we consider the following example of two syndrome sequences

$$\boldsymbol{S}^{(1)} = (S^{(1)}_1, S^{(1)}_2, S^{(1)}_3)$$

and

$$\boldsymbol{S}^{(2)} = (S^{(2)}_1, S^{(2)}_2, S^{(2)}_3, S^{(2)}_4, S^{(2)}_5)$$

of lengths $N^{(1)} = 3$ and $N^{(2)} = 5$ respectively. Then we obtain the maximum sequence length $N = 5$ and compose the syndrome matrix as follows

$$S = \begin{pmatrix} & & S_1^{(1)} & S_2^{(1)} & S_3^{(1)} \\ S_1^{(2)} & S_2^{(2)} & S_3^{(2)} & S_4^{(2)} & S_5^{(2)} \end{pmatrix}.$$

This means that the sequences are aligned to the *right*. In Algorithm 4 the elements $S_n^{(l)}$ in the matrix S are processed one-by-one and downwards by columns in the following order

$$S_1^{(2)}, S_2^{(2)}, S_1^{(1)}, S_3^{(2)}, S_2^{(1)}, S_4^{(2)}, S_3^{(1)}, S_5^{(2)}.$$

This processing order in Algorithm 4 is implemented in Line 7 and Line 8. The last elements of the syndrome sequences are in the last column of the syndrome matrix S which is processed column wise. This allows to simplify our decoding as it is shown by Algorithm 13, where Step 1 is implemented in Line 5 and Step 2 in Line 6. One of the steps or both can fail. By t_i and $\Lambda_i(x)$ we denote the number of errors and the error locator polynomial found in the ith step, $i = 1, 2$, respectively. For every found $\Lambda_i(x)$ we compute positions of errors in Line 8, correct them separately in each interleaved code in Line 10, and compute the code matrix in Line 11.

From the above discussion and from Theorem 5 we obtain the following theorem.

Theorem 20. *For the code $\mathcal{IERS}(n+1, k^{(1)}, \ldots, k^{(\ell)})$, Algorithm 13 corrects errors of weight t up to decoding radius $t \le \min\{t^+_{\max}, n - k_{\max} + 1\}$ with failure probability $P_f(t)$ upper bounded by (4.20), with*

$$t^+_{\max} = \frac{\ell}{\ell+1}\left(n - \overline{k} + 1\right), \tag{4.21}$$

where $\overline{k}$ and $k_{\max}$ are defined in Theorem 5.

The complexity of Algorithm 13 is also $\mathcal{O}(\ell n^2)$ operations in $\mathbb{F}_Q$ similar to the one for classical IRS codes. We localize errors calling the shift register synthesis algorithm once.

4.5 Power Decoding

To describe an idea of "power" decoding [SSB10] of an (extended) RS code, consider a code vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathcal{RS}(n, k)$ and define $\boldsymbol{c}^l = (c_0^l, \ldots, c_{n-1}^l)$ for $l = 1, 2, \ldots$. From Definition 3 it immediately follows that if $\boldsymbol{c} \in \mathcal{RS}(n, k)$ then $\boldsymbol{c}^l \in \mathcal{RS}(n, (k-1)l + 1)$ for $(k-1)l + 1 \le n$. In particular: if $\boldsymbol{c} \in \mathcal{ERS}(n, k) = \mathcal{C}$ then $\boldsymbol{c}^l \in \mathcal{ERS}(n, (k-1)l + 1) = \mathcal{C}^{(l)}$.

The idea of power decoding is as follows. We transmit a codeword $\boldsymbol{c}$ of $\mathcal{ERS}(n, k)$ and receive a word $\boldsymbol{r}$ with t errors. At the transmitter we can compute ℓ powers of $\boldsymbol{c}$ and get codewords $\boldsymbol{c}, \boldsymbol{c}^2, \ldots, \boldsymbol{c}^\ell$, from ERS codes $\mathcal{C}^{(l)}$, $l = 1, \ldots, \ell$, where ℓ is the maximum number, such that $(k-1)\ell + 1 \le n$. We can think that we virtually transmitted interleaved words $\boldsymbol{c}, \boldsymbol{c}^2, \ldots, \boldsymbol{c}^\ell$, despite in reality we transmit $\boldsymbol{c}$ only.

Algorithm 13: Decoding an IERS code

Input: Receive $R = (R_1 \dots R_{n+1}) = \left(\boldsymbol{r}^{(1)^T} \dots \boldsymbol{r}^{(\ell)^T}\right)^T$

begin

Compute syndromes $\boldsymbol{S}^{(l)} = \boldsymbol{r}^{(l)} H^{(l)T}$, $l = 1, \dots, \ell$

Run Algorithm 4 for ℓ sequences $\boldsymbol{S}^{(l)}$ of length $N^{(l)} = n - k^{(l)}$ as follows:

First run Algorithm 4 for $n = 1 \dots, N-1$ and get $t_1 = \lambda_1$ and $\Lambda_1(x)$. If $\varepsilon \neq 0$ skip $\Lambda_1(x)$

Run Lines 6-16 of Algorithm 4 for the last syndrome elements, and get $t_2 = \lambda$ and $\Lambda_2(x)$. If $\varepsilon \neq 0$ skip $\Lambda_2(x)$. If both $\Lambda_i(x)$ are skipped, declare failure

for *each not skipped* $\Lambda_i(x)$, $i = 1, 2$ **do**

Find roots $\alpha^{i_1}, \dots, \alpha^{i_\tau}$ of $\Lambda_i(x)$ in $\mathbb{F}$. If number of roots of $\Lambda_i(x)$ not equal t_i skip $\Lambda_i(x)$

for $l = 1, \dots, \ell$ **do**

Compute $\boldsymbol{c}^{(l)}$ by correcting t erasures in positions $i_1, \dots, i_t$ of $\boldsymbol{r}^{(l)}$, get the $\ell \times n$ matrix $\widetilde{C}^{(i)}$ of the punctured IRS code

Compute $C_{n+1}^{(i)} = -C_1^{(i)} - C_2^{(i)} - \dots - C_n^{(i)}$

Compute the code matrix $C^{(i)} = (\widetilde{C}^{(i)}, C_{n+1}^{(i)})$

Output: Code matrix $C^{(i)}$ nearest to R in the burst metric or decoding failure

At the receiver we compute ℓ powers of $\boldsymbol{r}$ and get virtually received words $\boldsymbol{r}, \boldsymbol{r}^2, \dots, \boldsymbol{r}^\ell$. Observe that error free positions in $\boldsymbol{r}$ stay error free in all $\boldsymbol{r}^l$ as well. Hence at the receiver we have a matrix R of interleaved ERS code with t corrupted columns. Since we know how to decode IERS code, we are able to decode the matrix R. This decoding allows to decode a single extended RS code up to the following radius for $\ell \geq 2$

$$t_{\text{power}} = \left\lfloor \frac{2\ell(n) - \ell(\ell+1)k + \ell(\ell-1)}{2(\ell+1)} \right\rfloor + 1, \tag{4.22}$$

which coincides with the Sudan decoding radius [Sud97]. The decoding failure probability for $\ell = 2$, which corresponds to code rates between 1/3 and 1/6, is given by (3.20) with a slightly different coefficient γ

$$\gamma = \frac{Q}{Q-1}\left(\frac{Q}{Q-1} + \frac{1}{Q}\right)^t \approx 1.$$

For code rates below 1/6, the failure probability can be estimated using (3.20) or (3.21) as well under the assumption that the virtual error vectors $\boldsymbol{r}^i - \boldsymbol{c}^i$, $i = 1 \dots, \ell$, are statistically independent, correctness of which is supported by simulations.

The idea of power decoding can be applied for interleaved (extended) RS codes [SSB07], if some of them have low rate. In this case every codeword of a low rate code can be virtually extended to two or more codewords. This increases the order of interleaving, the number of equations and the decoding radius or decreases the failure probability.

For the interleaving of low rate codes, the interleaving order can be increased even more using "mixed" decoding based on the following observation by Wachter–Zeh et.al. [WZZB12]. If $\boldsymbol{c}, \tilde{\boldsymbol{c}}$ are codewords of $\mathcal{RS}(n,k)$ and $\mathcal{RS}(n,\tilde{k})$ respectively, then the word with componentwise product is a codeword of an $(n, k+\tilde{k}-1)$ RS code. Simulations show that usually the new created equations are linearly independent with the original ones and one still can use (3.21) to estimate failure probability, despite this was not proved.

4.6 Decoding Hermitian Codes

Decoding of a Hermitian code is shown by Algorithm 14.

Algorithm 14: Decoding a Hermitian code

1 **Input**: Received matrix V

2 **begin**

3 For $j = 1, \ldots, Q$ solve the system (4.12) and get all $f^{(i)}(x_j)$, $i = 1, \ldots, q$ (with errors)

4 Form matrix R of IERS code (4.13)

5 Decode R by Algorithm 13, get a codeword C of IERS code

6 Find information polynomial $f^{(i)}(x)$, $i = 1, \ldots, \ell$, for every component ERS codeword $\boldsymbol{c}^{(i)}$

7 **Output**: Information polynomials $f^{(i)}(x)$, $i = 1, \ldots, \ell$, or decoding failure

Theorem 21. *For a Hermitian (N, K) code $\mathcal{H}_m$, see Definition 7, Algorithm 5 corrects t erroneous columns (bursts) up to decoding radius $t_{\max}$ with failure probability $P_f(t)$ if*

$$t \le t_{\max} = \min\left\{ \frac{N-K}{q+1}, N/q - k_{\max} \right\}, \tag{4.23}$$

where

$$k_{\max} = \max_l \{k^{(l)}\} = \lfloor m/q \rfloor + 1$$

see (4.6), and

$$P_f(t) \le \widehat{P}_f(t) = \gamma Q^{-(q+1)(t_{\max}-t)-1}, \tag{4.24}$$

$$\gamma = \left(\frac{Q^l - \frac{1}{Q}}{Q^l - 1} \right)^t \frac{Q}{Q-1} \approx 1.$$

For $t \le (N/q - k_{\max})/2$ holds $P_f(t) = 0$.

The time complexity of Algorithm 14 is $\mathcal{O}(N^{5/3})$ operations in $\mathbb{F}_Q$.

Proof: (sketch) After solving the system of linear equations (4.12) of full rank, every nonzero burst of Hermitian code is mapped in one-to-one manner to an nonzero burst of the IERS code. Hence, both codes have the same number of bursts and all nonzero bursts are equiprobable.

The Hermitian code will be successfully decoded as soon as we decode the correspondent IERS code. Hence, the decoding radius and the failure probability can be obtained from Theorems 5 and 20 by using parameters of the Hermitian and IERS codes in Theorem 5: interleaving order $\ell = q$, length of RS codes $n = Q$, dimensions $k^{(l)}$, and average dimension of RS codes is $\overline{k} = K/q$. As a result, for Algorithm 14, we obtain time complexity $\mathcal{O}(q^5) = \mathcal{O}(N^{5/3})$ operations in $\mathbb{F}_Q$, since $N = q^3$. □

Observe that when $\frac{N-K}{q+1} \leq N/q - k_{\max}$ which is frequently satisfied, we have a case which is interesting for practice:

$$t_{\max} = \frac{N-K}{q+1},$$

i.e., , we can correct $\frac{q}{q+1}(\widehat{d}_B - 1)$ bursts of errors which is very close to an upper bound $\widehat{d}_B$ (4.10) on the burst distance of the Hermitian code and is almost twice larger than the guaranteed burst correcting radius $(d_B - 1)/2$.

Decoding radius and/or failure probability can be improved for low rate codes using power or mixed decoding techniques as described in Section 4.5. Power decoding can be applied when the minimal rate $k^{(q)}/Q$ of the interleaved RS codes is at most 1/3, i.e., when the rate $r_{\mathcal{H}_m}$ of the Hermitian code is bounded by

$$r_{\mathcal{H}_m} = \frac{m-g+1}{q^3} \leq \frac{\left\lceil \frac{q^2-3}{3} \right\rceil}{q^2} + \frac{1}{2q} + \frac{1}{2q^2}. \tag{4.25}$$

Table 4.1 shows dependence of the threshold rate on q. For $q = 4$ we can apply power decoding staring from rate 1/2, however, for large q the threshold tends to 1/3. Mixed decoding can be applied when $k^{(q-1)}/Q \leq 1/3$.

	$q = 4$	$q = 8$	$q = 16$
$r_{\mathcal{H}_m}$	0.51	0.40	0.36

Table 4.1: Threshold rate for Hermitian codes to use "power" decoding

We made simulations to verify precision of our bounds for failure probability. This can be done when the number of bursts is close to the possible maximum, otherwise, the failure probability is usually too small to simulate. Our simulations show that all obtained bounds for the failure probability coincides with simulations results up to a factor < 3. Some simulations results are shown in Table 4.2.

Example 3. *To compare the performance of our algorithm with the ones by Ren [Ren04] and by Özbudak and Yayla [OY14], let us consider the example from [Ren04, OY14, YB92]*

	$\mathcal{H}_{26}(64,21,38)$	$\mathcal{H}_{30}(64,25,34)$
$t_{\max}$	8	7
t_{power}	9	8
$P_f(t_{\text{power}})$	$31056/10^6 \approx 3.1 \cdot 10^{-2}$	$34574/10^6 \approx 3.5 \cdot 10^{-2}$
$\widehat{P}_f(t_{\text{power}})$	$6 \cdot 10^{-2}$	$6 \cdot 10^{-2}$

$t_{\max}$: decoding radius without syndrome extension
t_{power} : decoding radius with syndrome extension
P_f : simulated failure probability for running 10^6 times
$\widehat{P}_f$: theoretical failure probability upper bound

Table 4.2: Failure probability $P_f(t)$ for Hermitian codes over $\mathbb{F}_{4^2}$

of rate $R = 1/2$ Hermitian code $\mathcal{H}_{37}(64, 32)$ over $\mathbb{F}_{16}$ with $q = 4$ and $Q = 16$. The decoding of the code is reduced to the decoding of $q = 4$ interleaved extended RS $(n + 1, k^{(i)}, d + 1)$ codes: $(16, 10, 7)$, $(16, 9, 8)$, $(16, 7, 10)$, and $(16, 6, 11)$.

It follows from Theorems 5 that $t_{\max} = 6.4$, hence we can correct up to 6 bursts. The failure probability in case of 6 bursts in the channel is $P_f(6) \le \gamma Q^{-3} = 2.6 \times 10^{-4}$, where $\gamma = 1.067$. In case of 5 bursts we have $P_f(5) \le \gamma Q^{-8} = 2.3 \times 10^{-10}$.

The decoding algorithm proposed by Özbudak and Yayla [OY14] also can correct up to 6 busts with failure probability $P_f(t) \le (1 - R)Q/(Q + 1) = 0.4$. Actually, for these code parameters they give a slightly better bound $P_f(t) \le 0.37$. This bound does not depend on the number of bursts t in the channel and is very weak. Their simulations show that for $t = 6$ bursts the failure probability is 1.7×10^{-3}.

Ren [Ren04] has shown one example, when his algorithm can correct 6 bursts. Let us estimate the decoding failure probability of his algorithm for $t = 6$ bursts. First, he decodes the $(16, 6, 11)$ ERS code, correcting up to 5 errors, which was the case in his example. However, with probability $(Q - 1)/Q)^6 = 0.67$ there will be 6 errors in the first word and the decoder will fail. Hence, for Ren's decoder we have $P_f(6) \ge 0.67$.

4.7 Discussion and Future Work

A joint decoding algorithm of interleaved extended RS codes is proposed, having quadratic complexity in the code length. Then we apply this algorithm to decode Hermitian codes $\mathcal{H}_m(N, K)$ over $\mathbb{F}_Q = \mathbb{F}_{q^2}$ resulting in correcting up to $t_{\max} = (N - K)/(q + 1)$ bursts with complexity $\mathcal{O}(N^{\frac{5}{3}})$ field operations. For the failure probability $P_f(t)$ we give an upper bound $P_f(t) \le \gamma Q^{-(q+1)(t_{\max}-t)-1}$, which is at most $1/Q$ when $t = t_{\max}$, and exponentially decreases when t decreases, see Theorem 21 for details. Simulations show that the bound is precise. As a result, our algorithm has less failure probability than the one of Ren [Ren04], and less complexity and better bound on the decoding failure probability in comparison with Özbudak and Yayla [OY14].

We also show that low rate Hermitian codes can correct even more bursts of errors using "power" and "mixed" decoding [SSB10, WZZB12].

5

Decoding Gabidulin Codes with Errors and Erasures

Gabidulin codes [Del78, Gab85, Rot91] are closely related to Reed–Solomon codes and have maximum possible distance in rank metric. These codes have many applications and recently they have drawn significant attention since they can provide a near-optimal solution to the error control problem in network coding [KK08, SKK08]. It was shown in [SJB11] that interleaving or the direct sum of ℓ Gabidulin codes allows decreasing the redundancy and increasing the error correcting radius nearly twice. So, it is important to be able to correct Gabidulin codes and their interleaving efficiently.

There are a number of known *error* correcting algorithms for Gabidulin codes similar to Reed–Solomon codes: a "standard" Berlekamp–Massey like approach [PT91] by Paramonov and Tretjakov [RP04] or a Sugiyama et al. like approach [Gab85] by Gabidulin, in contrast to Welch–Berlekamp like algorithm [Loi06] by Loidreau or Gao like algorithm [WZAS13] by Wachter et al. We will consider the standard approach, where first a key equation should be solved to find the "positions" of errors, and second the error vector can be computed. In contrast to Reed–Solomon codes, where the second step is relatively simple, for Gabidulin codes, the second step is very complicated to understand and to compute. In [SK09], Silva and Kschischang suggested an elegant solution for the second decoding step using a transform domain approach. This approach allows for simplification of derivation and proofs of the second decoding step, and also allows for a decrease in the decoding complexity.

In contrast to Hamming metric, it is not obvious how to define erasures in rank metric. The general definition of erasures for rank metric was proposed in [SKK08] by Silva, Kschischang and Kötter, and in [GP08] by Gabidulin and Pilipchuk. This definition arises from network coding applications and generalizes the previous definition [GPT91], where it was assumed that a number of rows and columns in the code matrix are erased. Most decoding algorithms for Gabidulin codes and for interleaved Gabidulin codes [LO06], [Ove07], [SJB11] were obtained for correcting errors only without erasures. However, in [SKK08] and [GP08] the standard decoding algorithms were extended for the case of errors and erasures. These algorithms work in the "time domain" where the second decoding step

(finding the error vector after solving the key equation) is complicated. Error and erasure correction algorithms for the transform domain have not been known so far.

In this chapter, we propose a transform domain algorithm correcting errors and erasures for Gabidulin codes. We also generalize this algorithm for interleaved Gabidulin codes. The transform-domain approach is used here because it dramatically decreases the number of the decoding steps and simplifies the proofs, so that the algorithm is more transparent. These are similar to the benefits gained from Silva and Kschischang's use of the transform-domain for decoding in the case of errors only [SK09]. However, our approach does not yield an improvement in complexity order and requires quadratic number of the field operations in the code length, though, we believe that it has more potential to be accelerated in comparison with known time domain algorithms.

5.1 Basics of Gabidulin Codes

Let $q \geq 2$ be a power of prime. Denote by $\mathbb{F}_q^{m\times n}$ the set of all $m \times n$ matrices over the finite field $\mathbb{F}_q$. A *code* $\mathcal{C}$ is defined as any nonempty subset of $\mathbb{F}_q^{m\times n}$. The distance $d(V, W)$ between two matrices $V, W \in \mathbb{F}_q^{m\times n}$ is defined as $d(V, W) = \operatorname{rank}(V - W)$. This rank distance was probably first introduced in [Hua51] as "Arithmetic distance", where it was observed that it is indeed a *metric*. The minimum distance of a code $\mathcal{C}$ is defined as the minimum rank distance between two different code matrices.

To define the class of Gabidulin codes we consider an extension field $\mathbb{F} = \mathbb{F}_{q^m}$, and the Frobenius automorphism $\theta(a) = a^q$ for $a \in \mathbb{F}_{q^m}$. For an integer i we define $\theta^i = \theta^{i \bmod m}$, $\theta^0(a) = a$, and $\theta^i(a) = \theta(\theta^{i-1}(a))$.

Given a basis of the field $\mathbb{F}_{q^m}$ over the subfield $\mathbb{F}_q$, we represent a element a of $\mathbb{F}_{q^m}$ as a column vector $\varphi(a)$ of length m over $\mathbb{F}_q$. In this way every row vector $\boldsymbol{v}$ of length n over $\mathbb{F}_{q^m}$ will be written as an $m \times n$ matrix $V = \varphi(\boldsymbol{v})$ over $\mathbb{F}_q$. The *rank norm (weight)* $\operatorname{rank} \boldsymbol{v}$ of a vector $\boldsymbol{v}$ is defined to be $\operatorname{rank} \varphi(\boldsymbol{v})$. The *rank distance* between two vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{F}_{q^m}^n$ is defined as $\operatorname{rank} \varphi(\boldsymbol{v} - \boldsymbol{w})$. Later we consider a code $\mathcal{C}$ in two equivalent forms: as the set of matrices C over the base field $\mathbb{F}_q$, or the set of vectors $\boldsymbol{c}$ over the extension field $\mathbb{F}_{q^m}$.

For $n \leq m$, let us fix the vector

$$\boldsymbol{h} = (h_1, h_2, \ldots, h_n), \tag{5.1}$$

with components $h_i \in \mathbb{F}_{q^m}$ linearly independent over $\mathbb{F}_q$, and define the following $n \times n$ transform matrix Φ over $\mathbb{F}_{q^m}$

$$\Phi = \begin{pmatrix} h_1 & h_2 & \ldots & h_n \\ \theta(h_1) & \theta(h_2) & \ldots & \theta(h_n) \\ \vdots & \vdots & \vdots & \vdots \\ \theta^{n-1}(h_1) & \theta^{n-1}(h_2) & \ldots & \theta^{n-1}(h_n) \end{pmatrix}. \tag{5.2}$$

The transform matrix Φ is nonsingular [LN83] and has the inverse matrix Φ^{-1}.

Definition 11 (Gabidulin code). *A Gabidulin code $\mathcal{G}$ in the vector form is a linear (n,k) code of length n and dimension k over the field $\mathbb{F}_{q^m}$, $n \leq m$, with parity check matrix H, i.e.,*

$$\mathcal{G}(q^m; n, k) = \{\boldsymbol{c} \in \mathbb{F}_{q^m}^n : \boldsymbol{c}H^T = 0\},$$

where the $(n-k)\times n$ parity check matrix H consists of the first $n-k$ rows of the transform matrix Φ.

Later we will give another equivalent definition of Gabidulin codes as evaluation codes.

The distance of a Gabidulin (n,k) code in rank metric is $d = n-k+1$ and it reaches the Singleton type upper bound [Gab85].

5.1.1 Skew Polynomials

To work with Gabidulin codes it will be convenient to use the skew polynomial ring of automorphism type [Ore33] which we now define. Given a field $\mathbb{F}$ and an automorphism θ of $\mathbb{F}$, one defines a ring structure on the set

$$\mathbb{F}[x;\theta] = \{a(x) = a_n x^n + \cdots + a_1 x + a_0 \mid a_i \in \mathbb{F} \text{ and } n \in \mathbb{N}\}.$$

This is the set of formal polynomials where the coefficients are written on the left of the variable x. The addition in $\mathbb{F}[x;\theta]$ is defined to be the usual addition of polynomials and the multiplication is defined by the basic rule $xa = \theta(a)x$ and extended to all elements of $\mathbb{F}[x;\theta]$ by associativity and distributivity. This means that for two skew polynomials $a(x)$ of degree d_a and $b(x)$ of degree d_b, the coefficients c_k of the product polynomial $c(x) = a(x)b(x)$ of degree $d_a + d_b$ are defined for $k = 0, 1, \ldots, d_a + d_b$ as follows :

$$c_k = \sum_{i+j=k} a_i\theta^i(b_j) = \sum_{i=\max\{0,k-d_b\}}^{\min\{k,d_a\}} a_i\theta^i(b_{k-i}) = \sum_{j=\max\{0,k-d_a\}}^{\min\{k,d_b\}} a_{k-j}\theta^{k-j}(b_j). \tag{5.3}$$

In particular, for $d_a \leq d_b$ we have

$$c_k = \sum_{i=0}^{d_a} a_i\theta^i(b_{k-i}) \quad \text{for} \quad d_a \leq k \leq d_b. \tag{5.4}$$

Given an automorphism θ, by $\mathcal{M}(N)$ we denote the complexity of the multiplication of two skew polynomials of maximum degree N. Using (5.3) this can be done with quadratic complexity $\mathcal{M}(N) = \mathcal{O}(N^2)$.

Given a skew polynomial $a(x)$ of degree n, we define a θ-reverse skew polynomial $\overline{a}(x)$ having coefficients

$$\overline{a}_i = \theta^{i-n}(a_{n-i}) \quad \text{for} \quad i = 0, ..., n. \tag{5.5}$$

In our case of the finite field and the Frobenius automorphism θ, there is a ring isomorphism between the ring $\mathbb{F}[x;\theta]$ of skew (or twisted) polynomials and the ring of *linearized*

polynomials. The last term is used more often in the coding theory literature. Given a skew polynomial $a(x) \in \mathbb{F}[x;\theta]$, we denote the corresponding linearized q-polynomial by $a_{(q)}(x)$, where

$$a_{(q)}(x) = \sum_{j=0}^{n} a_j \theta^j(x) = a_n x^{q^n} + \cdots + a_1 x^{q^1} + a_0 x,$$

and n is called the q-degree of $a_{(q)}(x)$ if $a_n \neq 0$, $\deg_q a_{(q)}(x) = n$. The polynomial $a_{(q)}(x)$ is called *monic* if $a_n = 1$. For linearized polynomials $a_{(q)}(x)$ and $b_{(q)}(x)$, the addition is the usual addition of polynomials and the (symbolic) product of linearized polynomials is defined as the composition $a_{(q)}(x) \otimes b_{(q)}(x) = a_{(q)}(b_{(q)}(x))$, with q-degree $\deg_q a_{(q)}(x) + \deg_q b_{(q)}(x)$.

Evaluation (or operator evaluation) of a skew polynomial $a(x)$ at $\alpha \in \mathbb{F}_{q^m}$ we define as evaluation $a_{(q)}(\alpha)$ of correspondent linearized polynomial, and we say that α is a root of $a(x)$ and $a_{(q)}(x)$ if $a_{(q)}(\alpha) = 0$.

Definition 12 (Gabidulin codes as evaluation). *Given $h_1, \ldots, h_n \in \mathbb{F}_{q^m}$ linearly independent over $\mathbb{F}_q$, from evaluation point of view, the Gabidulin codes is defined as follows*

$$\mathcal{G}(q^m; n, k) = \{\boldsymbol{c} = (a_{(q)}(h_1), \ldots, a_{(q)}(h_n)) \ : \ \deg a < k\}.$$

Definition 13 (Minimal polynomial). *For a set $A = \{\alpha_1, \ldots, \alpha_s\} \subseteq \mathbb{F}_{q^m}$ define the minimal linearized polynomial with respect to $\mathbb{F}_{q^m}$, denoted $M_A(x)$ as the monic linearized polynomial over $\mathbb{F}_{q^m}$ of least degree whose root space contains A. The correspondent skew polynomial is called the minimal skew polynomial and is denoted by minpoly($\boldsymbol{a}$), where the set A is written as a vector $\boldsymbol{a} = (\alpha_1, \ldots, \alpha_s)$.*

Let us find a method to compute the minimal polynomial, which we will need later for decoding. If A is a set of roots of a linearized polynomial $a_{(q)}(x)$ then $\langle A \rangle$ are also roots of $a_{(q)}(x)$, where $\langle A \rangle$ is the linear span of A over $\mathbb{F}_q$ of dimension $\dim\langle A \rangle$. Thus we can compute the unique minimal linearized polynomial as usual polynomial

$$M_A(x) = \prod_{\alpha \in \langle A \rangle} (x - \alpha)$$

of usual degree $q^{\dim\langle A \rangle}$. Hence, $M_A(x)$ has q-degree $\dim\langle A \rangle$. Since complexity of this method is exponential in $\dim\langle A \rangle$, we should consider practical methods with polynomial complexity based on the following lemma, which is similar to some results in Section VI-A-2 in [SKK08]. For a set $B = \{\beta_i\}$ denote $M_A(B) = \{M_A(\beta_i)\}$.

Lemma 22. *1. For $A = \{\alpha\} \subset \mathbb{F}_{q^m}$ holds:*

$$M_{\{\alpha\}}(x) = \begin{cases} x & \text{for} \quad \alpha = 0, \\ x^q - \alpha^{q-1} x & \text{for} \quad \alpha \neq 0. \end{cases} \tag{5.6}$$

2. Given minimal polynomials $M_A(x)$ and $M_B(x)$ for the sets $A, B \subseteq \mathbb{F}_{q^m}$, then

$$M_{A \cup B}(x) = M_{M_A(B)}(x) \otimes M_A(x). \tag{5.7}$$

Proof: Statement 1 follows from $M_{\{\alpha\}}(\alpha) = 0$. To prove Statement 2 we have to show that for $x \in A \cup B$ holds

$$M_{A\cup B}(x) = M_{M_A(B)}\left(M_A(x)\right) = 0, \tag{5.8}$$

and $\deg_q M_{A\cup B} = \dim\langle A \cup B\rangle$. Indeed, for $x = \alpha \in A$ we have $M_A(\alpha) = 0$ and (5.8) holds. For $x = \beta \in B$ we have $M_{M_A(\beta)}\left(M_A(\beta)\right) = 0$ and we get (5.8).

Since $M_A(\beta) = 0$ for $\beta \in \langle A\rangle \cap \langle B\rangle$, it follows that $\deg_q M_{M_A(B)} = \dim\langle B\rangle - \dim\{\langle A\rangle \cap \langle B\rangle\}$. Hence, from (5.7) for the degree of the product we obtain $\deg_q M_{A\cup B} = \deg_q M_A + \deg_q M_{M_A(B)} = \dim\langle A\rangle + \dim\langle B\rangle - \dim\{\langle A\rangle \cap \langle B\rangle\} = \dim\langle A \cup B\rangle$, and the statement of the lemma follows. □

Using this lemma we obtain a recurrent algorithm [SKK08], shown by Algorithm 15, to compute the minimal skew polynomial minpoly($\boldsymbol{a}$) for the set of roots, written as components of a vector $\boldsymbol{a}$ of length s.

Algorithm 15: Minimal skew polynomial `minpoly`

```
Input: Vector a = (α_1, α_2, ..., α_s) where α_i ∈ F_{q^m}
begin
    p_(q)(x) = M_{α_1}(x) according to (5.6)
    for each i from 2 to s do
        p_(q)(x) = M_{p_(q)(α_i)}(x) ⊗ p_(q)(x)
Output: p(x)
```

Algorithm 15 has complexity $\mathcal{O}(s^2)$ operations in $\mathbb{F}_{q^m}$.

All polynomials in this chapter belong to the ring $\mathbb{F}[x;\theta]$, except for their linearized polynomial equivalents denoted by the lower index (q). Later we show that to solve the key equation we need to solve the following problem of skew shift-register synthesis.

5.1.2 Skew Shift-register Synthesis

Problem 4 (Skew shift-register synthesis). *Given a field $\mathbb{F}$, an automorphism θ of the field, and ℓ sequences $\boldsymbol{S}^{(1)}, \boldsymbol{S}^{(2)}, \ldots, \boldsymbol{S}^{(\ell)}$ over $\mathbb{F}$ with lengths $N^{(1)}, N^{(2)}, \ldots, N^{(\ell)}$, respectively, i.e., $\boldsymbol{S}^{(l)} = S_1^{(l)}, S_2^{(l)}, \ldots, S_{N^{(l)}}^{(l)}$, find the smallest nonnegative integer λ for which there is a vector of coefficients $\boldsymbol{\Lambda} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_\lambda)$ over $\mathbb{F}$ such that for $l = 1, 2, \ldots, \ell$*

$$S_n^{(l)} = -\sum_{i=1}^{\lambda} \Lambda_i \theta^i \left(S_{n-i}^{(l)}\right) \quad \text{for } n = \lambda+1, \ldots, N^{(l)}. \tag{5.9}$$

Moreover, find a connection vector Λ *which fulfills (5.9) and detect when the found connection vector Λ is not unique.*

Given a connection vector $\mathbf{\Lambda} = (\Lambda_1, \ldots, \Lambda_\lambda)$ over $\mathbb{F}$, we define the corresponding connection polynomial $\Lambda(x) \in \mathbb{F}[x;\theta]$ as follows

$$\Lambda(x) \triangleq 1 + \Lambda_1 x^1 + \cdots + \Lambda_\lambda x^\lambda,$$

where $\Lambda_0 = 1$ and $\deg \Lambda(x) \leq \lambda$.

An effective Berlekamp–Massey type algorithm to solve Problem 4 was suggested in [SJB11] and is shown by Algorithm 16.

Algorithm 16: θ-skew shift-register synthesis (Problem 4)

Input: ℓ; $\boldsymbol{S}^{(l)} = S_1^{(l)}, \ldots, S_{N^{(l)}}^{(l)}$, $N^{(l)}$ for $l = 1, \ldots, \ell$

begin
- $\lambda \leftarrow 0$, $\Lambda(x) \leftarrow 1$, $N = max_l N^{(l)}$
- **for** *each* l *from* 1 *to* ℓ **do**
  - $\rho^{(l)} \leftarrow N - N^{(l)}$, $n^{(l)} \leftarrow \rho^{(l)}$, $\lambda^{(l)} \leftarrow 0$, $d^{(l)} \leftarrow 1$, $\Lambda^{(l)}(x) \leftarrow 0$
- **for** *each* n *from* 1 *to* N **do**
  - **for** *each* l *from* 1 *to* ℓ **do**
    - **if** $n > \lambda + \rho^{(l)}$ **then**
      - $d \leftarrow \sum_{j=0}^{\lambda} \Lambda_j \theta^j \left(S_{n-j-\rho^{(l)}}^{(l)} \right)$
      - **if** $d \neq 0$ **then**
        - **if** $n - \lambda \leq n^{(l)} - \lambda^{(l)}$ **then**
          - $\Lambda(x) \leftarrow \Lambda(x) - d x^{n-n^{(l)}} \frac{1}{d^{(l)}} \Lambda^{(l)}(x)$
        - **else**
          - $\widetilde{\lambda} \leftarrow \lambda$, $\widetilde{\Lambda}(x) \leftarrow \Lambda(x)$
          - $\Lambda(x) \leftarrow \Lambda(x) - d x^{n-n^{(l)}} \frac{1}{d^{(l)}} \Lambda^{(l)}(x)$
          - $\lambda \leftarrow \lambda^{(l)} + n - n^{(l)}$
          - $\lambda^{(l)} \leftarrow \widetilde{\lambda}$, $\Lambda^{(l)}(x) \leftarrow \widetilde{\Lambda}(x)$, $d^{(l)} \leftarrow d$, $n^{(l)} \leftarrow n$

Output: λ, $\Lambda(x)$; $\Lambda^{(l)}(x)$, $n^{(l)}$, $\lambda^{(l)}$ for $l = 1, \ldots, \ell$

Theorem 23 ([SJB11]). *If the output of Algorithm 16 is λ, $\Lambda(x)$, and $\Lambda^{(l)}(x)$, $n^{(l)}$, $\lambda^{(l)}$ for $l = 1, \ldots, \ell$, then the pair $(\lambda, \Lambda(x))$ is a solution of Problem 4. The solution is unique if and only if $\varepsilon = 0$, where $\varepsilon = \sum_{l=1}^{\ell} \varepsilon^{(l)}$, $\varepsilon^{(l)} = \max\{0, n^{(l)} - \lambda^{(l)} - z^{(l)} - (N - \lambda)\}$, and $z^{(l)} = \max\{0, \lambda - N^{(l)}\}$.*

Algorithm 16 has time complexity $\mathcal{O}(\ell N^2)$ operations in $\mathbb{F}$, where N is the length of a longest sequence.

A fast algorithm for solving Problem 4 was suggested in [SB14]. The asymptotic time complexity of this algorithm is $\mathcal{O}(\mathcal{M}(N) \log N)$, for a fixed number of sequences.

5.1.3 Errors and Erasures

Assume a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_{q^m}^n$ of the code $\mathcal{C} = \mathcal{G}(q^m; n, k)$ was transmitted and a word $\boldsymbol{r} = (r_1, \ldots, r_n) \in \mathbb{F}_{q^m}^n$ was received. It means that the error word is $\boldsymbol{e} = (e_1, \ldots, e_n) \in \mathbb{F}_{q^m}^n$, where $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{c}$. Denote by τ the rank of the error matrix $E = \varphi(\boldsymbol{e}) \in \mathbb{F}_q^{m \times n}$, $\tau = \operatorname{rank} E$ then we can write

$$E = AB \tag{5.10}$$

for some full rank matrices $A \in \mathbb{F}_q^{m \times \tau}$ and $B \in \mathbb{F}_q^{\tau \times n}$. Decomposition of E in (5.10) is not unique, but we use just one of them. Let $A_1, \ldots, A_\tau$ denote the columns of A and let $B_1, \ldots, B_\tau$ denote the rows of B. Then we can rewrite (5.10) as follows

$$E = AB = \sum_{i=1}^{\tau} A_i B_i. \tag{5.11}$$

The task of *error* correction is to find a codeword $\boldsymbol{c}$ such that the error matrix $E = \varphi(\boldsymbol{r} - \boldsymbol{c})$ has a minimum rank. There are a number of efficient decoding algorithms for the case when the rank of the error is less than $d/2$.

We consider a more general problem of *error and erasure* correction. Assume that the decoder has some side information from the channel about the expansion (5.11) of the error matrix E. More precisely, we assume that in the expansion (5.11), μ_{C} vectors B_j, $j \in \mathcal{J}$, $|\mathcal{J}| = \mu_{\mathrm{C}}$, and μ_{R} vectors A_i, $i \in \mathcal{I}$, $|\mathcal{I}| = \mu_{\mathrm{R}}$, are known from the channel, later we will say that there were $\mu_{\mathrm{R}} + \mu_{\mathrm{C}}$ erasures in this case. It was shown in [SKK08] that such side information is available in the case of random network coding, hence error and erasure correction is very interesting for practical applications.

Let us define errors and erasures more precisely. Without loss of generality we assume that $\mathcal{J} \cap \mathcal{I} = \emptyset$, otherwise if $i = j$, we can subtract the known component $A_i B_i$ from $R = \varphi(\boldsymbol{r})$. Let us write the known μ_{C} row vectors B_j, $j \in \mathcal{J}$, as the matrix $B_{\mathrm{C}} \in \mathbb{F}_q^{\mu_{\mathrm{C}} \times n}$ and the correspondent column vectors A_j as the matrix $A_{\mathrm{C}} \in \mathbb{F}_q^{m \times \mu_{\mathrm{C}}}$. Similarly, we write the μ_{R} known column vectors A_i, $i \in \mathcal{I}$, as the matrix $A_{\mathrm{R}} \in \mathbb{F}_q^{m \times \mu_{\mathrm{R}}}$, and the row vectors B_i as the matrix $B_{\mathrm{R}} \in \mathbb{F}_q^{\mu_{\mathrm{R}} \times n}$. Finally, the rest of the column vectors A_l and row vectors B_l we write as the matrices $A_{\mathrm{F}} \in \mathbb{F}_q^{m \times \varepsilon}$ and $B_{\mathrm{F}} \in \mathbb{F}_q^{\varepsilon \times n}$ respectively, where $\varepsilon = \tau - \mu_{\mathrm{R}} - \mu_{\mathrm{C}}$. Then we can rewrite expansion (5.11) as follows $E = A_{\mathrm{C}} B_{\mathrm{C}} + A_{\mathrm{F}} B_{\mathrm{F}} + A_{\mathrm{R}} B_{\mathrm{R}}$, where all matrices have full rank, since they are sub matrices of A and B of full rank. Now we will a give more general definition, where we do not require that all matrices have full rank. We need this more general definition for interleaved codes.

Definition 14 (Errors and erasures). *Given an error matrix E and matrices A_R with* $\operatorname{rank} A_R = \mu_R$*, and B_C with* $\operatorname{rank} B_C = \mu_C$*, we say that there were ε (full) errors, μ_C column erasures, and μ_R row erasures in the channel if the error matrix E can be represented as follows*

$$E = A_C B_C + A_F B_F + A_R B_R, \tag{5.12}$$

where sizes of matrices were defined above, and we use an expansion (5.12) that minimizes the size ε of A_F and B_F.

The decoding problem is now as follows.

Problem 5 (Decoding with errors and erasures). *Given a received matrix R and matrices A_R and B_C of erasures, find a codeword C, having the minimum number ε of full errors in expansion (5.12) of the error matrix $E = R - C$.*

I.e., similar to Hamming metric, the task of the decoder is to find a codeword nearest to the received word (minimum ε) in unerased positions.

We pay attention that expansion (5.12) is not unique and for decoding we can use any of them. We will show later in Theorem 24 that a code $\mathcal{C}$ can correct an error E of larger rank by using erasures, hence it is useful to have erasures from the channel. Note however that in general, erasures A_i and B_j from the channel are not necessarily related to the expansion of E. In this case, erasures do not help in decoding, similar to the case of codes in Hamming metric, when a correct symbol was erased. For instance, we could have an error free transmission $E = 0$ with erasures $\mu_{\mathrm{C}} + \mu_{\mathrm{R}} \geq d$, in which case the decoder will fail according to Theorem 24, but would output a correct result without using the erasures. Fortunately, this never happens in a network coding application as it was shown in [SKK08], so decoding with erasures is always at least as good and usually better than without erasures.

Definition 14 requires one to know matrices A_{R} and B_{C}. We will show that the case studied in [GPT91], where a number of columns and rows are erased in the received matrix, is included in Definition 14 as well. We describe it by the following example.

Example 4. *First, assume that we receive a matrix*

$$R = \begin{pmatrix} ? & c_{12} & ? \\ ? & c_{22} & ? \\ ? & c_{32} & ? \end{pmatrix},$$

obtained from a code matrix $C = (c_{ij})$ by erasing the first and the third columns, i.e., symbols denoted by "?" can have any value from $\mathbb{F}_q$. Then an error matrix $E = R - C$ can be written as follows

$$E = \begin{pmatrix} e_{11} & 0 & e_{13} \\ e_{21} & 0 & e_{23} \\ e_{31} & 0 & e_{33} \end{pmatrix} = \begin{pmatrix} e_{11} & e_{13} \\ e_{21} & e_{23} \\ e_{31} & e_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = A_C B_C,$$

where A_C is an unknown $m \times \mu_C$ matrix, $\mu_C = 2$, and the matrix B_C is determined by the known positions of erasures. Up to column permutation, the matrix B_C can be obtained by the $\mu_C \times \mu_C$ identity matrix appended by all zero columns to the length n. Such a matrix B_C we call an extended identity matrix. *To decode the received matrix R with column erasures the decoder will get the following input: R, $\mu_C = 2$, B_C, $\mu_R = 0$.*

Second, assume that the first and the third rows of a code matrix are erased and we receive

$$R = \begin{pmatrix} ? & ? & ? \\ c_{21} & c_{22} & c_{23} \\ ? & ? & ? \end{pmatrix}.$$

An error matrix $E = R - C$ can be written as follows

$$E = \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ 0 & 0 & 0 \\ e_{31} & e_{32} & e_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} = A_R B_R,$$

where the matrix A_R is determined by the known $\mu_R = 2$ positions of erasures. Up to row permutation, the matrix A_R can be obtained by the $\mu_R \times \mu_R$ identity matrix appended by all zero rows to the height m. We call such A_R an extended identity matrix *as well. The matrix B_R is an unknown $\mu_R \times n$ matrix over $\mathbb{F}_q$. To decode the received matrix R with row erasures the decoder will get the following input: R, $\mu_R = 2$, A_R, $\mu_C = 0$.*

Finally, consider the received matrix having two types of erasures and full errors

$$R = \begin{pmatrix} r_{11} & r_{12} & ? \\ ? & ? & ? \\ r_{31} & r_{32} & ? \end{pmatrix}.$$

In this case an error matrix E can be represented by (5.12) with $A_R = (0, 1, 0)^T$ and $B_C = (0, 0, 1)$. The decoder will get the following input: R, $\mu_R = \mu_C = 1$, A_R, B_C.

By this example we show that Definition 14 includes the previous definitions of erasures [GPT91] and explain why the terms $A_R B_R$ and $A_C B_C$ are called row and column erasures respectively.

The following theorem tells us how many errors and erasures can be corrected by the code. This theorem was proved in [SKK08], however we give here a simpler proof of the theorem.

Theorem 24 ([SKK08]). *A code $\mathcal{C}$ with minimum rank distance d corrects simultaneously ε errors, μ_C column erasures, and μ_R row erasures if $2\varepsilon + \mu_C + \mu_R \leq d - 1$.*

Proof: We have $\mu_{\mathrm{C}} + \mu_{\mathrm{R}}$ erasures and ε errors in the channel. It means that we get the erasure matrices A_{R} and B_{C} of full ranks μ_{R} and μ_{C} from the channel, respectively. According to the Definition 14, the received matrix R can be written using (5.12) as

$$R = C + A_{\mathrm{C}} B_{\mathrm{C}} + A_{\mathrm{F}} B_{\mathrm{F}} + A_{\mathrm{R}} B_{\mathrm{R}},$$

where this expansion is not unique, but as we have already mentioned, we can use any of these expansions, which have the minimum number ε of full errors, i.e., $\operatorname{rank} A_{\mathrm{F}} = \operatorname{rank} B_{\mathrm{F}} = \varepsilon$. By applying Gaussian elimination to rows of the matrix A_{R}, we can transform it to an extended identity matrix $\widetilde{A}_{\mathrm{R}}$, i.e., $\widetilde{A}_{\mathrm{R}} = U A_{\mathrm{R}}$, where U is a nonsingular $m \times m$ matrix. Similarly, we transform the matrix B_{C} to an extended identity matrix $\widetilde{B}_{\mathrm{C}}$, i.e., $\widetilde{B}_{\mathrm{C}} = B_{\mathrm{C}} V$, where V is a nonsingular $n \times n$ matrix. We have $URV = UCV + UA_{\mathrm{C}}\widetilde{B}_{\mathrm{C}} + UA_{\mathrm{F}}B_{\mathrm{F}}V + \widetilde{A}_{\mathrm{R}}B_{\mathrm{R}}V$, hence we reduce our problem to correcting ε errors and $\mu_{\mathrm{C}} + \mu_{\mathrm{R}}$ erased columns and rows at known positions in the received matrix $\widetilde{R} = URV$ by the code $\widetilde{\mathcal{C}} = \{UCV\}$, which has the same distance d since the transform matrices U, V are nonsingular.

Remove μ_{C} erased columns and μ_{R} erased rows from every code matrix $\widetilde{C}$ and from $\widetilde{R}$, and obtain a new code with distance at least $d - \mu_{\mathrm{C}} - \mu_{\mathrm{R}}$ correcting ε errors if $\varepsilon \leq (d - \mu_{\mathrm{C}} - \mu_{\mathrm{R}} - 1)/2$, and the statement of the theorem follows. □

5.2 Decoding of a Single Code

5.2.1 Key Equation for Errors and Erasures

Recall that we assume that a codeword $\boldsymbol{c} \in \mathbb{F}_{q^m}^n$ of the code $\mathcal{C} = \mathcal{G}(q^m; n, k)$ was transmitted and a word $\boldsymbol{r} \in \mathbb{F}_{q^m}^n$ was received from the channel. We consider transmission with errors and erasures according to Definition 14. It means that we obtain the vector $\boldsymbol{r}$ and matrices of erasures A_{R} and B_{C} of ranks μ_{R} and μ_{C} from the channel, respectively. The error word is $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$, where $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{c}$. We can rewrite (5.12) in a vector form as follows

$$\boldsymbol{e} = \boldsymbol{a}_{\mathrm{C}} B_{\mathrm{C}} + \boldsymbol{a}_{\mathrm{F}} B_{\mathrm{F}} + \boldsymbol{a}_{\mathrm{R}} B_{\mathrm{R}}, \tag{5.13}$$

where

$$\begin{gathered}
\boldsymbol{a}_{\mathrm{C}} \in \mathbb{F}_{q^m}^{\mu_{\mathrm{C}}}, \varphi(a_{\mathrm{C}}) = A_{\mathrm{C}}, \quad B_{\mathrm{C}} \in \mathbb{F}_q^{\mu_{\mathrm{C}} \times n}, \operatorname{rank} B_{\mathrm{C}} = \mu_{\mathrm{C}}, \\
\boldsymbol{a}_{\mathrm{F}} \in \mathbb{F}_{q^m}^{\varepsilon}, \varphi(\boldsymbol{a}_{\mathrm{F}}) = A_{\mathrm{F}}, \quad B_{\mathrm{F}} \in \mathbb{F}_q^{\varepsilon \times n}, \operatorname{rank} \boldsymbol{a}_{\mathrm{F}} = \operatorname{rank} B_{\mathrm{F}} = \varepsilon, \\
\boldsymbol{a}_{\mathrm{R}} \in \mathbb{F}_{q^m}^{\mu_{\mathrm{R}}}, \varphi(\boldsymbol{a}_{\mathrm{R}}) = A_{\mathrm{R}}, \quad B_{\mathrm{R}} \in \mathbb{F}_q^{\mu_{\mathrm{R}} \times n}, \operatorname{rank} \boldsymbol{a}_{\mathrm{R}} = \mu_{\mathrm{R}}.
\end{gathered}$$

In [SKK08] and [GP08] the following definitions were introduced to get a modified key equation incorporating errors and erasures. Given a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ and a skew-polynomial $p(x)$, we use the notation $p(\boldsymbol{v}) = (p(v_1), \ldots, p(v_n))$. Let us introduce skew polynomials of row erasures

$$\Lambda_{\mathrm{R}}(x) = \operatorname{minpoly}(\boldsymbol{a}_{\mathrm{R}}), \quad \deg \Lambda_{\mathrm{R}}(x) = \mu_{\mathrm{R}}, \tag{5.14}$$

of full errors

$$\Lambda_{\mathrm{F}}(x) = \operatorname{minpoly}(\Lambda_{\mathrm{R}(q)}(\boldsymbol{a}_{\mathrm{F}})), \quad \deg \Lambda_{\mathrm{F}}(x) = \varepsilon \tag{5.15}$$

and skew polynomial

$$\Lambda_{\mathrm{FR}}(x) = \Lambda_{\mathrm{F}}(x) \Lambda_{\mathrm{R}}(x), \tag{5.16}$$

which has components of $\boldsymbol{a}_{\mathrm{R}}$ and $\boldsymbol{a}_{\mathrm{F}}$ as roots due to Lemma 22.

Using $\boldsymbol{h}$ as in (5.1) we define

$$\boldsymbol{f} = (f_1, \ldots, f_{\mu_{\mathrm{C}}}) = \boldsymbol{h} B_{\mathrm{C}}^T,$$

and the polynomial of column erasures

$$\lambda_{\mathrm{C}}(x) = \operatorname{minpoly}(\boldsymbol{f}). \tag{5.17}$$

We compute the syndrome vector $\boldsymbol{S}$ as usual

$$S = (S_1, \ldots, S_{d-1}) = \boldsymbol{r} H^T = \boldsymbol{e} H^T, \tag{5.18}$$

and introduce the syndrome polynomial $S(x)$

$$S(x) = \sum_{i=1}^{n-k} S_i x^{i-1}. \tag{5.19}$$

Like in [SKK08], we define the modified syndrome polynomial $s_{\mathrm{RC}}(x)$, which incorporates known information about row and column erasures

$$S_{\mathrm{RC}}(x) = \Lambda_{\mathrm{R}}(x) S(x) \overline{\lambda}_{\mathrm{C}}(x) \tag{5.20}$$

and get the key equation.

Theorem 25 (Key equation for Gabidulin codes [SKK08]). *Assume ε errors and $\mu_C + \mu_R$ erasures in the word to be decoded with Gabidulin (n, k) code. Then the following equation holds*

$$\Lambda_F(x) S_{RC}(x) \equiv \Omega(x) \mod x^{n-k}, \tag{5.21}$$

where the error evaluator polynomial $\Omega(x)$ has $\deg \Omega(x) < \tau \triangleq \varepsilon + \mu_C + \mu_R$ and is defined by the first τ components of the syndrome $S(x)$.

Recall that $\deg \Lambda_{\mathrm{F}}(x) = \varepsilon$ by definition. Here and later mod means the right modulo operation in $\mathbb{F}[x; \theta]$.

From Theorem 25 it follows that monomials $\Omega_j x^j$ of the polynomial $\Omega(x)$ vanish for $j \geq \tau$. Using (5.4) for the product $\Lambda_{\mathrm{F}}(x) S_{\mathrm{RC}}(x)$, which gives coefficients Ω_j, we obtain an equivalent form of the key equation.

Corollary 26. *Assume ε errors and $\mu_C + \mu_R$ erasures in the word decoded by Gabidulin (n, k) code. Then the coefficients of polynomials $\Lambda_F(x)$ and $S_{RC}(x)$ for $j = \mu_C + \mu_R + \varepsilon + 1, \ldots, n - k$ satisfy*

$$\sum_{i=0}^{\varepsilon} \Lambda_{F,i} \, \theta^i(S_{RC,j-i}) = 0 \, . \tag{5.22}$$

To solve the key equation (5.21) or (5.22) one should find the minimum number of errors ε, for which a solution $\Lambda_{\mathrm{F}}(x)$ of the key equation exists. This means that we should find a minimum non-negative ε, for which

$$S_{\mathrm{RC},j} = -\sum_{i=1}^{\varepsilon} \Lambda_{\mathrm{F},i} \, \theta^i(S_{\mathrm{RC},j-i}) \text{ for } j = \mu_{\mathrm{C}} + \mu_{\mathrm{R}} + \varepsilon + 1, \ldots, n - k. \tag{5.23}$$

This means that we should solve Problem 4 for the sequence $\boldsymbol{S}^{(1)}$ of length $n - k - \mu_{\mathrm{C}} - \mu_{\mathrm{R}}$ obtained from the modified syndrome with removed first $\mu_{\mathrm{C}} + \mu_{\mathrm{R}}$ elements. Recall that we enumerate elements of the sequences $\boldsymbol{S}$ and $\boldsymbol{S}_{\mathrm{RC}}$ starting from 1.

5.2.2 Decoding in the Transform Domain

Let us introduce the transformed error vector and polynomial

$$\boldsymbol{E} = \boldsymbol{e} \Phi^T, \quad E(x) = \sum_{i=1}^{n} \widetilde{E}_i x^{i-1}. \tag{5.24}$$

To find the transformed error vector $\boldsymbol{E}$ and then the error vector $\boldsymbol{e}$, we propose the following key equation for $\boldsymbol{E}$.

Theorem 27. *The transformed error polynomial $E(x)$ satisfies the following equation*

$$\Lambda_{FR}(x)E(x)\overline{\lambda}_C(x) \equiv \Omega(x) \mod x^n, \tag{5.25}$$

where the polynomial $\Omega(x)$ is defined by (5.21).

Proof: Given a Gabidulin (n, k) code and an error vector $\boldsymbol{e}$ with fixed decomposition (5.12), then from Theorem 25, we can compute the error evaluator polynomial $\Omega(x)$ as follows

$$\Omega(x) \equiv \Lambda_F(x)S_{RC}(x) \mod x^{n-k}.$$

From (5.20) and (5.16)

$$\Lambda_F(x)S_{RC}(x) = \Lambda_F(x)\Lambda_R(x)S(x)\overline{\lambda}_C(x) = \Lambda_{FR}(x)S(x)\overline{\lambda}_C(x),$$

and we obtain

$$\Lambda_{FR}(x)S(x)\overline{\lambda}_C(x) \equiv \Omega(x) \mod x^{n-k}. \tag{5.26}$$

Consider another Gabidulin $(n, 0$ code with $k' = 0$ and the same error vector $\boldsymbol{e}$ as before with the same decomposition as in (5.12). In this case, the polynomials $\Lambda_{FR}(x)$ and $\overline{\lambda}_C(x)$ stay unchanged. The parity check matrix H' of this code is $H' = \Phi$, and from (5.18) the syndrome vector $\boldsymbol{S}'$ of length n is the same as the transformed error vector $\widetilde{\boldsymbol{e}}$. Since the first $n - k$ components of syndromes $\boldsymbol{S}$ and $\boldsymbol{S}' = \boldsymbol{E}$ coincide, the polynomial $\Omega'(x)$ for the new code stay unchanged by Theorem 25, i.e., $\Omega'(x) = \Omega(x)$. As a result, the theorem statement follows from (5.26) and Theorem 25 applied to the new code $\mathcal{G}'$. □

We are ready to describe our decoding algorithm shown by Algorithm 17.

In Lines 3-6 we compute erasure polynomials $\Lambda_{R(q)}(x)$, $\lambda_{C(q)}(x)$ and the modified syndrome $S_{RC}(x)$ using (5.14), (5.17) and (5.20).

In Line 7 we find $\Lambda_F(x)$ by solving the key equation (5.22). To do this we observe that according to Corollary 26 solving the key equation is equivalent to solving Problem 4 for the single $\ell = 1$ sequence $\boldsymbol{S}^{(1)}$ of length $N^{(1)} = n - k - \mu_{\mathrm{C}} - \mu_{\mathrm{R}}$ obtained from the modified syndrome with removed first $\mu_{\mathrm{C}} + \mu_{\mathrm{R}}$ elements. Problem 4 can be solved using Algorithm 16 (see also [SRB11]) or by fast algorithm [SB14], and we get the number of full errors $\varepsilon = \lambda$, and the full error span polynomial $\Lambda_F(x) = \Lambda(x)$. In addition, this algorithm indicates when the found polynomial $\Lambda(x)$ is not unique. In this case we declare a decoding failure.

In Lines 8-9 we compute the error evaluator polynomial $\Omega(x) = \Lambda_F(x)s_{RC}(x) \mod x^{n-k}$ using (5.21) from Theorem 25 and the polynomial $\Lambda_{FR}(x)$ using (5.16).

In Line 10 we compute the transformed error word using Theorem 27 by left and right power series expansion $E(x) = \Lambda_{FR}^{-1}(x)\, w(x)\, \overline{\lambda}_C^{-1}(x) \;|_0^{n-1}$ of skew polynomials, where $a^{-1}(x)b(x)c^{-1}(x) = f(x)$ means that $b(x) = a(x)f(x)c(x)$, and $f(x)|_0^{n-1}$ means that from the power series $f(x)$ we only take items $f_i x^i$ for $i = 0, \ldots, n-1$.

In Line 11 we compute the error word $\boldsymbol{e} = E\,(\Phi^{-1})^T$ by the inverse transform.

From the above derivations and from Theorem 24 similar to [SKK08] and [GP08] we obtain the following theorem.

Algorithm 17: Decoding a Gabidulin code

1 **Input**: Received word $\boldsymbol{r} \in \mathbb{F}_{q^m}^n$, vector $\boldsymbol{a}_R$ of row erasures , matrix B_C of column erasures

2 **begin**

3 Row erasure polynomial: $\Lambda_{R(q)}(x) = \text{minpoly}(\boldsymbol{a}_R)$

4 Column erasure polynomial: $\boldsymbol{f} = \boldsymbol{h}B_C^T$, $\lambda_{C(q)}(x) = \text{minpoly}(f)$

5 Syndrome: $\boldsymbol{S} = \boldsymbol{r}H^T$

6 Modified syndrome: $S_{RC}(x) = \Lambda_R(x)S(x)\overline{\lambda}_C(x)$

7 Find $\Lambda_F(x)$ by solving the key equation (5.22) using the Berlekamp–Massey type Algorithm 16; in case of multiple solutions, output decoding failure

8 The error evaluator polynomial $\Omega(x) = \Lambda_F(x)S_{RC}(x) \mod x^{n-k}$

9 $\Lambda_{FR}(x) = \Lambda_F(x)\Lambda_R(x)$

10 The transformed error word $E(x) = \Lambda_{FR}^{-1}(x)\, w(x)\, \overline{\lambda}_C^{-1}(x) \,|_0^{n-1}$

11 The error word $\boldsymbol{e} = E\left(\Phi^{-1}\right)^T$

12 **Output**: The codeword $\boldsymbol{c} = \boldsymbol{r} - \boldsymbol{e}$ or decoding failure

Theorem 28. *Algorithm 17 corrects ε full errors, μ_R row erasures and μ_C column erasures as long as*

$$2\varepsilon + \mu_R + \mu_C \leq n - k = d - 1. \tag{5.27}$$

Time complexity of Algorithm 17 is $\mathcal{O}(n^2)$ operations in $\mathbb{F}_{q^m}$ for m, n, d of the same order, see also [SKK08] and [GY08] for details. We would like to emphasize that after the key equation is solved in Line 7 and the error span polynomial $\Lambda_F(x)$ is found, we need only two polynomial multiplications and two power series expansions, which is equivalent to two divisions of skew polynomials to find the error word in the transform domain. Then the error vector can be computed by the low complexity inverse transform [SK09]. Recall that in previous algorithms [SKK08] and [GP08] instead of these steps (Lines 8-11) one should do the following more complicated steps:

- One polynomial multiplication to find $\Lambda_{FR}(x)$,
- Solve a system of linear equations (41) in [SKK08] to find $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{\mu_C})$,
- Compute $\Lambda_{C(q)}(x)$ =minpoly($\boldsymbol{\beta}$),
- Compute $\Lambda(x) = \Lambda_C(x)\Lambda_F(x)\Lambda_R(x)$,
- Find a basis for the root space of $\Lambda_{(q)}(x)$,
- Solve a system of linear equations (36) in [SKK08] to find error locators,
- Compute error locations,

- Compute the error word.

As a result, the algorithms in [SKK08] and [GP08] are more difficult for understanding, despite the fact that they have the same order of time complexity $\mathcal{O}(n^2)$.

5.3 Interleaved Gabidulin Codes

Let us consider interleaving of ℓ in general different $(n^{(l)}, k^{(l)})$ Gabidulin codes over $\mathbb{F}_{q^m}$. Let us agree that the index l runs always from 1 to ℓ, i.e., $l = 1, \ldots, \ell$. Denote by $c = \left(c^{(1)} \ \ldots \ c^{(\ell)}\right) \in \mathbb{F}_{q^m}^{\ell n}$ the concatenation of ℓ vectors $c^{(l)}$.

Definition 15. *The ℓ-interleaved Gabidulin code is*

$$\mathcal{IG}(q^m, \ell; n, k^{(1)} \ldots \ k^{(\ell)}) \triangleq \left\{ \left(\boldsymbol{c}^{(1)} \ldots \ \boldsymbol{c}^{(\ell)}\right) : \boldsymbol{c}^{(l)} \in \mathcal{G}(q^m; n, k^{(l)}) \right\}.$$

The interleaved code is defined over $\mathbb{F}_{q^m}$ and has length ℓn. For the case of identical codes, $k^{(l)} = k$, the distance of the interleaved code in rank metric is $d = n - k + 1$, which reaches the Singleton upper bound if $m = n$. Recall that Loidreau and Overbeck ([LO06, Ove07]) suggested another construction of an interleaved code, which is over $\mathbb{F}_{q^{\ell m}}$ and has length n.

Assume a codeword $\boldsymbol{c} \in \mathcal{IG}(q^m, \ell; n, k^{(1)} \ldots \ k^{(\ell)})$ was transmitted and a word $\boldsymbol{r} = \left(\boldsymbol{r}^{(1)} \ldots \ \boldsymbol{r}^{(\ell)}\right) \in \mathbb{F}_{q^m}^{\ell n}$ was received. It means that the error word is $\boldsymbol{e} = \left(\boldsymbol{e}^{(1)} \ldots \ \boldsymbol{e}^{(\ell)}\right) \in \mathbb{F}_{q^m}^{\ell n}$, where $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{c}$. Denote $\operatorname{rank} \boldsymbol{e} = \tau$, then we can represent the error word as

$$\boldsymbol{e} = \boldsymbol{a}B, \quad \text{where } \boldsymbol{a} \in \mathbb{F}_{q^m}^{\tau}, \ B \in \mathbb{F}_q^{\tau \times \ell n},$$

where for every component code we have the following error vector

$$\boldsymbol{e}^{(l)} = \boldsymbol{a}_{\mathrm{F}} B_{\mathrm{F}}^{(l)} + \boldsymbol{a}_{\mathrm{R}} B_{\mathrm{R}}^{(l)} + \boldsymbol{a}_{\mathrm{C}} B_{\mathrm{C}}^{(l)}, \tag{5.28}$$

where $\boldsymbol{a}_{\mathrm{R}}$, and $B_{\mathrm{C}}^{(l)}$ are known, and hence $\mu_{\mathrm{R}} = \operatorname{rank} \boldsymbol{a}_{\mathrm{R}}$ and ${\mu_{\mathrm{C}}}^{(l)} = \operatorname{rank} B_{\mathrm{C}}^{(l)}$ are also known, $\operatorname{rank} \boldsymbol{a}_{\mathrm{F}} = \varepsilon$, $\operatorname{rank} B_{\mathrm{F}}^{(l)} \leq \varepsilon$.

We define error and erasure span polynomials and syndrome polynomials in a way similar to a single code. We exploit the fact that the vector $\boldsymbol{a}$ is common for all interleaved codes, and this allows us to get more equations for the common error span polynomial $\Lambda_{\mathrm{F}}(x)$, defined by the common vector $\boldsymbol{a}_{\mathrm{F}}$. In a way similar to a single code, we obtain a key equation for the interleaved Gabidulin codes as follows.

Theorem 29 (Key equation for interleaved Gabidulin codes). *The following equation holds for $l = 1, \ldots, \ell$*

$$\Lambda_F(x) S_{RC}^{(l)}(x) \equiv \Omega^{(l)}(x) \mod x^{n-k^{(l)}}, \tag{5.29}$$

where the error evaluator polynomial $\Omega^{(l)}(x)$ is defined by the first $\varepsilon + \mu_R + {\mu_C}^{(l)}$ components of the modified syndrome $S_{RC}^{(l)}(x)$, and $\deg \Omega^{(l)}(x) < \varepsilon + \mu_R + {\mu_C}^{(l)}$.

A decoding algorithm for interleaved Gabidulin codes is given by Algorithm 18, which is similar to the one for a single code. In Line 8 of the new algorithm, to solve the key equation (5.29) we solve Problem 4 using Algorithm 16 for ℓ sequences $\boldsymbol{S}^{(l)}$ of length $N^{(l)} = n - k^{(l)} - {\mu_{\mathrm{C}}}^{(l)} - \mu_{\mathrm{R}}$ obtained from the modified syndrome $S^{(l)}_{\mathrm{RC}}(x)$ with removed first ${\mu_{\mathrm{C}}}^{(l)} + \mu_{\mathrm{R}}$ elements.

After the key equation (5.29) is solved and the error span polynomial $\Lambda_{\mathrm{F}}(x)$ is found, the error words $\boldsymbol{e}^{(l)}$ can be computed separately for every interleaved code as in Algorithm 17 applying Theorem 27 for every component code:

$$\Lambda_{\mathrm{FR}}(x) E^{(l)}(x) \overline{\lambda}^{(l)}_{\mathrm{C}}(x) \equiv \Omega^{(l)}(x) \mod x^n, \tag{5.30}$$

where the polynomials $\Omega^{(l)}(x)$ are defined by (5.29).

Algorithm 18: Decoding an interleaved Gabidulin code

1 **Input**: Received word $\boldsymbol{r} = (\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(\ell)}) \in \mathbb{F}_{q^m}^{\ell n}$, $\boldsymbol{a}_R$, $B_C^{(l)}$, $l = 1, \ldots, \ell$

2 **begin**

3 Row erasure polynomial: $\Lambda_{R(q)}(x) = \mathrm{minpoly}(\boldsymbol{a}_R)$

4 **for** $l = 1, \ldots, \ell$ **do**

5 Column erasure polynomials: $\boldsymbol{f}^{(l)} = \boldsymbol{h} B_C^{(l)T}$, $\lambda^{(l)}_{C(q)}(x) = \mathrm{minpoly}(\boldsymbol{f}^{(l)})$

6 Syndromes: $\boldsymbol{S}^{(l)} = \boldsymbol{r}^{(l)} H^{(l)T}$

7 Modified syndromes: $S^{(l)}_{RC}(x) = \Lambda_R(x) S^{(l)}(x) \overline{\lambda}^{(l)}_C(x)$

8 Find $\Lambda_F(x)$ by solving the key equation (5.29) using the Berlekamp–Massey type Algorithm 16; in case of non single solution output decoding failure

9 **for** $l = 1, \ldots, \ell$ **do**

10 The error evaluator polynomial: $\Omega^{(l)}(x) = \Lambda_F(x) S^{(l)}_{RC}(x) \mod x^{n-k^{(l)}}$

11 $\Lambda_{FR}(x) = \Lambda_F(x) \Lambda_R(x)$

12 The transformed error: $E^{(l)}(x) = (\Lambda_{FR}(x))^{-1} \; w^{(l)}(x) \left(\overline{\lambda}^{(l)}_C(x) \right)^{-1} \; |_0^{n-1}$

13 The error word: $\boldsymbol{e}^{(l)} = E^{(l)} \left(\Phi^{-1}\right)^T$

14 $\boldsymbol{e} = (\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(\ell)})$

15 **Output**: The codeword $\boldsymbol{c} = \boldsymbol{r} - \boldsymbol{e}$ or decoding failure

A time domain error correcting algorithm (without erasures) for interleaved Gabidulin codes was proposed in [SJB11], see Algorithm 4. Let us compare our Algorithm 3 and Algorithm 4 in [SJB11]. In contrast to [SJB11], our algorithm corrects errors and erasures. To proceed with it, in Lines 3-7 we reduce the decoding problem to multi-sequence skew shift register synthesis (solving the key equation). Then in Line 8 we make this synthesis, like in [SJB11], where we can get a failure, and hence, we can use the failure probability bound from [SJB11]. Then in Lines 9-13 we compute error words $\boldsymbol{e}^{(l)}$ separately for every

interleaved code using transform domain approach, which decreases the number of steps in comparison with time domain method used in [SJB11]. Since the failure probability and the decoding radius are determined by solving the key equations, which coincide with the ones in [SJB11], we can summarize our results using [SJB11] as follows.

Theorem 30. *The fraction $P_f(\varepsilon)$ of full error vectors of $\operatorname{rank} e_F = \varepsilon$ uncorrectable by Algorithm 18 in presence of μ_R row erasures and $\mu_C{}^{(1)}, \ldots, \mu_C{}^{(\ell)}$ column erasures is upper bounded by*

$$P_f(\varepsilon) \leq 3.5q^{-m\{(\ell+1)(\varepsilon_{max}-\varepsilon)+1\}} < \frac{4}{q^m}$$

if

$$\ell \leq \varepsilon \leq \varepsilon_{max} \triangleq \frac{\ell}{\ell+1}(\overline{d}-1) \tag{5.31}$$

and

$$P_f(\varepsilon) = 0 \text{ for } \varepsilon < d_{min}/2, \tag{5.32}$$

where

$$\overline{d} = \frac{1}{\ell}\sum_{l=1}^{\ell} d^{(l)} - \mu_R - \mu_C{}^{(l)}, \quad d_{min} = \min_l\{d^{(l)} - \mu_R - \mu_C{}^{(l)}\}$$

are the average and minimum code distances respectively after erasings in interleaved $(n^{(l)}, k^{(l)})$ Gabidulin codes having distances $d^{(l)} = n^{(l)} - k^{(l)} + 1$.

This means that errors of rank weight less than $d_{\min}/2$ are always corrected by Algorithm 18, and all other errors of weight up to the decoding radius $\varepsilon_{\max}$ are corrected with very high probability. For the fixed order of interleaving ℓ, the asymptotic complexity of Algorithm 18 is the same as the one of Algorithm 17.

5.4 Discussion and Future Work

For a Gabidulin code we propose a transform domain algorithm correcting errors and erasures. We also give a generalization of the algorithm for interleaved Gabidulin codes.

For a single code and the interleaved codes, if the complexity of polynomial multiplication is $\mathcal{M}(N) = N^2$, then these algorithms can be straightforwardly implemented with time complexity $\mathcal{O}(n^2)$ operations over $\mathbb{F}_{q^m}$ and $\mathcal{O}(\ell n^2)$, respectively. Therefore, for fixed ℓ, both of the algorithms have quadratic complexity in the code length, i.e., $\mathcal{O}(n^2)$. The order of complexity is the same as the one in [SJB11], and is less than the one in [LO06, Ove07], which has order $\mathcal{O}(n^3)$.

The proposed algorithm has the potential to be accelerated and we are working in this direction. Fast methods for multiplication of linearized polynomials were found in [WZAS13] and [SK09]. A fast solution of the key equations was suggested in [SB14]. We still need to find a fast method for finding a minimal degree linearized polynomial, given a

basis of its roots, and a fast method for left and right series power expansion or for left and right division of skew polynomials. Afterwards, we will have a fast decoding algorithm having sub-quadratic complexity in the code length.

6

Decoding Chinese Remainder Codes

In this chapter, we consider another class of evaluation codes — Chinese remainder (CR) codes. The name comes from the ancient Chinese remainder theorem which was found in a the 5th-century book "Sunzi's Mathematical Classic" (Figure 6.1). The Chinese remainder theorem was used to determine an integer number $0 \leq M < \prod_{i=0}^{k} p_i$ that when divided by some given co-prime divisors $\{p_1, p_2, ..., p_k\}$ gives remainders $\{m_1, m_2, \ldots, m_k\}$. Therefore, given divisors, the number M is uniquely reconstructed by a set of k remainders.

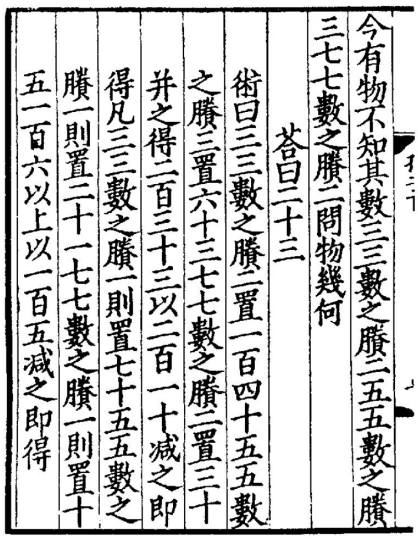

今有物不知其數三三數之賸二五五數之賸三七七數之賸二問物幾何
答曰二十三
術曰三三數之賸二置一百四十五五數之賸三置六十三七七數之賸二置三十并之得二百三十三以二百一十減之即得凡三三數之賸一則置七十五五數之賸一則置二十一七七數之賸一則置十五一百六以上以一百五減之即得

Figure 6.1: The Chinese remainder theorem in "Sunzi's Mathematical Classic".

The Chinese remainder code is defined as follows. Given $n \geq k$ co-prime numbers (divisors) p_i where $p_1 < p_2 < \cdots < p_n$ and an integer number $0 \leq M < \prod_{i=0}^{k} p_i$, the codeword is the sequence of n remainders, i.e., $c_i = M \mod p_i$, denoted by $(c_1, \ldots, c_k, c_{k+1}, \ldots, c_n)$. The codeword is transmitted over the channel. If the channel is noiseless, then M is determined by Chinese remainder theorem from any k components of the received word. For erroneous channel, the CR codes can correct errors up to half the minimum Hamming distance, when divisors do not differ too much.

The redundancy of the Chinese remainder representation of integers has been exploited often in theoretical computer science and in many practical applications. It was shown by Goldreich, Ron and Sudan [GRS00], that CR codes can be efficiently applied for distributive computations, secret sharing, and permanent computation of random matrices.

In computer science, the Chinese remainder code is usually called residue number system (RNS) which has been applied in the fields of computer arithmetic [OP07] and digital signal processing [SJJT86].

As a class of evaluation codes, CR codes are similar to RS codes and Gabidulin codes in many respects. In particular, all these codes are maximum distance separable, and the key equation of CR codes are analogous to those of RS and Gabidulin codes. One important difference of CR codes from others is that, the CR codes are defined over the integer ring, whereas the RS and Gabidulin codes are defined over finite field. One can still use algorithms created for RS codes for reference to decode the CR codes, nevertheless, the realization can be slightly or completely different. In fact, several decoding algorithms have already been studied to decode a single CR code. For example, Mandelbaum ([Man76, Man78]) gave a decoding algorithm for the Chinese remainder codes, correcting up to $(d-1)/2$ errors, where d is the code distance in the Hamming metric. Unfortunately the complexity of this algorithm is exponential in general. A more recent research in [GRS00, GSS00] propose an unique decoder for CR codes which has almost linear complexity in the bit length of N where $N = \prod_{i=1}^{n} p_i$.

Recently, the construction of interleaved RS (IRS) codes have been intensively studied in several publications, e.g., [SSB09, SSB07] where such construction allow decoding beyond half the minimum distance and can be applied in concatenated designs. Among decoding algorithms for IRS codes, Nielsen [Nie13b] proposes a module minimization approach for solving multiple key equations by finding short vectors in a certain space.

Algebraic similarities of the interleaved CR (ICR) codes and IRS codes mean that we can apply the method in [Nie13b] for ICR codes to solve multiple key equations by finding short vectors in a certain space; on the other hand, algebraic differences of these two codes result in that the entire analysis is different.

The rest of this chapter is structured as follows. The introduction of CR codes and some known results are stated in Section 6.1. To decode a single CR code, we propose a unique decoder based on syndrome in Section 6.2 and a decoding algorithm for correcting both errors and erasures in Section 6.3. Since the CR codes profits from interleaving as well, we consider interleaved CR (ICR) codes, and propose the corresponding decoding algorithm in the Section 6.4.

6.1 Basics of CR Codes

We begin with defining the classical Chinese Remainder codes. Let n be the code length and

$$0 < p_1 < p_2 < \cdots < p_n \tag{6.1}$$

be a list $\mathcal{P}$ of n relatively prime positive integers. We construct a polyalphabetic code, where the i-th component of codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ is taken from the alphabet $\mathbb{Z}_{p_i}$, being the ring of integers modulo p_i. Thus the codewords are selected from the code space $\mathbb{Z}_{\mathcal{P}} = \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \cdots \times \mathbb{Z}_{p_n}$ of size $N = p_1 p_2 \ldots p_n$. Given $\mathcal{P}$, let us define the function $F(a, b)$ for integers a, b, where $0 < a \leq b \leq n$, as follows

$$F(a, b) = \prod_{i=a}^{b} p_i, \tag{6.2}$$

and for $a > b$, $F(a, b) = 0$. So, we have

$$N = F(1, n). \tag{6.3}$$

We also need a *cardinality* of our code K; we mostly deal with the classical case where K is selected as $K = F(1, k)$ for some $0 < k < n$. Such an integer k can be regarded as the number of information symbols of the code. For integers x and $y \neq 0$, we denote by $[x]_y$ the remainder when x is divided by y, $0 \leq [x]_y \leq y - 1$.

Definition 16 (Chinese remainder code). *A Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ or shortly $\mathcal{CR}(n, K)$ having cardinality $K = F(1, k)$ for some k, $0 < k < n$ and length n over alphabets $\mathcal{P}$ is defined as follows*

$$\mathcal{CR}(\mathcal{P}; n, K) = \{\ ([C]_{p_1}, \ldots, [C]_{p_n}) : C \in \mathbb{N} \text{ and } C < K\}.$$

Example 5. *Given a list $\mathcal{P} = \{3, 5, 7, 11, 13\}$ and $k = 2$. Consider a CR code $\mathcal{CR}(\mathcal{P}; n, K)$ of length $n = 5$, with cardinality $K = F(1, k) = 15$. For the message integer $0 \leq C = 14 < K$, the codeword*

$$\boldsymbol{c} = ([14]_3, \ldots, [14]_{13}) = (2, 4, 0, 3, 1).$$

In [SSG+05], upper and lower bounds on the cardinality of a polyalphabetic code with given Hamming distance were analyzed, for instance, the Singleton-type bound is an upper bound.

Theorem 31 (Singleton-type bound). *Consider n abstract alphabets $Q_1, \ldots, Q_n$ of n-codewords $\boldsymbol{c} = (c_1, \ldots, c_n)$ where $c_i \in Q_i$. Assume w.l.o.g. that the alphabets are ordered according to*

$$q_1 \leq \cdots \leq q_n.$$

where $q_i = |Q_i|$ for $i = 1, \ldots, n$. The cardinality of a polyalphabetic code $\mathcal{C}$ with distance d is upper bounded by

$$|\mathcal{C}| \leq \prod_{i=1}^{n-d+1} q_i. \tag{6.4}$$

Proof: Consider the code $\mathcal{C}'$ obtained by puncturing the code $\mathcal{C}$ in the last $d-1$ positions. The Hamming distance of the code $\mathcal{C}'$ is at least 1, since puncturing one position decreases the code distance by one at most. It means that codewords of the code $\mathcal{C}'$ of length $n' = n - (d-1)$ should be pairwise different and hence (6.4) holds, since $F(n')$ is the number of different words of length n' over alphabets of sizes $p_1, \ldots, p_{n'}$. □

Theorem 32. *A CR code $\mathcal{CR}(\mathcal{P}; n, K)$ is maximum distance separable (MDS), i.e., its minimum Hamming distance is $d = n - k + 1$.*

Proof: The Singleton-type bound for the CR code can be written as $|\mathcal{C}| = F(1,k) \leq F(1, n-d+1)$ from which it follows that $k \leq n - d + 1$.

For the CR code, two codewords (with corresponding information integers C_1 and C_2) agree at i-th coordinate if and only if $[C_1 - C_2]_{p_i} = 0$, i.e., C_1 and C_2 are said to be *congruent modulo* p_i. But $|C_1 - C_2| < k$ can have at most $(k-1)$ prime factors from the list $\mathcal{P}$ which means that at most $(k-1)$ coordinates of the two codewords are the same. So, the minimum distance of the CR code $d \geq n - k + 1$. From this and the Singleton-type bound, the statement of the theorem follows. □

Assume we have settled on a CR code $\mathcal{CR}(n, K)$, and we transmit some codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{Z}_{\mathcal{P}}$ over an additive noisy channel and receive the word $\boldsymbol{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_{\mathcal{P}}$. Then we say that the error vector is $\boldsymbol{e} = (e_1, \ldots, e_n) \in \mathbb{Z}_{\mathcal{P}}$ in the channel with $e_i = [r_i - c_i]_{p_i} \in \mathbb{Z}_{p_i}$. Let t be the number of errors, i.e., the Hamming weight of $\boldsymbol{e}$. By the Chinese remainder theorem, if the receiver knows any k error free positions of $\boldsymbol{r}$, then it can reconstruct C.

Theorem 33 (Chinese remainder theorem (CRT)). *If $a_1, a_2, \ldots, a_m$ is a sequence of integers and $0 \leq a_i < p_i$ for $0 < i \leq m$ where $p_1, \ldots, p_m$ are relatively prime positive integers, then there exists a unique nonnegative integer $X < N = F(1,m)$ such that*

$$[X]_{p_1} = a_1, [X]_{p_2} = a_2, \ldots, [X]_{p_m} = a_m. \tag{6.5}$$

The solution of (6.5) *is*

$$X = \sum_{i=1}^{m} a_i b_i \frac{N}{p_i} \pmod{N} \tag{6.6}$$

where b_i are determined from

$$b_i \frac{N}{p_i} \equiv 1 \mod p_i.$$

The CRT allows us to solve a system of m congruence equations (6.5). The complexity of the CRT was analyzed in [GG03]. Denote the word length of a nonzero integer by $\mathrm{len}(\cdot)$, i.e., the number of digits for some base. Assume that we have a 64-bit processor, the length of N is

$$\mathrm{len}(N) = \lfloor \log_{2^{64}} |N| \rfloor + 1 = \lfloor \log_2 |N|/64 \rfloor + 1. \tag{6.7}$$

For the complexity of the Chinese remainder theorem, we refer [GG03, Theorem 5.8] which is stated as follows.

Theorem 34. *Let $p_1, \ldots, p_m$, $a_1, \ldots, a_m$, N as defined in Theorem 33, and len(N) as defined in* (6.7)*. Then the unique solution $X \in \mathbb{Z}$ with $0 \leq X < N$ of the Chinese remainder theorem can be computed using $\mathcal{O}(len(N)^2)$ word operations.*

Furthermore, we refer [GG03, Theorem 10.25] which states that the Chinese remainder theorem can be accelerated with $\mathcal{O}(M(\log N) \log\log N)$ word operations where $M(n)$ is the multiplication time of two integers of length at most n. For Karatsuba's algorithm [KO63], $M(n)$ is $\mathcal{O}(n^{\log_2 3})$, and for Schönhage–Strassen's algorithm [SS71], $M(n)$ is $\mathcal{O}(n \log n \log\log n)$.

A common decoding strategy, which we will use throughout this dissertation, is firstly to identify the erroneous positions in the received word $\boldsymbol{r}$ and secondly the codeword can be calculated from at least k error free positions.

Since each component has different alphabet size p_i, in addition to the usual Hamming distance, let us define the weighted Hamming distance between words $\boldsymbol{r}$ and $\boldsymbol{c}$ as follows

$$d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) = \sum_{i: r_i \neq c_i} \log p_i. \tag{6.8}$$

Using the Chinese remainder theorem, we can compute the received word in frequency domain $R < N$ such that $[R]_{p_i} = r_i$, and likewise an $E < N$ such that $[E]_{p_i} = e_i$; then $R \equiv C + E \mod N$. We will find the positions of the errors by determining the *error locator* Λ, defined as:

$$\Lambda = \prod_{i: r_i \neq c_i} p_i. \tag{6.9}$$

Thus $d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) = \log \Lambda$.

According to the definition of Λ, if we encode Λ to a vector $\boldsymbol{\lambda}$, then the coordinates λ_i where errors occur are zeros, i.e., $\lambda_i = 0$ for $c_i \neq r_i, \forall i = 1, \ldots, n$. Thus, we have the following lemma:

Lemma 35. *The product of the error locator and the error value is a multiple of N, i.e.,*

$$\Lambda E \equiv 0 \mod N. \tag{6.10}$$

Proof: Let

$$\Gamma = \prod_{i: c_i = r_i} p_i.$$

The error word $\boldsymbol{e} = (e_1, e_2, \ldots, e_n)$ has zero coordinates at error free positions, that is $\Gamma | E$. Since $\Lambda\Gamma = N$ and E is a multiple of Γ, it is straightforward to see that $N | (\Lambda E)$ i.e., $[\Lambda E]_{p_i} = 0 \ \forall i$ and the statement follows. □

The CR code indicates a one-to-one mapping between a number $0 < C < K$ and a code vector $\boldsymbol{c}$. We denote this mapping by $C \bullet\!\!-\!\!\circ\ \boldsymbol{c}$, and $\boldsymbol{c} \circ\!\!-\!\!\bullet\ C$, which are similar to Fourier transform for RS codes (c.f. Definition 4).

Theorem 36 (Convolution property). *Given two integers A and B, two vectors $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, \ldots, b_n)$, and a prime list $\{p_1, \ldots, p_n\}$ where $a_i = [A]_{p_i}$ and $b_i = [B]_{p_i}$ for all $i = 1, \ldots, n$, the component-wise multiplication of $\boldsymbol{a}$ and $\boldsymbol{b}$ modulo p_i corresponds to the product of A and B modulo $N = \prod_{i=1}^{n} p_i$ (and vise versa), i.e., if*

$$\boldsymbol{a} \circ\!\!-\!\!\bullet\ A, \qquad \boldsymbol{b} \circ\!\!-\!\!\bullet\ B,$$

then

$$c_i = a_i b_i \mod p_i \quad \Leftrightarrow \quad \boldsymbol{c} \circ\!\!-\!\!\bullet\ C = AB \mod N.$$

Proof: In Theorem 33, for all i, the coefficients b_i are independent of the integer X. Denote b_i in (6.6) by $[N/p_i]_{p_i}^{-1}$.

Then by CRT,

$$\begin{aligned} C &= \sum_{i=1}^{n} c_i \left[\frac{N}{p_i}\right]_{p_i}^{-1} \frac{N}{p_i} \mod N = \sum_{i=1}^{n} [a_i b_i]_{p_i} \left[\frac{N}{p_i}\right]_{p_i}^{-1} \frac{N}{p_i} \mod N \\ &= \sum_{i=1}^{n} [[A]_{p_i}[B]_{p_i}]_{p_i} \left[\frac{N}{p_i}\right]_{p_i}^{-1} \frac{N}{p_i} \mod N \\ &= \sum_{i=1}^{n} [AB]_{p_i} \left[\frac{N}{p_i}\right]_{p_i}^{-1} \frac{N}{p_i} \mod N \\ &= AB \mod N, \end{aligned}$$

and the statement follows. □

One can compare Chinese remainder codes with Reed–Solomon codes in a perspective of the Fourier transform and it's convolution property. For the RS codes, we define an error locater polynomial $\Lambda(x)$ which has roots at all erroneous positions, whereas all the prime factors from $\mathcal{P}$ of the error locator Λ for the Chinese remainder codes indicate error positions.

Lemma 37. *The product of the error locator and $[E]_K$ is a multiple of K:*

$$\Lambda \cdot [E]_K \equiv 0 \mod K. \tag{6.11}$$

Proof: The integer $[E]_K$ can be written as $[E]_K = E - mK$ where m is some integer. Therefore, for $i = 1, \ldots, k$,

$$[E - mK]_{p_i} = [[E]_{p_i} - [mK]_{p_i}]_{p_i} = [E]_{p_i}. \tag{6.12}$$

For $i = k+1, \ldots, n$, we can not guarantee that (6.12) holds. In the vector form, $\boldsymbol{e}$ and $\boldsymbol{e}_K (:= \circ\!\!-\!\!\bullet\ [E]_K)$ have the same error position(s) in the first k coordinates. Therefore, the vector form of $\Lambda \cdot [E]_K$ has all zero in the first k positions, and (6.11) holds by Theorem 36. □

Adding ΛC to both sides of (6.10) in Lemma 35, it immediately leads to a *key equation*:

$$\Lambda R \equiv \Lambda C \mod N. \tag{6.13}$$

For a not too large number of errors, then $\Lambda R \gg N$ while $\Lambda C \ll N$, and Λ turns out to be the only relatively small number such that $[\Lambda R]_N$ is also small.

In [GRS00] an error correcting GRS algorithm was suggested for the classical Chinese remainder codes. The GRS decoder is shown by Algorithm 19, where the logarithm of the integer parameter D is the error correcting radius in the weighted Hamming metric of the GRS algorithm.

Algorithm 19: The GRS decoder for Chinese remainder codes

1 **Input**: The list $\mathcal{P}$, the received word $(r_1, \ldots, r_n)$, N, K, D

2 **begin**

3 Using the CRT compute $0 \leq R < N$ such that $r_i = [R]_{p_i}$.

4 Find integers Λ, Ω such that

$$\begin{array}{l} 1 \leq \Lambda \leq D \\ 0 \leq \Omega < N/D \\ \Lambda R \equiv \Omega \mod N \end{array} .$$

5 Output Ω/Λ if it is an integer.

6 **Output**: The message C

The cost of performing Algorithm 19 is as follows. Using the fast algorithm to compute the CRT with Schönhage–Strassen's algorithm, the integer R in Line 3 is computed in time complexity $\mathcal{O}(\mathrm{len}(N) \log^2 \mathrm{len}(N) \log\log \mathrm{len}(N))$. For the realization of Line 4, one can use fast Euclidean algorithm for integers [GG03, Chapter 11], or equivalently, continued fractions method [Len83]. Both algorithms have the same complexity which is $\mathcal{O}(\mathrm{len}(N) \log^\epsilon \mathrm{len}(N))$ for some constant ϵ [GRS00, Theorem 17], which means that this step performs in nearly linear time in bit size of N. Therefore, the overall complexity of Algorithm is determined by Line 4 which is $\mathcal{O}(\mathrm{len}(N) \log^\epsilon \mathrm{len}(N))$.

Lemma 38 ([GRS00], Lemma 5). *If* $d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) = \log D \leq \log(\sqrt{N/(K-1)})$ *then the decoding Algorithm 19 finds* Λ *using* (6.13).

Define $D_t = F(n-t+1, n)$ as the maximal value of Λ given that at most t errors have occurred. The decoder of Lemma 38 succeeds whenever $\log D_t \leq \log(\sqrt{N/K}) < \log(\sqrt{N/(K-1)})$. We can relax this to a decoding radius in the Hamming metric. Using $D_t < p_n^t$ we have

$$p_n^t \leq \sqrt{N/K},$$

thus

$$t \leq \left\lfloor \frac{1}{2} \cdot \frac{\log(N/K)}{\log p_n} \right\rfloor . \tag{6.14}$$

There is a different expression of the decoding radius in [GRS00]

$$t \leq \left\lfloor \frac{\log p_1}{\log p_1 + \log p_n}(d-1) \right\rfloor \tag{6.15}$$

which shows a direct connection to the code distance. In most cases, (6.14) and (6.15) give the same result. We will discuss (6.15) in details in Theorem 43. If p_1 and p_n do not differ too much, then up to half the minimum distance errors can be corrected by the GRS decoder.

6.2 Syndrome Decoding

Equipped with the results mentioned in the previous section, we now define the syndrome of the CR codes and later on the key equation to decode CR codes.

We define the syndrome S of a received word $\boldsymbol{r}$ with corresponding R as follows:

$$S = \frac{R - [R]_K}{K}. \tag{6.16}$$

There are some remarks regarding to the syndrome definition.

1. The syndrome of a codeword $\boldsymbol{c}$ is zero, since $C - [C]_K = 0$.

2. The syndrome is an integer and depends on the error word E and essentially does not depend on the codeword. Assume that the error word E is known, we shall discuss the value of the syndrome in two cases as follows

$$S = \frac{R - [R]_K}{K} = \begin{cases} \frac{E-[E]_K}{K} & \text{if } 0 \leq [E]_K < K - C; \\ \frac{E-[E]_K+K}{K} & \text{otherwise.} \end{cases}$$

   We denote it as $S = (E - [E]_K + \delta_K(C,E)K)/K$ where

$$\delta_K(C,E) = \begin{cases} 0 & \text{if } 0 \leq [E]_K < K - C; \\ 1 & \text{otherwise.} \end{cases}$$

   Although $\delta_K(C,E)$ is a function related to the codeword C, and makes the syndrome have two values, later it can be seen that the analysis of the two cases of S come up to the same solution.

The multiplication of Λ and the syndrome S can be written as

$$\Lambda \cdot S = \Lambda \left(\frac{E - [E]_K + \delta_K(C,E)K}{K} \right).$$

With Lemma 35 and Lemma 37, we obtain

$$\Lambda \cdot S = \frac{iN - jK + \delta_K(C,E)\Lambda K}{K} = i\frac{N}{K} - j + \delta_K(C,E)\Lambda. \tag{6.17}$$

where $i \triangleq \Lambda E/N$ and $j \triangleq \Lambda[E]_K/K$ are some nonzero integers. We know from (6.11) that $1 \leq j < \Lambda$. Let $\Omega = -j + \delta_K(C,E)\Lambda$, then there are two cases we have to consider:

1. $\Omega = -j < 0$. Since $\Omega + \Lambda = -j + \Lambda > 0$, we obtain $-\Lambda < \Omega < 0$.
2. $\Omega = -j + \Lambda > 0$. Furthermore, $\Omega - \Lambda = -j < 0$, hence, $0 < \Omega < \Lambda$.

In both cases, the absolute value of Ω should be smaller than Λ. Note that, if $\Omega = 0$, then the received word is error free.

Using $\Lambda R = \Lambda C + \Lambda E = \Lambda C \mod N$, one can find the decoding radius is $\Lambda < \sqrt{N/(K-1)}$. Rewrite (6.17) as

$$\Lambda \cdot S \equiv \Omega \mod \frac{N}{K} \quad \text{with } |\Omega| < \Lambda < \sqrt{\frac{N}{K-1}}. \tag{6.18}$$

As a result, we have the syndrome-based *key equation* (6.18). Given S, N and K, one can solve the key equation and obtain Λ, using the following Algorithm 20.

Algorithm 20: Syndrome-based decoder of Chinese remainder codes

1 **Input**: Syndrome S calculated by (6.16), N, K

2 **begin**

3 Solve $\Lambda \cdot S \equiv \Omega \mod N/K$ by extended Euclidean algorithm iteratively to find the greatest common divisor of S and N/K, which is $\Lambda_i S + \Psi_i(N/K) = \Omega_i$.

4 Stop when $|\Lambda_i| < \Omega_i$ and $|\Lambda_{i+1}| > \Omega_{i+1}$.

5 $\Lambda \leftarrow |\Lambda_{i+1}|$.

6 **Output**: Error locator Λ

The inspiration of the algorithm comes from decoding RS codes by the key equation. The proposed decoder finds the error locator Λ given parameters (N, K) of the code and the syndrome S. Up to $(n-k)\log p_1/(\log p_1 + \log p_n)$ errors can be always corrected, since the decoding radius is the same as the one for the GRS decoder. In contrast to [GRS00] which finds the codeword directly, our decoder finds the number of errors and their positions.

It is easy to see that the complexity of the step in Line 3 determines the whole complexity of Algorithm 20. As is known in [GG03, Corollary 11.13], the extended Euclidean algorithm has nearly linear complexity in bit size of N/K ($\mathcal{O}(M(\text{len}(N/K))\log \text{len}(N/K))$). The difference from Algorithm 19 is that, firstly we aim to find the error locations, the message can be computed from any k correct positions by the Chinese remainder theorem, secondly the parameter D is no longer needed here, and thirdly the numbers (S and N/K) we start to deal with are K times smaller than those (R and N) in Algorithm 19.

Let us continue with Example 5.

Example 6 (Continued). *Assume the received word is* $\boldsymbol{r} = (1, 4, 0, 3, 1)$*. By CRT, the corresponding* $R = 10024$*. The syndrome* $S = 668$ *according to* (6.16)*.* $N/K = 1001$ *is known at the receiver. Table 6.1 illustrates the computation of extended Euclidean algorithm.*

i	Ω_i	Λ_i	Ψ_i
0	1001	0	1
1	668	1	0
2	333	-1	1
3	2	3	-2

Table 6.1: Intermediate results of decoding the Chinese remainder code in Example 6.

We read from $i = 2$ *and* $i = 3$ *that* $|\Lambda_2| < \Omega_2$ *and* $|\Lambda_3| > \Omega_3$*. The error locator is* 3*, i.e., the first position is erroneous. The message* C *can be calculated by CRT from any* $k = 2$ *of the correct positions.*

To find the error positions, the error locator should to be factorized using factors from $\mathcal{P}$. Usually integer factorization is an $\mathcal{NP}$-complete problem. Since all the prime factors are known at the receiver, one can employ Chien like search for Λ, i.e., check every element in $\mathcal{P}$ to see if it is a factor of Λ.

Algorithm 20 has the same decoding performance as the one in (6.14).

Note that, at the same time we obtain the codeword, the error locator is also provided. If we only focus on finding the error positions, Algorithm 20 gives a syndrome-based decoder based on the syndromes which is equivalent to this unique decoder. Moreover, the form of the syndrome-based decoder is similar to solving the classical key equation to decode the Reed–Solomon codes.

6.3 Error and Erasure Decoding

Up to now, all the decoders for the CR codes correct errors only. In this section, in order to correct erasures, we first give a slightly more general than the usual definition of the Chinese remainder codes and then propose for these codes a decoding algorithm correcting errors and erasures. The proposed decoder is based on the decoder by Goldreich et.al. in [GRS00] and has similar complexity. The decoding capability of the decoder is given by Theorem 44.

The section is organized as follows. First, a generalized Chinese remainder codes is given, then we introduce the punctured Chinese remainder code which forms a subcode of the classical codes. After demonstrating GRS decoder for generalized CR codes in Section 6.3.2, our error-erasure-decoder will be shown in Algorithm 21.

6.3.1 Generalized CR Codes

To be able to correct both errors and erasures we need the following slightly more general than the usual definition of Chinese remainder codes. Let us explain why the classical definition of the Chinese remainder codes (see Definition 16) is not enough for the case of error and erasure correction. Let us use a classical $\mathcal{CR}(\mathcal{P}; n, F(k))$ code. Assume we have τ erasures in a received word of the code. Then we should correct errors in the following punctured code. Denote the list of $n-\tau$ unerased positions by $\mathcal{U} = \{u_1, u_2, \ldots, u_{n-\tau}\}$, and by $\mathcal{P}_\mathcal{U}$ denote the list of p_i at unerased positions, i.e., $\mathcal{P}_\mathcal{U} = \{p_i : \ i \in \mathcal{U}\}$. By puncturing the classical code $\mathcal{CR}(\mathcal{P}; n, F(k))$ in the τ erased positions we obtain the code $\mathcal{CR}(\mathcal{P}_\mathcal{U}; n-\tau, F(k))$, which is not classical anymore if at least one of the first k positions of the original code was punctured. Indeed, if for example the first position is punctured, then to be classical the cardinality $K = p_1 p_2 \cdots p_k$ of the punctured code should be equal to $p_2 p_3 \cdots p_i$ for some i, which is not possible since p_j are co-prime.

Our plan is as follows. We will be able to correct errors and erasures if we can correct errors only in the punctured original code, which is not the classical code any more. So we extend the error-only-correcting GRS decoder in Algorithm 19 to our wider class of the Chinese remainder codes and apply this algorithm for the punctured code. After successful decoding the punctured code, we obtain correct symbols at unerased positions and can reconstruct the sent message of the original code. Before doing this, let us give some properties of the wider class of the Chinese remainder codes given by Definition 17.

Definition 17 (Generalized Chinese remainder code). *A Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ or shortly $\mathcal{CR}(n, K)$ having cardinality $0 \leq K \leq N$ and length n over alphabets $\mathcal{P}$ is defined as follows*

$$\mathcal{CR}(\mathcal{P}; n, K) = \{ \ ([C]_{p_1}, \ldots, [C]_{p_n}) : C \in \mathbb{N} \text{ and } C < K \} .$$

A generalized Chinese remainder code is also said to be an *arbitrary* Chinese remainder code. If cardinality of the code satisfies

$$K = F(k)$$

for some k then the code is a classical Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, F(k))$.

Corollary 39 ([SSG$^+$05]). *For cardinality $|\mathcal{C}|$ of any polyalphabetic code $\mathcal{C}$ of length n over alphabets of sizes $p_1, p_2, \ldots, p_n$ satisfying (6.1) having distance d in the Hamming metric holds the following Singleton-type upper bound*

$$|\mathcal{C}| \leq \prod_{i=1}^{n-d+1} p_i = F(n-d+1). \tag{6.19}$$

Notice that we do not require in Corollary 39 that p_i are relatively prime. It is also interesting that the bound (6.19) does not depend on the $d-1$ largest alphabet sizes! The

classical code $\mathcal{CR}(n, F(k))$ has code distance $d = n-k+1$, which satisfy the Singleton-type upper bound (6.19) with equality.

For the function $F(k)$ let us define an inverse function $f(K)$ as follows. For every integer $0 < K \leq N$, the function $f(K)$, takes the integer value $0 < f(K) \leq n$ that satisfies

$$F(\ f(K)-1\ ) < K \leq F(\ f(K)\ ), \tag{6.20}$$

and $f(0) = 0$.

It follows from Definition 17 that $f\left(F(k)\right) = k$ for every $0 \leq k \leq n$. In the classical case $K = F(k)$, and the code has $k = f(K)$ first information positions.

Lemma 40. *Every code $\mathcal{CR}(\mathcal{P}; n, K)$ is a subcode of the classical code $\mathcal{CR}(\mathcal{P}; n, F(f(K)))$ and a supercode for the classical code $\mathcal{CR}(\mathcal{P}; n, F(f(K)-1))$, i.e.,*

$$\mathcal{CR}(n, F(f(K)-1)) \subset \mathcal{CR}(n, K) \subseteq \mathcal{CR}(n, F(f(K))). \tag{6.21}$$

Proof: All codewords of the code $\mathcal{CR}(n, K)$ are obtained by encoding the messages C, where $0 \leq C \leq K-1$, using the list $\mathcal{P}$. Hence, the statement of the lemma follows from (6.20). □

From Lemmas 39, 40 and from Theorem 33 we have the followinig properties of the generalized CR codes.

Theorem 41. *The Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ has minimum distance $d = n - f(K)+1$ in the Hamming metric. Every $f(K)$ positions of a codeword form a reconstruction set, i.e., by knowing $f(K)$ positions, the codeword can be uniquely reconstructed using the CRT. For a classical Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, F(k))$ the minimum distance is $d = n-k+1$, and every k positions of a codeword form a reconstruction set.*

Proof: The second part of the theorem about classical Chinese remainder codes was proved in [GRS00].

According to Lemma 40 the code $\mathcal{C} = \mathcal{CR}(\mathcal{P}; n, K)$ is a subcode of the classical code $\mathcal{CR}(\mathcal{P}; n, F(f(K)))$ with distance $n - f(K) + 1$. Hence for the distance d of the code $\mathcal{C}$ we have $d \geq n - f(K) + 1$. Let us show that only equality is possible here. Indeed, if inequality holds and say $d = n - f(K) + 2$, then $n - d + 1 = f(K) - 1$, and from the upper bound (6.19) $K = |\mathcal{C}| \leq F(f(K)-1)$ that contradicts to (6.20).

Since for the classical code $\mathcal{CR}(\mathcal{P}; n, F(f(K)))$ every $f(K)$ codeword's positions form a reconstruction set, the same holds for the subcode $\mathcal{C}$. □

6.3.2 Error Correction

In [GRS00] an error correcting GRS algorithm was suggested for the classical Chinese remainder codes. In this section we show that this algorithm can be extended for correcting errors for an arbitrary Chinese remainder code.

Assume a codeword $\boldsymbol{c} = (c_1, \ldots, c_n)$ of an arbitrary Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ is transmitted, and a word $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ from the code space $\mathbb{Z}_{\mathcal{P}}$ is received. We assume t errors in the received word, i.e., $\boldsymbol{r}$ is obtained from $\boldsymbol{c}$ by replacing t components.

Let us recall the weighted Hamming distance between words $\boldsymbol{r}$ and $\boldsymbol{c}$ and error locator as follows

$$d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) = \sum_{i: r_i \neq c_i} \log p_i = \log \Lambda.$$

The GRS decoder for an arbitrary Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ is shown by Algorithm 19, where logarithm of the integer parameter D is the error correcting radius in the weighted metric of the GRS algorithm.

Lemma 42 ([GRS00]). *If $\boldsymbol{r}$ is such that for some $C \in \mathbb{Z}_K$ holds $d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) \leq \log D$, where $D < \sqrt{N/(K-1)}$, then Algorithm 19 returns C.*

Proof: The lemma was proved in [GRS00] for the case of classical Chinese remainder codes, i.e., when the condition $K = F(k)$ holds for some k. However, this condition was not used in the proof in [GRS00]. Hence the proof in [GRS00] holds for our case of general Chinese remainder codes as well. □

Lemma 42 gives decoding radius of Algorithm 19 in the weighted Hamming metric. The following theorem answers the question how many errors can be corrected by the algorithm. But first we need one more definition. Given the list $\mathcal{P}$, we define the geometric mean $p(m)$ of m numbers p_i, $m \leq n$, as follows

$$p(m) = \sqrt[m]{p_n p_{n-1} \cdots p_{n-m+1}}.$$

Theorem 43. *Given a code $\mathcal{CR}(\mathcal{P}; n, K)$ with minimum distance d in the Hamming metric, any pattern of t errors will be corrected by Algorithm 19 with $D = \sqrt{N/K}$, provided that $t \leq t_{max}$, where the decoding radius is*

$$t_{\max} = \max\left\{ t \in \mathbb{N} : \; t \log p(t) \leq \log \sqrt{N/K} \right\}. \tag{6.22}$$

The decoding radius can be approximated by the following lower bound, which shows dependance of the radius from the code distance:

$$t_{\max} \geq \left\lfloor (d-1) \frac{\log p_{f(K)+1}}{\log p_{f(K)+1} + \log p_n} \right\rfloor. \tag{6.23}$$

Proof: The parameter $D = \sqrt{N/K}$ satisfies $D < \sqrt{N/(K-1)}$, hence due to Lemma 42 the received word $\boldsymbol{r}$ will be decoded correctly to the codeword $\boldsymbol{c}$ if $d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) \leq \log D$. If t is the Hamming distance $t = d(\boldsymbol{r}, \boldsymbol{c})$, then the weighted distance $d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r})$ is maximum when the t errors occur at the last components and

$$\max_{\boldsymbol{c} \in \mathcal{C}} d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) = \log \prod_{i=n-t+1}^{n} p_i = t \log p(t).$$

Hence t errors will be always corrected if $t \log p(t) \leq \log D$ and (6.22) follows, which gives the precise value for the error correcting radius of Algorithm 21.

Let us prove (6.23). It follows from (6.20) that $K \leq F(f(K))$, hence

$$\max d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) \leq \log \sqrt{N/K}$$

if

$$\max d_{\mathcal{P}}(\boldsymbol{c}, \boldsymbol{r}) \leq \log \sqrt{N/F(f(K))}.$$

The last is equivalent to

$$\left(\prod_{i=n-t+1}^{n} p_i \right)^2 \leq \prod_{i=f(K)+1}^{n} p_i,$$

which gives

$$\prod_{i=n-t+1}^{n} p_i \leq \prod_{i=f(K)+1}^{n-t} p_i.$$

This holds if

$$p_n^t \leq p_{f(K)+1}^{n-f(K)-t} \tag{6.24}$$

and (6.23) follows since $n - f(K) = d - 1$ by Theorem 41. □

Notice that in (6.24) we can replace $p_{f(K)+1}$ by a smaller value p_1 and get a less precise bound in Theorem 43 like in [GRS00]

$$t_{\max} \geq \left\lfloor (d-1) \frac{\log p_1}{\log p_1 + \log p_n} \right\rfloor. \tag{6.25}$$

Up to now, we obtain two decoding radii (6.14) and (6.25) by different approximation. Two radii might differ, given the same parameters of the code. From (6.25), the connection of decoding radius and minimum distance is seen more clearly. When all entries in $\mathcal{P}$ are in same order of magnitude, one can correct errors up to half the minimum distance.

6.3.3 Error and Erasure Correction

We are ready to present the error and erasure decoding algorithm. Still assume a codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ of an arbitrary Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ was transmitted, and a word $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) \in \mathbb{Z}_{\mathcal{P}}$ was received. The receiver knows the list $\mathcal{U}$ of $n - \tau$ unerased positions, i.e., the rest τ positions can be considered as erased (unknown). In addition we assume that the received word has t errors in unerased positions.

The decoder for an arbitrary Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ correcting errors and erasures is shown by Algorithm 21. We give the error correction radius for this error and erasure decoder in Theorem 44. For simplicity we give the theorem using the bound (6.25), despite it can be done for the expressions (6.14), (6.22) and (6.23) as well.

Algorithm 21: Correcting errors and erasures for Chinese remainder codes

1 **Input**: The list $\mathcal{P}$, the received word $\boldsymbol{r}$, unerased set $\mathcal{U}$

2 **begin**

3 By puncturing erased positions in $\boldsymbol{r}$ get punctured received word $\boldsymbol{r}'$.

4 Correct errors in $\boldsymbol{r}'$ by Algorithm 19 for the punctured code $\mathcal{CR}(\mathcal{P}_{\mathcal{U}}; n-\tau, K)$, get a codeword $\boldsymbol{c}' \in \mathcal{CR}(\mathcal{P}_{\mathcal{U}}; n-\tau, K)$.

5 The codeword $\boldsymbol{c}' \in \mathcal{CR}(\mathcal{P}_{\mathcal{U}}; n-\tau, K)$ gives $n-\tau$ correct symbols of the sent codeword $\boldsymbol{c}$. Reconstruct the message C by applying the CRT to the known symbols of $\boldsymbol{c}$.

6 **Output**: The message C

Theorem 44. *Given a code $\mathcal{CR}(\mathcal{P}; n, K)$ with minimum distance d, any pattern of t errors and τ erasures will be corrected by Algorithm 21, provided that*

$$(\log p_1 + \log p_n)/\log p_1 t + \tau \leq d - 1. \tag{6.26}$$

Proof: The function $f(K)$ depends on the list $\mathcal{P}$ and we show this dependence by $f_{\mathcal{P}}(K)$. Notice that $f_{\mathcal{P}}(K) \geq f_{\mathcal{P}_{\mathcal{U}}}(K)$.

According to Theorem 41, the original code $\mathcal{C} = \mathcal{CR}(\mathcal{P}; n, K)$ has the code distance $d = n - f_{\mathcal{P}}(K) + 1$, and the punctured code $\mathcal{C}_p = \mathcal{CR}(\mathcal{P}_{\mathcal{U}}; n-\tau, K)$ has distance

$$d_p = n - \tau - f_{\mathcal{P}_{\mathcal{U}}}(K) + 1 \geq d - \tau.$$

If (6.26) is satisfied then the bound (6.25) of Theorem 43 holds for $\mathcal{C}_p$, i.e.,

$$\frac{\log p_1 + \log p_n}{\log p_1} t \leq d_p - 1. \tag{6.27}$$

Hence, t errors will be corrected in Step 2 of Algorithm 21 due to Theorem 43 and we obtain correct codeword $\boldsymbol{c}'$ of the punctured code of length $n-\tau$.

This codeword $\boldsymbol{c}'$ gives $n-\tau$ correct symbols of the sent codeword $\mathbf{c}$. If (6.26) is satisfied then $n - \tau \geq f_{\mathcal{P}}(K)$. This allows us to reconstruct the message C by applying the CRT to the known symbols of $\mathbf{c}$, since $f_{\mathcal{P}}(K)$ symbols form a reconstruction set according to Theorem 41. □

Since the complexity of the Algorithm 21 mainly depends on the Line 4 and in [GRS00] Goldreich et.al. showed that Algorithm 19 has a nearly linear time complexity in the bit sizes of N , we can assume the same complexity also applies for the Algorithm 21.

Thus, in this section, we gave a more general than the usual definition for the Chinese remainder codes and proposed punctured Chinese remainder codes afterwards. To correct a received word with t errors and τ erasures satisfying (6.26), we introduced an error-erasure-decoder which is an extension of the GRS decoder. We also showed the error correcting radius and analyzed the complexity for our decoder.

6.4 Decoding Interleaved CR Codes

Now we consider interleaved Chinese remainder (ICR) codes. As is mentioned at the beginning of this chapter, the ICR codes are algebraic similar to interleaved Reed–Solomon (IRS) codes in many respects. Recently, the construction of IRS codes have been intensively studied in several publications, e.g. [SSB09] where decoding beyond half the minimum distance is proposed. Among these decoding algorithms for IRS codes, Nielsen [Nie13b] proposed a module minimization approach for solving multiple key equations by finding short vectors in a certain space.

In this section, we adapt module minimization approach in [Nie13b] to decode ICR codes, i.e., we model the decoding of ICR codes as that of finding a short vector in a $\mathbb{Z}$-lattice. Using the Lenstra–Lenstra–Lovász (LLL) algorithm, we obtain an efficient decoding algorithm, correcting errors beyond half the minimum distance and having nearly linear complexity. The algorithm can fail with a probability dependent on the number of errors, and we give an upper bound for the failure probability. Simulation results indicate that the bound is close to the truth. We apply the proposed decoding algorithm for decoding a single CR code using the idea of "power" decoding, suggested for Reed–Solomon codes. A combination of these two methods can be used to decode low-rate ICR codes.

In Section 6.4.1, we introduce ICR codes and state the decoding problem. In Section 6.4.3 we give the decoder for ICR codes as well as theoretical considerations, such as complexity, failure probability and decoding radius. Some simulation results are shown; in Sections 6.4.4 and 6.4.5, we discuss how this method can be extended using power decoding for single and interleaved CR codes.

6.4.1 Interleaving of CR Codes

Interleaving is a technique for making long codes from shorter ones which efficiently handle burst errors. Up to now, we investigated interleaved Reed–Solomon codes and interleaved Gabidulin codes. We also improved the corresponding collaborative decoding approaches. Since the CR codes are similar to RS codes and Gabidulin codes, it is inevitable to consider interleaved CR codes for improving their decoding performance.

Definition 18 (Interleaved Chinese remainder code). *Let us consider ℓ classical CR codes $\mathcal{CR}(\mathcal{P}; n, K_l), l \in 1, \ldots, \ell$. Denote the list $K_1, K_2, \ldots, K_\ell$ by $\mathcal{K}$. The Interleaved Chinese Remainder code $\mathcal{ICR}(\mathcal{P}; n, \mathcal{K})$ or shortly $\mathcal{ICR}(n, \mathcal{K})$ is defined as the set of matrices*

$$\begin{pmatrix} c_1^{(1)} & c_2^{(1)} & \ldots & c_n^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \ldots & c_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(\ell)} & c_2^{(\ell)} & \ldots & c_n^{(\ell)} \end{pmatrix}$$

where $\boldsymbol{c}^{(l)} = (c_1^{(l)}, \ldots, c_n^{(l)}) \in \mathcal{CR}(n, K_l), l = 1, \ldots, \ell$.

For the remainder of this section, consider some received matrix with rows $\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(\ell)}$ where $\boldsymbol{r}^{(l)} = \boldsymbol{c}^{(l)} + \boldsymbol{e}^{(l)}$ for some error row $\boldsymbol{e}^{(l)}$. We now define a *complete* error locator which identifies all columns having any errors, i.e.,

$$\Lambda = \prod_{i:\exists l: r_i^{(l)} \neq c_i^{(l)}} p_i.$$

We only consider burst errors, i.e., the error positions are the same for all rows. When we refer to "the number of errors", it is also the number of factors in the above product, and the number of erroneous columns in the received matrix.

For each $\boldsymbol{c}^{(l)}$ and $\boldsymbol{r}^{(l)}$ corresponds a $C^{(l)}$ and an $R^{(l)}$, respectively. For a particular row l, the key equation (6.13) holds with Λ; thus, collaboratively decoding of the ICR codes becomes solving a system of ℓ key equations as follows

$$\begin{cases} \Lambda R^{(1)} & \equiv & \Lambda C^{(1)} \mod N \\ \Lambda R^{(2)} & \equiv & \Lambda C^{(2)} \mod N \\ & \vdots & \\ \Lambda R^{(\ell)} & \equiv & \Lambda C^{(\ell)} \mod N \end{cases} \tag{6.28}$$

where Λ is a positive integer and relatively small.

6.4.2 Lattice Reduction

The problem of solving the system of key equations (6.28) is in abeyance until we know another problem — finding a short vector in a lattice. We refer this section to [[GG03], Chapter 16].

Definition 19 (Lattice). *Let $n \in \mathbb{N}$ and $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{R}^n$ with $\boldsymbol{f}_i = (f_{i1}, \ldots, f_{in})$. Then*

$$\mathcal{L} = \sum_{1 \leq i \leq n} \mathbb{Z}\boldsymbol{f}_i = \left\{ \sum_{1 \leq i \leq n} r_i \boldsymbol{f}_i : r_1, \ldots, r_n \in \mathbb{Z} \right\}$$

is the lattice or $\mathbb{Z}$-module generated by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$. If these vectors are linearly independent, they are a basis *of $\mathcal{L}$.*

The *norm of $\mathcal{L}$* is $|\mathcal{L}| = |\det(f_{ij})_{1 \leq i,j \leq n}| \in \mathbb{R}$. The norm does not depend on the choices of the generators of $\mathcal{L}$. The *norm (or length) of a vector $\boldsymbol{f} = (f_1, \ldots, f_n) \in \mathcal{L}$* is given by

$$\|\boldsymbol{f}\| = \left(\sum_{1 \leq i \leq n} f_i^2 \right)^{1/2} = \langle \boldsymbol{f}, \boldsymbol{f} \rangle^{1/2} \in \mathbb{R}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors in $\mathbb{R}^n$. The norm $\|\boldsymbol{f}\|$ is also called Euclidean norm or L_2 norm.

A very interesting problem is to find the shortest vector in a lattice. However, this problem is $\mathcal{NP}$-hard [Ajt98]. Therefore, lots of work has been done on a relaxed topic — *lattice reduction*, which means to find a lattice basis such that it contains "relatively" short and almost orthogonal vectors. The earliest work regarding to the lattice reduction problem was done by Hermite [Her50], where whether the complexity of the algorithm is polynomial time is still an open question [NS06]. In 1982, lattice reduction was resurged by the Lenstra–Lenstra–Lovász (LLL or L^3) algorithm [LLL82]. It is an efficient method in polynomial time to find such a relatively short vector, length of which is *close* to the shortest one up to some constant factor.

Let us briefly revisit the Gram–Schmidt orthogonalization (GSO). It finds an orthogonal basis $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n$ of $\mathbb{R}^n$ given an arbitrary basis $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ of $\mathbb{R}^n$ where $\boldsymbol{f}'$ are defined as follows

$$\boldsymbol{f}'_i = \boldsymbol{f}_i - \sum_{1 \le j < i} \mu_{ij} \boldsymbol{f}'_j, \text{ where } \mu_{ij} = \frac{\langle \boldsymbol{f}_i, \boldsymbol{f}'_j \rangle}{\|\boldsymbol{f}'_j\|^2} \text{ for } 1 \le j < i$$

with initialization $\boldsymbol{f}'_1 = \boldsymbol{f}_1$. Algorithm 22 describes the inductive procedure of GSO, and outputs an orthogonal basis $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n$ and the matrix M of the linear transform

$$(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)^T = M(\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n)^T.$$

The Gram–Schmidt orthogonalization mainly performs Gaussian elimination on the *Gramian matrix* $\big(\langle \boldsymbol{f}_i, \boldsymbol{f}_j \rangle\big)_{1 \le i,j \le n} \in \mathbb{R}^{n \times n}$. GSO is carried out with $\mathcal{O}(n^3)$ arithmetic operations in $\mathbb{R}$.

Algorithm 22: Gram–Schmidt orthogonalization (GSO)

```
1  Input: A basis f_1, ..., f_n ∈ R^n
2  begin
3      f'_1 ← f_1
4      for i from 1 to n do
5          δ ← 0
6          for j from 1 to i do
7              μ_ij ← ⟨f_i, f'_j⟩ / ||f'_j||^2
8              δ ← δ + μ_ij f'_j
9          f'_i ← f_i − δ
10 Output: A basis f'_1, ..., f'_n and a lower triangle matrix M = (μ_ij)
```

From Line 6 to Line 9, the $\boldsymbol{f}'_i$ are computed as follows.

$$\boldsymbol{f}'_i = \boldsymbol{f}_i - \sum_{1 \le j < i} \mu_{ij} \boldsymbol{f}'_j.$$

One can consider $\boldsymbol{f}'_i$ which is obtained by removing the projection part onto $\boldsymbol{f}'_j$ $1 \le j < i$, from $\boldsymbol{f}_i$, in particular, $\|\boldsymbol{f}'_i\| \le \|\boldsymbol{f}_i\|$ which immediately leads to the following theorem.

Theorem 45 (Hadamard's inequality). *Given* $A = (\boldsymbol{f}_1^T, \ldots, \boldsymbol{f}_n^T)^T \in \mathbb{R}^{n\times n}$ *with* $\boldsymbol{f}_i = (f_{i1}, \ldots, f_{in})$*, and* $B = \max\{|f_{ij}|_{1\le i,j\le n} \in \mathbb{R}\}$*. Then*

$$|\det A| \le \|\boldsymbol{f}_1\| \cdots \|\boldsymbol{f}_n\| \le n^{n/2} B^n.$$

There are some remarks regarding to GSO. Firstly, the norm of the lattice spanned by the GSO basis doesn't change, but the norm of the vectors in the basis are shortened. Secondly, for any $\boldsymbol{f} \in \mathcal{L}$, we have

$$\|\boldsymbol{f}\| \ge \min\{\|\boldsymbol{f}'_1\|, \ldots, \|\boldsymbol{f}'_n\|\}.$$

In our problem, we consider the integer lattice $\mathcal{L}$ generated by a basis $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{Z}^n$. After GSO, we will get an orthogonal basis $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n \in \mathbb{Q}^n$. If the GSO basis is the one for the lattice generated by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{Z}^n$, then one of the $\boldsymbol{f}'_i$ is a shortest one. On the other hand, the corresponding GSO basis can not be guaranteed to still stay in the lattice generated by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ since $\boldsymbol{f}'_i \in \mathbb{Q}^n$. We would like to search an orthogonal or quasi-orthogonal basis where all the basis vectors $\boldsymbol{f}'_i$ are in the integer lattice generated by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$, and the shortest vector is in the basis.

Definition 20 (Reduced basis). *Given* $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ *a basis of* $\mathbb{R}^n$ *and* $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n$ *the corresponding GSO basis. Then* $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ *is a reduced basis if*

$$\|\boldsymbol{f}'_i\|^2 \le 2\|\boldsymbol{f}'_{i+1}\|, \quad \text{for } 1 \le i < n.$$

Theorem 46 ([GG03], Theorem 16.9). *Let* $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ *be a reduced basis of the lattice* $\mathcal{L} \subseteq \mathbb{R}$*. Then* $\|\boldsymbol{f}_1\| \le 2^{(n-1)/2}\|\boldsymbol{f}\|$ *for any* $\boldsymbol{f} \in \mathcal{L}$*.*

Now we present the Algorithm 23 which is called Lenstra–Lenstra–Lovász (LLL) basis reduction [LLL82] algorithm. It computes a reduced basis of a lattice in $\mathbb{Z}$ from any arbitrary basis in $\mathbb{Z}$. The reduced basis is a subset of the lattice generated by the input basis, and the reduced basis is almost orthogonal. From Theorem 46, the LLL algorithm guarantees to find a "relatively short" vector which is at most $2^{(n-1)/2}$ times larger than the shortest one where n is the dimension of the lattice.

The algorithm runs repeatably in two steps: The *replacement* step in the `for` loop and the *swap* step in the `if` check. The sub-function `Round` in Line 7 is to round $\mu_{ij} \in \mathbb{R}$ to the nearest integer, i.e., $\texttt{Round}(\mu_{ij}) = \lfloor \mu_{ij} + 1/2 \rfloor$.

Let $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n \in \mathbb{Z}^n$ and $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{Z}^n$ be the bases before and after the replacement or the swapping, respectively. Let $\boldsymbol{g}'_1, \ldots, \boldsymbol{g}'_n \in \mathbb{Q}^n$ and $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n \in \mathbb{Q}^n$ be the their corresponding GSO bases. In the first step, the GSO basis before and after the replacement does not change ([GG03, Lemma16.12]), i.e.,

$$\boldsymbol{g}'_i = \boldsymbol{f}'_i \text{ for } 1 \le i \le n. \tag{6.29}$$

Indeed, the spaces which are spanned by $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$ and $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$ are the same. After the `for` loop, we have $|\mu_{ij}| \le 1/2$ for $1 \le j < i$, $1 \le i \le n$.

Algorithm 23: LLL basis reduction

1 **Input**: An arbitrary basis $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{Z}^n$
2 **begin**
3 $\quad \boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n, (\mu_{ij}) \leftarrow \texttt{GSO}(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)$
4 $\quad i \leftarrow 2$
5 $\quad$ **while** $i \leq n$ **do**
6 $\quad\quad$ **for** j *from* i *to* 1 **do**
7 $\quad\quad\quad$ replace $\boldsymbol{f}_i$ by $(\boldsymbol{f}_i - \texttt{Round}(\mu_{ij})\boldsymbol{f}_j)$
8 $\quad\quad\quad \boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n, (\mu_{ij}) \leftarrow \texttt{GSO}(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)$
9 $\quad\quad$ **if** $i > 1$ *and* $\|\boldsymbol{f}'_{i-1}\|^2 > 2\|\boldsymbol{f}'_i\|^2$ **then**
10 $\quad\quad\quad$ swap $\boldsymbol{f}_{i-1}$ and $\boldsymbol{f}_i$
11 $\quad\quad\quad \boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n, (\mu_{ij}) \leftarrow \texttt{GSO}(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n)$
12 $\quad\quad\quad i \leftarrow i - 1$
13 $\quad\quad$ **else**
14 $\quad\quad\quad i \leftarrow i + 1$
15 **Output**: A reduced orthogonal basis $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n \subseteq \mathcal{L}_{\mathbb{Z}}$

In the second step, after Line 11, the vectors in the GSO basis remain the same except $\boldsymbol{f}'_i$ and $\boldsymbol{f}'_{i-1}$ ([GG03, Lemma 16.13]), i.e.,

$$\begin{aligned} \boldsymbol{f}'_k &= \boldsymbol{g}'_k \text{ for } k = \{1, \ldots, n\} \setminus \{i-1, i\}, && (6.30) \\ \|\boldsymbol{f}'_{i-1}\|^2 &\leq \frac{3}{4}\|\boldsymbol{g}'_{i-1}\|^2, && (6.31) \\ \|\boldsymbol{f}'_i\| &\leq \|\boldsymbol{g}'_{i-1}\|. \end{aligned}$$

Convergence

Definition 21 (Gramian determinant). *Given a matrix* $F_k = (\boldsymbol{f}_1^T, \ldots, \boldsymbol{f}_k^T)^T \in \mathbb{Z}^{k \times n}$, $1 \leq k \leq n$. *The Gramian determinant* d_k *of* F_k *is the determinant of its Gramian matrix* $F_k F_k^T = \langle \boldsymbol{f}_i, \boldsymbol{f}_j \rangle_{1 \leq i,j \leq k} \in \mathbb{R}^{k \times k}$, *i.e.,*

$$d_k = \det(F_k F_k^T) \in \mathbb{Z}.$$

In [GG03, Lemma 16.15], the Gramian determinant is $d_k = \prod_{1 \leq i \leq k} \|\boldsymbol{f}'_i\|^2 > 0$, for $1 \leq k \leq n$. Let us denote d'_k as the new value of d_k after replacement or swapping in one `while` loop of Algorithm 23. In the replacement step, according to (6.29), we obtain

$$d'_k = d_k.$$

In the swapping step, for $1 \leq k < i-1$, we have $d'_k = d_k$ due to (6.30); for $k = i-1$, since $d_{i-1} = d_{i-2}\|\boldsymbol{f}'_{i-1}\|^2$ and (6.31), we have $d'_{i-1} \leq (3/4)d_{i-1}$; the swapping of matrix can be

considered as the original matrix multiplied with a permutation matrix whose determinant is 1 or -1, hence for $i-1 < k \leq n$, $d'_k = d_k$. As a result ([GG03, Lemma 16.16]),

$$\begin{array}{rcll} d'_{i-1} & \leq & (3/4)d_{i-1}, & \text{for some } 1 < i \leq n, \\ d'_k & = & d_k, & \text{for } k = \{1, \ldots, n\} \setminus \{i-1\}. \end{array}$$

Now let us see that the LLL algorithm converges at a finite number of steps. Let us consider a variant

$$D = \prod_{1 \leq k \leq n-1} d_k. \tag{6.32}$$

Algorithm 23 every time runs the `while` loop, the value of D decreases by at least a factor of $(3/4)$. Since $D \in \mathbb{Z}_+$, Algorithm 23 converges. The reason that d_n is not included in (6.32) is as follows. From the above deduction, either in the replacement step or in the swapping step, the d_n does not change. The tendency of D is exactly the same as Dd_n and much smaller by a factor of d_n. Thus, we use D instead of Dd_n for complexity computation.

Complexity

The GSO in Line 3 of Algorithm 23 costs $\mathcal{O}(n^3)$ arithmetic operations in $\mathbb{Q}$.

The replacement in Line 7 and 8 costs $\mathcal{O}(n)$ operations in $\mathbb{Q}$ because the GSO basis $\boldsymbol{f}'_1, \ldots, \boldsymbol{f}'_n$ doesn't change, only one row in the matrix (μ_{ij}) is updated. Thus the `for` loop needs $\mathcal{O}(n^2)$ operations in $\mathbb{Q}$. In Line 9, computing the squared norm has $\mathcal{O}(n)$ multiplications and additions in $\mathbb{Z}$. If the vectors are needed to be swapped, then only $\boldsymbol{f}'_{i-1}$ and $\boldsymbol{f}'_i$ are recomputed in Line 11 while other GSO basis vectors remain the same, which yields $\mathcal{O}(n)$ operations in $\mathbb{Q}$. Overall, the complexity of every `while` loop is $\mathcal{O}(n^2)$ in $\mathbb{Q}$.

How many iteration does Algorithm 23 need? In the last subsection, we already discussed the variant D decreases in each iteration at least by a factor of $3/4$. The initial value D_0 is calculated by

$$\begin{array}{rcl} D_0 & = & (\|\boldsymbol{f}'_1\|^2) \cdot (\|\boldsymbol{f}'_1\|^2 \|\boldsymbol{f}'_2\|^2) \cdot \cdots \cdot (\|\boldsymbol{f}'_1\|^2 \cdots \|\boldsymbol{f}'_{n-1}\|^2) \\ & \leq & (\|\boldsymbol{f}_1\|^2) \cdot (\|\boldsymbol{f}_1\|^2 \|\boldsymbol{f}_2\|^2) \cdot \cdots \cdot (\|\boldsymbol{f}_1\|^2 \cdots \|\boldsymbol{f}_{n-1}\|^2) \\ & \leq & A^{2(1+2+\cdots+n-1)} \\ & = & A^{n(n-1)} \end{array}$$

where $A = \max\{\|\boldsymbol{f}_i\|_{1 \leq i \leq n}\}$. The decrease of D implies that, the maximum number of iterations is $\log_{3/4} D_0 \in \mathcal{O}(n^2 \log A)$.

To sum up, the complexity of Algorithm 23 is $\mathcal{O}(n^4 \log A)$ arithmetic operations in $\mathbb{Q}$.

6.4.3 Decoding of ICR Codes

Recently, Nielsen [Nie13b] used a module minimization approach to solve multiple key equations over polynomial ring $\mathbb{F}[x]$, such as those arising when decoding IRS codes. We will apply essentially the same approach for our key equations, but the algebraic differences between $\mathbb{F}[x]$ and $\mathbb{Z}$ implies fundamental differences in the final algorithms.

Let us recall the system of key equations (6.28). The l-th key equation means that there exists some $v_l \in \mathbb{Z}$ such that $\Lambda R^{(l)} - v_l N = \Lambda C^{(l)}$. We can collect these ℓ equations into one in a vectorized form and say that $\boldsymbol{s} = (\Lambda, \Lambda C^{(1)}, \ldots, \Lambda C^{(\ell)})$ must be a vector in the $\mathbb{Z}$-row space of the matrix

$$M = \begin{pmatrix} 1 & R^{(1)} & R^{(2)} & \ldots & R^{(\ell)} \\ 0 & N & 0 & \ldots & 0 \\ 0 & 0 & N & \ldots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \ldots & N \end{pmatrix}. \tag{6.33}$$

The essential observation is now that whenever few errors have occurred, $\boldsymbol{s}$ is often the *shortest* vector in the row space of M; we will explain and formalize this later with Theorem 48. To increase the probability that $\boldsymbol{s}$ is the shortest vector, we will actually consider the row space of M_ω, a weighted version of M, where we scale the i-th column with some $\omega_i \in \mathbb{Z}$. We are thus seeking a $\boldsymbol{s}_\omega = (\Lambda\omega_0, \Lambda C^{(1)}\omega_1, \ldots, \Lambda C^{(l)}\omega_\ell)$. We will come back to how exactly we assign the ω_i in Corollary 49.

As we explained in the last subsection, computing the shortest vector in the row space of a matrix under the L_2 norm is unfortunately an $\mathcal{NP}$-hard problem; however, the Lenstra–Lenstra–Lovász (LLL) algorithm [LLL82] in polynomial time finds a vector whose L_2 norm is at most $\gamma\|\boldsymbol{v}\|$, where $\boldsymbol{v}$ is a shortest vector and γ is a constant. In the worst case, $\gamma = \sqrt{2}^{\ell}$, where $\ell + 1$ is the dimension of the row space; however, the LLL and its modifications perform nicer in experiments with $\gamma \approx 1.02^{\ell+1}$ [NS06]. To be certain that our computation will lead us to $\boldsymbol{s}_\omega$, we must therefore not only be sure that $\boldsymbol{s}_\omega$ is the shortest vector in the row space, but that there are no other vectors of length at most $\gamma\|\boldsymbol{s}_\omega\|$.

Theorem 48 essentially says that whenever not too many errors have occurred, this is indeed almost always the case. Therefore, one can construct M_ω, apply the LLL algorithm to find a short vector in it, and with high probability, the output will be $\boldsymbol{s}_\omega$. This immediately leads to the decoding approach given as Algorithm 24.

Let us use an example with somewhat small numbers to illustrate the algorithm.

Example 7. *As in Example 5, let us use a list* $\mathcal{P} = \{3, 5, 7, 11, 13\}$ *and* $k = 2$ *to construct a single CR code. The interleaved CR code is constructed with* $\ell = 3$ *CR codes* $\mathcal{CR}(\mathcal{P}; 5, 2)$. *The received word matrix*

$$R = \begin{pmatrix} 0 & 0 & 2 & 5 & 9 \\ 0 & 3 & 5 & 10 & 12 \\ 2 & 0 & 0 & 6 & 1 \end{pmatrix}$$

with corresponding $R^{(1)} = 555,\ R^{(2)} = 3288,\ R^{(3)} = 10115$. *And* $N = 15015$. *The initial matrix*

$$M = \begin{pmatrix} 1 & 555 & 3288 & 10115 \\ 0 & 15015 & 0 & 0 \\ 0 & 0 & 15015 & 0 \\ 0 & 0 & 0 & 15015 \end{pmatrix}.$$

Algorithm 24: Decoding an ICR code

Input: The lists $\mathcal{P}$ and $\mathcal{K}$, the received words $\boldsymbol{r}^{(l)},\ l = 1, \ldots, \ell,\ N$
Preprocessing: $\omega_0, \ldots, \omega_\ell$ according to Corollary 49
begin
    Compute $R^{(1)}, \ldots, R^{(\ell)}$.
    Construct $\mathbf{M}$ using (6.33) and multiply the ith column by ω_i for $i = 0, \ldots, \ell$.
    Run the LLL algorithm which returns a short vector $\boldsymbol{v}_\omega = (v_{\omega 0}, v_{\omega 1}, \ldots, v_{\omega \ell})$.
    if $\boldsymbol{v}_{\omega 0}$ *has the form* $\omega_0 \Lambda$ *where* Λ *is a valid error locator* **then**
        Return Λ.
    else
        Return Fail.
Output: The error locator Λ or Fail

After multiplied with weights $(15, 1, 1, 1)$*, the weighted matrix*

$$M_\omega = \begin{pmatrix} 15 & 555 & 3288 & 10115 \\ 0 & 15015 & 0 & 0 \\ 0 & 0 & 15015 & 0 \\ 0 & 0 & 0 & 15015 \end{pmatrix}.$$

A short vector $\boldsymbol{v}_\omega$ *is in the first row of the matrix*

$$\begin{pmatrix} 825 & 495 & 660 & 770 \\ -420 & -525 & -1974 & 2065 \\ -1290 & -2685 & 2517 & 9800 \\ -2805 & 1320 & 759 & 385 \end{pmatrix}.$$

After running LLL algorithm. The error locator is $825/15 = 55 = 5 \cdot 11$ *which indicates the errors have occurred on the second and fourth positions. From any two of error-free positions, we know the information are* $C^{(1)} = 9,\ C^{(2)} = 12,\ C^{(3)} = 14$.

For a single CR code $\mathcal{CR}(5, 2)$*, only* 1 *error can be corrected by Algorithm 19 or Algorithm 20 according to* (6.14)*. With interleaving scheme, in our example, we can correct* 2 *errors which is beyond half the minimum distance of a single CR code. In fact, later we will know from* (6.43)*, the maximum decoding distance in our case is* 2.

Failure Probability

With the overall idea explained, we can go on to analyze the probability that the above algorithm will fail, and from this derive how to assign the weights ω_i. Our failure probability will depend on the unknown Λ, but we discuss in the next subsection how this can be interpreted as a decoding radius.

The algorithm fails when there is a vector in the row space M_ω different from $\boldsymbol{s}_\omega$ but which has L_2 norm within $\gamma\|\boldsymbol{s}_\omega\|$, and we will upper bound this probability. Our theorem will assume that certain values behave as independent, uniformly distributed random variables, and so we will need the following lemma:

Lemma 47. *Given some $N, T \in \mathbb{Z}$ with $T < N$ and $c_1, \ldots, c_\ell \in \mathbb{Z}_+$, and let $X_1, \ldots, X_\ell$ be independent random variables, uniformly distributed on $[0, N-1]$. Then*

$$P = \text{Prob}[c_1X_1 + \ldots + c_\ell X_\ell < T] \leq \frac{T^\ell}{\ell! N^\ell c_1 \cdots c_\ell}. \tag{6.34}$$

Proof: To prove the lemma we relax integer variables X_i to real valued variables uniformly distributed on $[0, N]$ and show that for this case (6.34) holds with equality. If N is large, which will be our case, this relaxation gives a very precise upper bound (6.34) for the probability P in the lemma.

Assume that X_i are real valued variables uniformly distributed on $[0, N]$ with density $p[X_i = b] = 1/N$ for $b \in [0, N]$ and define the sum of the first i items $S_i = c_1X_1 + \ldots + c_iX_i$ for $1 < i \leq \ell$. Then we have

$$\begin{aligned} P[S_i < T] &= \int_0^{T/c_i} p[X_i = b] P[S_{i-1} < T - c_i b_i] \; db \\ &= \frac{1}{N} \int_0^{T/c_i} P[S_{i-1} < T - c_i b_i] \; db. \end{aligned} \tag{6.35}$$

We proceed by induction on i. For the base case $i = 1$, observe that

$$P[S_1 < T] = P[c_1X_1 < T] = P[X_1 < T/c_1] = \frac{T}{Nc_1}.$$

For the induction step, we continue from (6.35) using the induction hypothesis:

$$\begin{aligned} P[S_i < T] &= \frac{1}{N} \int_0^{T/c_i} P[S_{i-1} < T - c_i b] \; db \\ &= \frac{1}{N} \int_0^{T/c_i} \frac{(T - c_i b)^{i-1}}{(i-1)! N^{i-1} c_1 \ldots c_{i-1}} \; db \\ &= \frac{c_i^{i-1}}{(i-1)! N^i c_1 \ldots c_{i-1}} \int_0^{T/c_i} \left(\frac{T}{c_i} - b \right)^{i-1} db. \end{aligned}$$

Since

$$\int_0^{T/c_i} \left(\frac{T}{c_i} - b \right)^{i-1} db = \frac{1}{i} \left(\frac{T}{c_i} \right)^i$$

we have proved the step of induction and for $i = \ell$ we obtain the expression of the lemma. □

Theorem 48. *Let A be a random variable, uniformly distributed on $1, \ldots, \lfloor T/\omega_0 \rfloor$, where T is defined below, and assume then that we can regard $(AR^{(l)} \mod N)$ for $l = 1, \ldots, \ell$ as ℓ independent random variables, uniformly distributed on $0, \ldots, N-1$.*

Assume that the LLL algorithm finds a vector whose L_2 norm is at most $\gamma \|\boldsymbol{v}_\omega\|$ where $\boldsymbol{v}_\omega$ is a shortest vector in the row space of M_ω. For a random error locator Λ, the probability of decoding failure $P_f(\Lambda)$ satisfies

$$P_f(\Lambda) \leq 1 - \left(1 - \frac{T^\ell}{\ell! N^\ell \omega_1 \cdots \omega_\ell}\right)^{T/\omega_0}$$

where $T = \widetilde{\gamma} \max\{\omega_0 \Lambda, \omega_1 \Lambda K_1, \ldots, \omega_\ell \Lambda K_\ell\}$ and $\widetilde{\gamma} = \sqrt{\gamma(\ell+1)}$.

Proof: There are two cases we need to consider: $p_i \nmid R^{(l)}$ and $p_i \mid R^{(l)}$.

If $p_i \nmid R^{(l)}, \forall i = 1, \ldots, n, l = 1, \ldots, \ell$, then the variables $(AR^{(l)} \mod N)$ are uniformly distributed in $[0, \ldots, N-1]$. The decoder can only fail if there is a vector $\boldsymbol{v}_\omega \neq \boldsymbol{s}_\omega$ with $\|\boldsymbol{v}_\omega\| < \gamma \|\boldsymbol{s}_\omega\|$, i.e.,

$$\sum_{l=0}^{\ell} (\omega_l v_l)^2 < \gamma \left((\omega_0 \Lambda)^2 + \sum_{j=1}^{\ell} (\omega_j \Lambda C^{(j)})^2 \right)$$

where $\omega_l v_l$ are the components of $\boldsymbol{v}_\omega$. Let $\widetilde{T} = \max\{\omega_0 \Lambda, \omega_1 \Lambda K_1, \ldots, \omega_\ell \Lambda K_\ell\}$; then the above can only occur if $\sum_{l=0}^{\ell} (\omega_l v_l)^2 < \gamma(\ell+1)\widetilde{T}^2$. Due to Cauchy–Schwartz inequality, that implies

$$\sum_{l=0}^{\ell} \omega_l v_l < \sqrt{\gamma(\ell+1)}\widetilde{T} = T. \tag{6.36}$$

We will upper bound $P_f(\Lambda)$ by the probability that a vector $\boldsymbol{v}_\omega \neq \boldsymbol{s}_\omega$ satisfying (6.36) is in the row space. Such a vector can be written in the form

$$\big(\omega_0 A,\ \omega_1 (AR^{(1)} \mod N),\ \ldots,\ \omega_\ell (AR^{(\ell)} \mod N)\big)$$

where $\omega_0 A \in \{1, \ldots, T-1\}$. Now we use Lemma 47 to over-approximate the probability that for a given $A \in \{1, \ldots, \lfloor T/\omega_0 \rfloor\}$, the associated vector of the above form satisfies (6.36):

$$\begin{aligned} P &= \mathrm{Prob}\Big[\sum_{j=1}^{\ell} \omega_i (AR^{(j)} \mod N) < T - \omega_0 A \Big] \\ &< \mathrm{Prob}\Big[\sum_{j=1}^{\ell} \omega_i (AR^{(j)} \mod N) < T \Big] \\ &\leq \frac{T^\ell}{\ell! N^\ell \omega_1 \ldots \omega_\ell}. \end{aligned} \tag{6.37}$$

Thus the probability that none of the T/ω_0 choices of A satisfies (6.36) becomes at least $(1-P)^{T/\omega_0}$, and the statement follows.

If for some $i \in [1, n]$ and some $l \in [1, \ell]$, $p_i \mid R^{(l)}$, then $p_i \mid (AR^{(l)} \mod N)$. Therefore, the variables $(AR^{(l)} \mod N)$ are clearly not uniformly distributed in $\{0, 1, \ldots, N-1\}$, but still behave close to uniformly distributed in $\{0, p_i, 2p_i, \ldots, N - p_i\}$. In this case, the vector $\boldsymbol{v}_\omega$ can be written in the following way:

$$\left(\omega_0 A,\ \omega_1 \left[AR^{(1)}\right]_N, \ldots, \omega_i p_i \left[\frac{AR^{(l)}}{p_i}\right]_{\frac{N}{p_i}}, \ldots, \omega_\ell \left[AR^{(\ell)}\right]_N\right).$$

In the end we reach the same failure probability, since

$$\frac{T^\ell}{\ell! N^{\ell-1} \frac{N}{p_i} \omega_1 \ldots \omega_i p_i \ldots \omega_\ell} = \frac{T^\ell}{\ell! N^\ell \omega_1 \ldots \omega_\ell}. \tag{6.38}$$

If there are some $p_j \neq p_i$ which divide any one in $R^{(1)}, \ldots, R^{(\ell)}$, same arguments can be iterated. □

Though we have not proved that the following choice of weights is optimal, it seems intuitive:

Corollary 49. *With the weights chosen as $\omega_0 = K_\ell$ and $\omega_i = K_\ell / K_i$, $i = 1, \ldots, \ell$, the failure probability becomes*

$$P_f(\Lambda) \leq 1 - \left(1 - \frac{\widetilde{\gamma}^\ell \Lambda^\ell \prod_{l=1}^{\ell} K_l}{\ell! N^\ell}\right)^{\widetilde{\gamma}\Lambda}. \tag{6.39}$$

Decoding Radius

The guaranteed unique decoding radius t_g of the ICR code is given by (6.14), where $K = \max\{K_l\}$ since any unique decoder must fail with non-zero probability when $t > t_g$. However, one could almost always find Λ using the best protected code, giving a "usual" decoding radius t_u from $K = \min\{K_l\}$. Thus, the traditional definition of decoding radius is not very useful. Exactly the same applies for the collaborative decoders of Reed–Solomon codes [SSB09, SSB07].

To decode a single CR code, it follows from Lemma (38) that the decoding will never fail if $\Lambda \leq \sqrt{N/(K-1)})$. For ICR code, when $\Lambda > \sqrt{N/(K-1)}$, we can also correct errors, but we can not guarantee that our decoding algorithm will always work in this case. Therefore, we define a threshold ϕ, and say that "If Λ is within some range, Algorithm 24 can decode with probability $1 - \phi$". One could then set ϕ satisfactorily low. For this definition, one can use the failure probability estimated in Theorem 48 as a starting point. Since we assume that all error patterns for given t are equally likely, each Λ with t factors occurs equally often; thus the probability of failure for a given number of errors t is

$$\overline{P}_f(t) = \binom{n}{t}^{-1} \sum_I P_f\left(p_{I_1} \cdots p_{I_t}\right) \tag{6.40}$$

where the sum runs over all subsets of $\{1, \ldots, n\}$ of size t.

If we set the failure threshold as ϕ in (6.39), i.e.,

$$1 - \left(1 - \frac{\widetilde{\gamma}^{\ell} \Lambda^{\ell} \prod_{l=1}^{\ell} K_l}{\ell! N^{\ell}}\right)^{\widetilde{\gamma}\Lambda} \leq \phi. \tag{6.41}$$

The term with exponent in (6.41) is approximated as

$$1 - \frac{\widetilde{\gamma}^{\ell} \Lambda^{\ell} \prod_{l=1}^{\ell} K_l}{\ell! N^{\ell}} \widetilde{\gamma}\Lambda.$$

According to the first order truncation of the binomial series, the error locator is bounded as follows

$$\Lambda \leq \left(\frac{\phi \ell!}{[\gamma(\ell+1)]^{\frac{\ell+1}{2}}}\right)^{\frac{1}{\ell+1}} \left(\frac{N}{\overline{K}}\right)^{\frac{\ell}{\ell+1}} \triangleq \alpha \left(\frac{N}{\overline{K}}\right)^{\frac{\ell}{\ell+1}}, \tag{6.42}$$

where $\overline{K} = \sqrt[\ell]{K_1 \cdots K_\ell}$ and α is some constant close to 1 which depends on ϕ, γ and ℓ. The decoding radius in Hamming metric of our algorithm, in the above sense, is

$$t \leq \left\lfloor \frac{\ell}{\ell+1} \frac{\log(N/\overline{K})}{\log p_n} + \frac{\log \alpha}{\log p_n} \right\rfloor \tag{6.43}$$

where the latter term is a constant. By experiments $\log \alpha / \log p_n$ is always negative and close to 0. Hence, one can use

$$t \lesssim \left\lfloor \frac{\ell}{\ell+1} \frac{\log(N/\overline{K})}{\log p_n} \right\rfloor$$

to upper bound the decoding radius in Hamming distance. Note that, to measure the correction capability for ICR codes, it is more precise to use weighted Hamming distance, i.e., $\log \Lambda$ or directly Λ than usual Hamming metric.

To decode a single CR code ($\ell = 1, \phi = 1$), (6.43) coincides with (6.14). One can use $\gamma = 1$, 1.02^2 or $\sqrt{2}$ in (6.43), the result will not change.

Example 8. *We will consider two ICR codes with low rate and high rate respectively.*

Let us consider the ICR code $\mathcal{ICR}(n_1 = 20, \mathcal{K}_1 = [3, 5])$*, i.e., interleaving factor* $\ell = 2$*, and with the list of primes* $\mathcal{P}_1 = [101, 103, \ldots, 197]$*. Table 6.2 shows the value of* $\log \alpha / \log p_n$ *which depends on* ϕ, γ *when* $\ell = 2$*. We can see that* $\log \alpha / \log p_n$ *is much smaller than* 1 *such that one can simply use* $\left\lfloor \frac{\ell}{\ell+1} \frac{\log(N/\overline{K})}{\log p_n} \right\rfloor$ *to bound the decoding radius. One can decode up to* 10 *errors pursuit to* (6.43)*.*

Let us consider the ICR code $\mathcal{ICR}(n_2 = 100, \mathcal{K}_2 = [81, 81, 82, 82, 83])$ *as a higher rate code, and with the list of primes* $\mathcal{P}_2 = [101, 103, \ldots, 691]$*. Similar behavior of* $\log \alpha / \log p_n$ *can be found in Table 6.3 in comparison with the above example.*

For single CR codes $\mathcal{CR}(\mathcal{P}_1; n_1 = 20, k_1 = 3)$ *and* $\mathcal{CR}(\mathcal{P}_2; n_2 = 100, k_2 = 81)$*, one can omit latter part in* (6.43) *for calculating correction radius since the absolute value of it is even smaller than* 0.1 *(See Table 6.4). Therefore,* (6.43) *coincides with* (6.14) *with* $\ell = 1, \phi = 1$*.*

	$\gamma = 1$	$\gamma = 1.02^{\ell+1}$	$\gamma = \sqrt{2}^{\ell}$
$\phi = 1$	-0.060	-0.066	-0.126
$\phi = 0.9$	-0.067	-0.073	-0.132

Table 6.2: The value of $\log \alpha / \log p_n$ in (6.43) for $\mathcal{ICR}(n_1 = 20, \mathcal{K}_1 = [3, 5])$

	$\gamma = 1$	$\gamma = 1.02^{\ell+1}$	$\gamma = \sqrt{2}^{\ell}$
$\phi = 1$	-0.015	-0.024	-0.148
$\phi = 0.9$	-0.018	-0.027	-0.150

Table 6.3: The value of $\log \alpha / \log p_n$ in (6.43) for $\mathcal{ICR}(n_2 = 100, \mathcal{K}_2 = [81, 81, 82, 82, 83])$

Complexity

The complexity of the LLL algorithm dominates the overall complexity in Algorithm 24. As stated in the last part of Subsection 6.4.2, Line 6 performs $\mathcal{O}((\ell+1)^4 \log A) \approx \mathcal{O}(\ell^4 \log A)$ operations on integers of bit-length $\mathcal{O}(\ell \log A)$ where A is the greatest norm of rows in M_ω. Choosing the ω_i as in Corollary 49, we have $A = \sqrt{\ell+1} N K_\ell / K_1$ which means $\log A < 1/2 \log(\ell+1) + 2n \log p_n$. The remaining computations of Algorithm 24 can be performed faster than this. In particular, since we know which primes are allowed to divide a valid error locator, the check in Line 7 can be done efficiently. Therefore, the complexity of Algorithm 24 is $\mathcal{O}(n\ell^{4.5} \log p_n)$ operations on integers of bit-length $\mathcal{O}(n\ell^{1.5} \log p_n)$.

Test Results

We have done quite intensive testing of the algorithm, and have in general observed that the failure probability of Theorem 48 coincides rather well with experiments: setting $\gamma = 1$, i.e., expecting that the LLL algorithm can always find the shortest vector, sometimes proves slightly too optimistic; while setting $\gamma = \sqrt{2}^{\ell}$, i.e., the worst case, is quite pessimistic. The difference between these two is usually within only a few errors, though.

We continue with our Example 8. Consider the ICR code $\mathcal{ICR}(n_1 = 20, \mathcal{K}_1 = [3, 5])$ with the list of primes $\mathcal{P}_1 = [101, 103, \ldots, 197]$. The guaranteed decoding radius for this ICR code is $t_g = 7$, while the "usual" radius is $t_u = 8$. Choosing the weights as in Corollary 49, we have run 10,000 tests with this code, creating random error patterns of weights ranging from 7 up to 12. For each number of errors t, we then calculated the following aggregate statistics

$$\begin{aligned} A_{Obs} &= \#\text{ of failures}/\#\text{ of tests}_t \\ A_T^\gamma &= \textstyle\sum_{\Lambda \in \text{tests}_t} P_f^\gamma(\Lambda)/\#\text{ of tests}_t, \end{aligned}$$

the latter calculated for $\gamma \in \{1, 1.02^{\ell+1}, \sqrt{2}^{\ell}\}$. We have also calculated

$$P_T^\gamma = P_f^\gamma(D_t),$$

	$\gamma = 1$	$\gamma = 1.02^2$	$\gamma = \sqrt{2}$
$\mathcal{CR}(\mathcal{P}_1; n_1 = 20, k_1 = 3)$	-0.066	-0.069	-0.098
$\mathcal{CR}(\mathcal{P}_2; n_2 = 100, k_2 = 81)$	-0.053	-0.056	-0.080

Table 6.4: The value of $\log \alpha / \log p_n$ in (6.43) for two CR codes with $\phi = 1$

i.e., the failure probability of the biggest Λ.

t	A_{Obs}	A_T^1	$A_T^{1.02^{\ell+1}}$	$A_T^{\sqrt{2}^\ell}$	P_T^1	$P_T^{1.02^{\ell+1}}$	$P_T^{\sqrt{2}^\ell}$
9	0	0	0	0	0	0	0
10	0	0	0	0.01	0.25	0.27	1.18
11	96.06	98.49	98.68	99.86	100	100	100
12	100	100	100	100	100	100	100

Table 6.5: Failure probabilities % ($n = 20$)

Table 6.5 summarizes our results. We see that the observed decoding failure rather sharply goes from 0% to 100%, and we see that the theoretical failure probabilities come very close to the observed behavior. It is interesting to note that with $\alpha = 1$, (6.43) evaluates to 10.

At a much higher rate, consider the ICR code $\mathcal{ICR}(100, [81, 81, 82, 82, 83])$, with the list of primes $\mathcal{P}_2 = [101, 103, \ldots, 691]$. The guaranteed decoding radius for this ICR code is $t_g = 8$, while the "usual" radius is $t_u = 9$. Running 10,000 tests with patterns of weights ranging from 14 to 18, and aggregating as before, we got the test results as given on Table 6.6. We see that the upper bounds of $P_T^{\hat{\gamma}}$ are quite pessimistic estimates on the average failure probability, and that it is slightly too optimistic to assume $\hat{\gamma} \leq 1.02$. According to (6.43) with $\alpha = 1$, the decoding radius is $t = 14$, which is also pessimistic.

t	A_{Obs}	A_T^1	$A_T^{1.02^{\ell+1}}$	$A_T^{\sqrt{2}^\ell}$	P_T^1	$P_T^{1.02^{\ell+1}}$	$P_T^{\sqrt{2}^\ell}$
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	4.68	3.51	3.81	10.71	99.77	99.98	100
17	89.66	87.25	87.79	94.84	100	100	100
18	99.94	99.95	99.95	100	100	100	100

Table 6.6: Failure probabilities % ($n = 100$)

Given an ICR code, the decoding radius is calculated in terms of (6.43). What will be the difference of the radius if we change the list of primes to a very large number or a very small one? The answer is in the following illustration. We consider two scenarios: one is

the "large" list which has a big range, e.g., we took $\mathcal{P}_3 = \{2, \ldots, 541\}$ of length 100 for $\mathcal{ICR}(100, [81, 81, 82, 82, 83])$ such that $p_n/p_1 \gg 1$; the other one is the "small"" list whose elements in the list of primes do not differ too much, e.g., $\mathcal{P}_4 = \{100003, \ldots, 101197\}$ of length 100 for the same ICR code, we have $p_n/p_1 \approx 1$. In comparison, we draw reference from $\mathcal{ICR}(\mathcal{P}_2; 100, [81, 81, 82, 82, 83])$.

Figure 6.2 depicts the performance using different lists of primes $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$ in decoding. The solid lines represent the theoretical failure probability obtained from (6.40) with

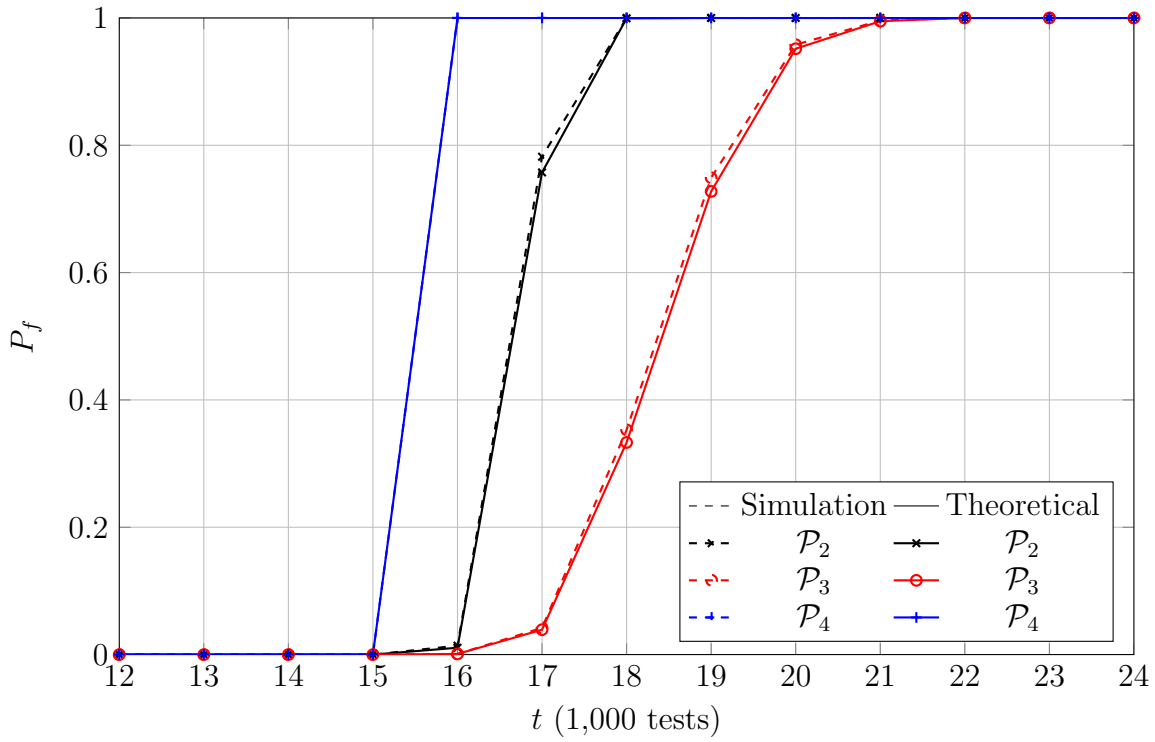


Figure 6.2: Simulation and theoretical results for $\mathcal{ICR}(100, [81, 81, 82, 82, 83])$ with $\mathcal{P}_2 = [101, \ldots]$, $\mathcal{P}_3 = [2, \ldots]$ and $\mathcal{P}_4 = [100003, \ldots]$.

$\gamma = 1.02^{\ell+1}$. In fact, given $t \in [12, \ldots, 24]$, we did not take all possible subsets $\mathcal{I}$ of size t because even $\binom{100}{12}$ has order $\mathcal{O}(10^{15})$ which exceeds the computation ability of CPU. Therefore, we draw $1,000$ random errors patterns of size t, calculate the failure probability for each pattern according to (6.39), and take the average value like (6.40) to compute theoretical failure probability, given fixed number of errors. Meanwhile, the failure probability for t errors by performing Algorithm 24 is obtained as well in each time running $1,000$ tests.

In Figure 6.2, for one prime set, the simulated curve almost coincides with the theoretical one, with the failure probability being slightly higher. The gap between dashed and solid lines gets smaller when "smaller" list is chosen. By "small", we mean the difference of the elements in a list of primes. We can see that when $\mathcal{P}_4$ is considered as a list of primes, two curves overlap each other.

By (6.43), the decoding radii regarding 3 primes lists are $t_{\mathcal{P}_2} = t_{\mathcal{P}_3} = 14$, $t_{\mathcal{P}_4} = 15$. An interesting observation is for "large" list of primes $\mathcal{P}_3$, the curve climbs slower than other ones. In other words, other than p_f for $\mathcal{P}_4$, the failure probability can not increase immediately to 1 when error number is gradually increasing beyond $t_{\mathcal{P}_3}$. This behavior comes down to different alphabets of each coordinate of the ICR codes. If the primes list has a large range such as $\mathcal{P}_3$, then an error at a "heavy" positions is equivalent to few errors at "light" positions. Since we use error locator Λ to bound the decoding radius, for the same Λ, the decoder can correct more errors at the light positions than those at heavy ones. One the other hand, for choosing lists of primes such as $\mathcal{P}_4$, the fact that similar weight of every coordinate leads to a very sharp increase of failure probability, and meanwhile gives a tighter bound of decoding radius than that for choosing big range lists of primes. As a result, the decoding radius for ICR codes in Hamming metric gives us an intuitive knowledge of the decoder performance, but it is more accurate to measure the weighted Hamming distance, or Λ in our case. The analysis of the influence of Λ to the failure probability can be considered as a future work.

As another reference, we show the comparison of simulation and theoretical results for $\mathcal{ICR}(n_1 = 20, \mathcal{K}_1[3, 5])$ with $\mathcal{P}_1$, $\mathcal{P}_3$ and $\mathcal{P}_4$. The theoretical results strictly follow (6.40), i.e., each subset I consists of Λs from *all* combination of t positions rather than Λs from random samples. The results are illustrated in Figure 6.3. Similar behavior to Figure 6.2

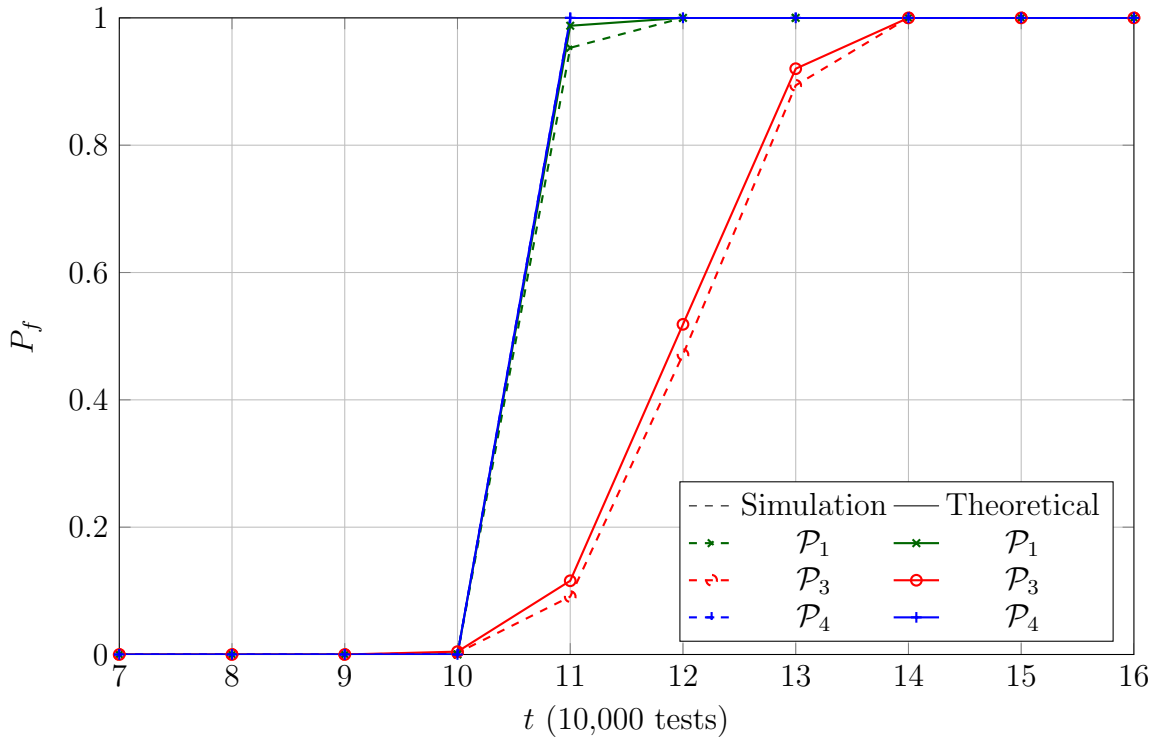


Figure 6.3: Simulation and theoretical results for $\mathcal{ICR}(20, [3, 5])$ with $\mathcal{P}_1 = [101, \dots]$, $\mathcal{P}_3 = [2, \dots]$ and $\mathcal{P}_4 = [100003, \dots]$.

for choosing different prime lists is shown. The theoretical lines are very close to those from simulations, respectively. The only difference is that, for the prime list with large range, the theoretical curve is a little higher than the simulated one. This is due to the way of calculating $\overline{P}_f(t)$ in (6.40). From Figure 6.2 and 6.3, one can use (6.40) to upper bound the failure probability of decoding ICR codes.

6.4.4 Power Decoding of a Single Low-rate CR Code

The power decoding of RS codes has already been descried in Section 4.5. In this part, we are dealing with power decoding of CR codes.

The key equation (6.13) for a single CR code can be "virtually extended" to multiple key equations whenever $K \ll N$; this technique, called "power decoding", was described for Reed–Solomon in [SSB10]. The resulting "virtually interleaved" code can be decoded by interleaved coding techniques beyond the unique decoding bound.

Each element of the received word is powered to be an element of a new CR code, i.e.,

$$\begin{aligned} \boldsymbol{r}^{(l)} &= ([r_1^l]_{p_1}, [r_2^l]_{p_2}, \ldots, [r_n^l]_{p_n}) \\ &= ([(c_1+e_1)^l]_{p_1}, [(c_2+e_2)^l]_{p_2}, \ldots, [(c_n+e_n)^l]_{p_n}) \\ &= ([c_1^l]_{p_1} + \tilde{e}_1, [c_2^l]_{p_2} + \tilde{e}_2, \ldots, [c_n^l]_{p_n} + \tilde{e}_n), \end{aligned}$$

where $\tilde{e}_i = [(c_1+e_1)^l - c_1^l]_{p_i}$. Note that the error positions do not change under powering. Therefore, a single CR code is virtually extended to an ICR code where each row has the same error locator. The cardinality of the new code $K_j = K^j$ can not be expressed by $F(\cdot)$, so these codes are not part of the classical definition, but generalized CR codes.

Recall that $N \mid \Lambda E$, for $l = 1, \ldots \ell$, the key equations for virtual extension are

$$\begin{aligned} \Lambda R^l \mod N &\equiv \Lambda(C+E)^l \mod N \\ &\equiv \Lambda C^l \mod N. \end{aligned}$$

Let ℓ be the greatest integer such that $\Lambda C^\ell < N$; the ℓ key equations for $l = 1, \ldots, \ell$ can be used to collaboratively determine Λ, and we can use exactly the same approach as we did for ICR codes.

Consider the $\mathbb{Z}$-row space of the matrix:

$$\left(\begin{array}{c|ccc} 1 & R & [R^2]_N & \ldots \quad [R^\ell]_N \\ \hline \mathbf{0} & \multicolumn{3}{c}{N\boldsymbol{I}} \end{array}\right), \tag{6.44}$$

where $\boldsymbol{I}$ is the $\ell \times \ell$ identity matrix. By the above key equations, the vector $(\Lambda, \Lambda C, \ldots, \Lambda C^\ell)$ will be in this space, and as in the case for ICR codes, it will be surprisingly short. We should choose weights for the columns, and emulating the choice of Corollary 49, we let $\omega_0 = K^\ell$ and $\omega_j = K^{\ell-l}$ for $l = 1, \ldots, \ell$.

The failure probability of Theorem 48 can be reused for this case. However, one should be noted that this is under heavier assumptions of randomness, since the various R-values, $R, [R^2]_N, \ldots, [R^\ell]_N$ obviously are more connected than for the usual ICR setting. We have

by simulation confirmed that the approach works and we can decode beyond the unique decoding bound; however, more experiments are needed for proper verification that the failure probabilities are well estimated.

Above, we chose ℓ depending on the unknown Λ and C, which is obviously problematic. Instead, one could choose a decoding radius t and choose ℓ maximal such that $D_t K^\ell < N$. However, since more interleaving allows higher decoding radius, there is a non-trivial connection here. Furthermore, since random Λ are usually much lower than D_t and random C lower than K, it might be the case that the decoding case at hand would benefit from a higher interleaving factor. We have not thoroughly investigated this issue.

6.4.5 Decoding of Low-rate ICR Codes

Power decoding can be straightforwardly combined with the ICR decoder, whenever one interleaves CR codes of low rate; this idea was first proposed for Reed–Solomon codes in [SSB07]. We will briefly sketch the idea, but we have not yet deeply analyzed this setting.

Theorem 48 essentially says that whenever not too many errors have occurred, this is indeed almost always the case. Therefore, one can construct M_ω, apply the LLL algorithm to find a short vector in it, and with high probability, the output will be $\boldsymbol{s}_\omega$. This immediately leads to the decoding algorithm given as Algorithm 24.

Consider a code $\mathcal{ICR}(\mathcal{P}; n, \mathcal{K} = [K_1, \ldots, K_\ell])$ as well as received matrix with rows $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_\ell$. Define the corresponding $R_1, \ldots, R_\ell$. For each of these, we can get virtually extended key equations $\Lambda[R_i^j]_N \equiv \Lambda C_i^j \mod N$ for $j = 1, \ldots, \rho_i$, where ρ_i is chosen maximally such that $\Lambda C_i^{\rho_i} < N$. This means that vector $\left(\Lambda, \Lambda C_1, \ldots, \Lambda C_1^{\rho_1}, \ldots, \Lambda C_\ell, \ldots, \Lambda C_\ell^{\rho_\ell}\right)$ is in the row space of the matrix:

$$\left(\begin{array}{c|cccccccc} 1 & [R_1^1]_N & \ldots & [R_1^{\rho_1}]_N & \ldots & [R_\ell^1]_N & \ldots & [R_\ell^{\rho_\ell}]_N \\ \hline \mathbf{0} & \multicolumn{7}{c}{N\boldsymbol{I}} \end{array}\right)$$

where $\boldsymbol{I}$ is an appropriately sized identity matrix From here the decoding algorithm progress as in Algorithm 24.

The issue with how to choose the ρ_i is even more compounded in this setting than for the Power decoding, and more analysis is needed for determining the right choice while minimizing computational effort. We also note that one can perform a "mixing" of the key equations, as for IRS codes in [WZZB12], to get a larger matrix and decode more errors.

6.5 Discussion and Future Work

For a Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ with minimum distance d, we propose a syndrome-based decoder which can correct up to $\lfloor \log(N/K)/(2 \log p_n) \rfloor$ errors with complexity $\mathcal{O}(\mathrm{len}(N) \log^\epsilon \mathrm{len}(N))$ in $\mathbb{Z}$, i.e., the bit size of N. The number of correctable errors of the can also be represented as $\lfloor (d-1) \frac{\log p_1}{\log p_1 + \log p_n} \rfloor$ which indicates a direct connection to the minimum distance. If the first and the last primes in the prime set $\mathcal{P}$ do not differ too much, then the number or errors which can be corrected almost reaches half the minimum

distance. Consider our proposed decoder, if there exist a method to calculate the syndrome S directly without computing R, then this is similar to obtaining syndrome for the RS codes, hence the algorithm becomes more efficient.

We generalize the Chinese remainder code with cardinality $K = \prod_{i=1}^{k} p_i$ to the Chinese remainder code with an arbitrary cardinality $0 \leq K' \leq K$ in order to correct both errors and erasures for a Chinese remainder code. The error-erasure decoder can correct t errors and τ erasures, provided that $\frac{\log p_1 + \log p_n}{\log p_1} t + \tau \leq d - 1$. The cost of the corresponding algorithm is almost linear in bit size of N, i.e., $\mathcal{O}(\text{len}(N) \log^{\epsilon} \text{len}(N))$.

We propose an algorithm using lattice reduction to decode an interleaved Chinese remainder code $\mathcal{ICR}(\mathcal{P}; n, \mathcal{K})$ correcting burst errors. The complexity of the algorithm is $O(n\ell^{4.5} \log p_n)$ operations on integers of bit length $O(n\ell^{1.5} \log p_n)$. Regarding the failure probability and the decoding radius of the decoder, theoretical and simulation results practically coincide. Nevertheless, we still need more experiments for proper verification that the failure probabilities are well estimated. The technique of power decoding can also be applied to decode low-rate Chinese remainder codes and low-rate interleaved Chinese remainder codes, for which deep analysis is still needed.

7

Conclusion

Within this dissertation, the decoding of four classes of evaluation codes and their interleaving was considered. More specifically, regarding the RS codes, Hermitian codes, Gabidulin codes, and Chinese remainder codes together with their interleaved codes, several decoding algorithms were proposed. Meanwhile, their performances were deeply studied in terms of complexity, decoding radius, and failure probability, if any. The new results in this dissertation concerning each class of evaluation codes are listed as follows.

RS Codes

Algorithms for decoding IRS codes based on the generalized extended Euclidean algorithm were considered in Chapter 3. To solve the multi-sequence linear feedback shift register synthesis problem, we modified Feng–Tzeng's generalized extended Euclidean algorithm (GEEA) [FT89] such that the input syndrome sequences can have different lengths and the obtained error locator polynomial is a minimal solution. Their algorithm requires quadratic number of field operations in the maximum sequence length. Our modified algorithm was accelerated by "divide and conquer" strategy, having sub-quadratic complexity in the maximum sequence length.

Hermitian Codes

Decoding (N, K) Hermitian codes in $\mathbb{F}_Q = \mathbb{F}_{q^2}$ was investigated in Chapter 4. Since correcting Hermitian codes with bursts of errors can be reduced to decoding IERS codes, an algorithm for joint decoding IERS codes was proposed first, having quadratic complexity in length. The proposed algorithm was then applied for decoding Hermitian codes having rate at least $1/2q$, correcting up to $t_{\max} = (N-K)/(q+1)$ bursts with time complexity $\mathcal{O}(N^{5/3})$ operations in $\mathbb{F}_Q$ which is lower than $\mathcal{O}(N^3)$ from Özbudak and Yayla. For the failure probability $P_f(t)$ we gave an upper bound $P_f(t) \leq \gamma Q^{-(q+1)(t_{\max}-t)-1}$, which is at most $1/Q$ when $t = t_{\max}$, and exponentially drops while t decreases. For up to $(N/q - k_{\max})/2$ burst errors where $k_{\max}$ is the maximum dimension of the transformed IERS codes, $P_f(t) = 0$.

Simulations showed that the bound is precise. As a consequence, our algorithm has less failure probability than that in [Ren04], and lower complexity and better bound on decoding failure probability in comparison with Özbudak–Yayla's conclusion [OY14].

We also showed that low rate Hermitian codes can correct even more bursts of errors using "power" and "mixed" decoding.

Gabidulin Codes

Chapter 5 firstly presented a transform domain decoding algorithm for (n, k) Gabidulin codes with minimum distance d in $\mathbb{F}_{q^m}$. It corrects ε full errors, μ_{R} row erasures and μ_{C} column erasures as long as $2\varepsilon + \mu_{\mathrm{R}} + \mu_{\mathrm{C}} \leq d - 1$ with complexity $\mathcal{O}(n^2)$ operations in $\mathbb{F}_{q^m}$, which is the same as those suggested in [SKK08, GY08]. Nevertheless, the transform-domain approach dramatically decreases the number of decoding steps in the algorithm and simplifies the relevant proofs. This algorithms was then generalized for interleaved Gabidulin codes, where an upper bound of failure probability was given. Concerning Gabidulin codes and their interleaved codes with a fixed interleaving order, both algorithms have quadratic complexity in the code length.

Chinese Remainder Codes

The Chinese remainder codes and their interleaving were the theme of Chapter 6. Regarding a Chinese remainder code $\mathcal{CR}(\mathcal{P}; n, K)$ with minimum distance d, we proposed a syndrome-based decoder which corrects up to $\lfloor \log(N/K)/(2 \log p_n) \rfloor$ errors with linear complexity in the bit size of N, i.e., $\mathcal{O}(\mathrm{len}(N) \log^{\epsilon} \mathrm{len}(N))$ in $\mathbb{Z}$ for some constant ϵ. The decoding radius in Hamming metric reaches half the minimum distance when sizes of the primes do not differ too much. An algorithm for decoding Chinese remainder codes was proposed, correcting t errors and τ erasures, provided that $\frac{\log p_1 + \log p_n}{\log p_1} t + \tau \leq d - 1$. The cost of the corresponding algorithm is almost linear in bit sized of N, i.e., $\mathcal{O}(\mathrm{len}(N) \log^{\epsilon} \mathrm{len}(N))$. We also proposed an algorithm to decode interleaved Chinese remainder codes $\mathcal{ICR}(\mathcal{P}; n, \mathcal{K})$ with burst errors, having complexity $O(n\ell^{4.5} \log p_n)$ operations on integers of bit-length $O(n\ell^{1.5} \log p_n)$. Regarding the failure probability and the decoding radius, simulations showed that our theoretical results are precise.

Discussions and future research directions were given at the end of each chapter.

List of Algorithms

Bibliography

References

[ACLY00] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network Information Flow," *Inform. Theory, IEEE Trans. on*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, 1st ed. Addison-Wesley, Jan. 1974.

[Ajt98] M. Ajtai, "The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions (extended abstract)," in *Proc. of the 30th annual ACM Symp. on Theory of computing*, ser. STOC '98. ACM Press, 1998, pp. 10–19.

[BB13] M. Bossert and S. Bezzateev, "A Unified View on Known Algebraic Decoding Algorithms and New Decoding Concepts," *Information Theory, IEEE Transactions on*, vol. 59, no. 11, pp. 7320–7336, Nov. 2013.

[Ber68] E. R. Berlekamp, *Algebraic Coding Theory*. McGraw-Hill, 1968.

[Bla83] R. E. Blahut, *Theory and Practice of Error Control Codes*, reprint. with corr ed. Addison-Wesley, 1983.

[Bla85] ——, *Fast Algorithms for Digital Signal Processing*, 1st ed. Addison-Wesley, Jan. 1985.

[BMvT78] E. R. Berlekamp, R. McEliece, and H. van Tilborg, "On the Inherent Intractability of Certain Coding Problems (Corresp.)," *Inform. Theory, IEEE Trans. on*, vol. 24, no. 3, pp. 384–386, May 1978.

[Bos99] M. Bossert, *Channel Coding for Telecommunications*, 1st ed. Wiley, Oct. 1999.

[BS96] M. Bossert and V. Sidorenko, "Singleton-Type Bounds for Blot-Correcting Codes," *Inform. Theory, IEEE Trans. on*, vol. 42, no. 3, pp. 1021–1023, May 1996.

[Chi64] R. Chien, "Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem codes," *Inform. Theory, IEEE Trans. on*, vol. 10, no. 4, pp. 357–363, Oct. 1964.

[Del78] P. Delsarte, "Bilinear Forms over a Finite Field, with Applications to Coding Theory," *Journal of Combinatorial Theory, Series A*, vol. 25, no. 3, pp. 226–241, 1978.

[For65] G. Forney, "On Decoding BCH Codes," *Inform. Theory, IEEE Trans. on*, vol. 11, no. 4, pp. 549–557, Oct. 1965.

[FR93] G. L. Feng and T. R. N. Rao, "Decoding Algebraic-Geometric Codes up to the Designed Minimum Distance," *Inform. Theory, IEEE Trans. on*, vol. 39, no. 1, pp. 37–45, Jan. 1993.

[FT85] G. L. Feng and K. K. Tzeng, "An Iterative Algorithm of Shift-Register Synthesis for Multiple Sequences," *Science China Mathematics*, vol. 28, no. 11, pp. 1222–1232, Nov. 1985.

[FT89] ——, "A Generalized Euclidean Algorithm for Multisequence Shift-Register Synthesis," *Inform. Theory, IEEE Trans. on*, vol. 35, no. 3, pp. 584–594, 1989.

[FT91] ——, "A Generalization of the Berlekamp-Massey Algorithm for Multisequence Shift-Register Synthesis with Applications to Decoding Cyclic Codes," *Inform. Theory, IEEE Trans. on*, vol. 37, no. 5, pp. 1274–1287, Sep. 1991.

[Gab85] E. M. Gabidulin, "Theory of Codes with Maximum Rank Distance," *Probl. of Inform. Transm.*, vol. 21, no. 1, pp. 1–12, Jul. 1985.

[GG03] J. v. z. Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, Jul. 2003.

[Gop77] V. D. Goppa, "Codes Associated with Divisors," *Problemy Peredachi Informatsii*, vol. 13, no. 1, pp. 33–39, 1977.

[Gor73] W. C. Gore, "Transmitting binary symbols with Reed-Solomon codes," in *Proc. of the 7th Annual Princeton Conf. on Inform. Sciences and Systems*. Dept. of Electrical Engineering, Princeton University, Princeton, NJ, 1973, pp. 495–499.

[GP08] E. M. Gabidulin and N. Pilipchuk, "Error and erasure correcting algorithms for rank codes," *Designs, Codes and Cryptography*, vol. 49, no. 1-3, pp. 105–122, 2008.

[GPT91] E. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, "Rank Errors and Rank Erasures Correction," in *Proc. 4th Int. Colloquium on Coding Theory*, Oct. 1991, pp. 11–19.

[GRS00] O. Goldreich, D. Ron, and M. Sudan, "Chinese Remaindering with Errors," *Inform. Theory, IEEE Trans. on*, vol. 46, no. 4, pp. 1330–1338, Jul. 2000.

[GS98] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes," *Inform. Theory, IEEE Trans. on*, vol. 45, pp. 1757–1767, 1998.

[GS99] ——, "Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes," *Inform. Theory, IEEE Trans. on*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.

[GSS00] V. Guruswami, A. Sahai, and M. Sudan, ""Soft-Decision" Decoding of Chinese Remainder Codes," in *Proc. of the 41st IEEE Symp. on Foundations of Computer Science*, 2000, pp. 159–168.

[GY08] M. Gadouleau and Z. Yan, "Complexity of decoding Gabidulin codes," in *Inform. Sciences and Systems, The 42nd Annual Conference on*, Mar. 2008, pp. 1081–1085.

[Her50] C. Hermite, "Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objects de la théorie des nombres," *Journal für die reine und angewandte Mathematik*, vol. 40, pp. 261–277, 1850.

[HKM+03] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Inform. Theory, IEEE Int. Symp. on*, Jun. 2003, p. 442.

[Hua51] L.-K. Hua, "A theorem on matrices over a sfield and its applications," *Acta Mathematica Sinica*, vol. 1, no. 2, pp. 109–163, 1951.

[KK08] R. Koetter and F. Kschischang, "Coding for Errors and Erasures in Random Network Coding," *Inform. Theory, IEEE Trans. on*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.

[KL97] V. Y. Krachkovsky and Y. X. Lee, "Decoding for Interleaved Reed-Solomon Schemes," *Trans. Magn.*, vol. 33, pp. 2740–2743, September 1997.

[KO63] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *Soviet Physics–Doklady*, vol. 7, pp. 595–596, 1963.

[Kra03] V. Y. Krachkovsky, "Reed-Solomon Codes for Correcting Phased Error Bursts," *Inform. Theory, IEEE Trans. on*, vol. 49, no. 11, pp. 2975–2984, Nov. 2003.

[Len83] J. Lenstra, H. W., "Integer Programming with a Fixed Number of Variables," *Mathematics of Operations Research*, vol. 8, no. 4, pp. 538–548, Nov. 1983.

[Lip81] J. D. Lipson, *Elements of Algebra and Algebraic Computing*. Addison-Wesley, Jan. 1981.

[LLL82] A. K. Lenstra, J. Lenstra, H. W., and L. Lovász, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[LN83] R. Lidl and H. Niederreiter, *Finite Fields*, ser. Encyclopedia of mathematics and its applications. Addison-Wesley, Advanced Book Program/World Science Division, 1983.

[LO06] P. Loidreau and R. Overbeck, "Decoding Rank Errors beyond the Error-Correction Capability," in *Proc. the 10th Int. workshop on algebraic and combinatorial coding theory (ACCT)*, Sep. 2006.

[Loi06] P. Loidreau, "A Welch-Berlekamp Like Algorithm for Decoding Gabidulin Codes," in *Coding and Cryptography*, ser. Lecture Notes in Computer Science (LNCS). Springer Berlin Heidelberg, 2006, vol. 3969, pp. 36–45.

[Man76] D. Mandelbaum, "On a Class of Arithmetic Codes and a Decoding Algorithm (Corresp.)," *IEEE Trans. on Inform. Theory*, vol. 22, no. 1, pp. 85–88, Jan. 1976.

[Man78] ——, "Further Results on Decoding Arithmetic Residue Codes (Corresp.)," *Inform. Theory, IEEE Trans. on*, vol. 24, no. 5, pp. 643–644, Sep. 1978.

[Mas69] J. Massey, "Shift-Register Synthesis and BCH Decoding," *Inform. Theory, IEEE Trans. on*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[Moo05] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, Jun. 2005.

[MS77] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North Holland Publishing Co., 1977.

[Nie13a] J. S. R. Nielsen, "Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation," in *Inform. Theory, IEEE Int. Symp. on*, Jul. 2013, pp. 882–886.

[Nie13b] ——, "Generalised Multi-sequence Shift-Register Synthesis using Module Minimisation," in *Inform. Theory, IEEE Int. Symp. on*, 2013.

[NS06] P. Q. Nguyen and D. Stehlé, "LLL on the Average," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4076, pp. 238–256.

[OP07] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts)*. Imperial College Press, Sep. 2007.

[Ore33] O. Ore, "Theory of Non-Commutative Polynomials," *Annals of Mathematics*, vol. 34, no. 3, pp. 480–508, 1933.

[Ove07] R. Overbeck, "Public Key Cryptography based on Coding Theory," PhD dissertation, Universität Darmstadt, 2007.

[OY14] F. Özbudak and O. Yayla, "Improved Probabilistic Decoding of Interleaved Reed-Solomon Codes and Folded Hermitian Codes," *Theoretical Computer Science*, vol. 520, pp. 111–123, Feb. 2014.

[Pet60] W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes," *Inform. Theory, IRE Trans. on*, vol. 6, no. 4, pp. 459–470, Sep. 1960.

[PT91] A. V. Paramonov and O. V. Tretjakov, "An Analogue of Berlekamp-Massey Algorithm for Decoding Codes in Rank Metric," in *Proc. of Moscow Institute for Physics and Technology (MIPT) (in Russian)*, 1991.

[PW72] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes - Revised*, 2nd ed. The MIT Press, Mar. 1972.

[Ren04] J. Ren, "On the Structure of Hermitian Codes and Decoding for Burst Errors," *Inform. Theory, IEEE Trans. on*, vol. 50, no. 11, pp. 2850–2854, Nov. 2004.

[Rot91] R. Roth, "Maximum-Rank Array Codes and their Application to Crisscross Error Correction," *Inform. Theory, IEEE Trans. on*, vol. 37, no. 2, pp. 328–336, Mar. 1991.

[RP04] G. Richter and S. Plass, "Fast Decoding of Rank-Codes with Rank Errors and Column Erasures," in *Inform. Theory, IEEE Int. Symp. on*, Jun. 2004, pp. 398–398.

[RS60] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[SB14] V. Sidorenko and M. Bossert, "Fast skew-feedback shift-register synthesis," *Designs, Codes and Cryptography*, vol. 70, no. 1-2, pp. 55–67, 2014.

[Sha48] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[Sha79] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[Sin64] R. C. Singleton, "Maximum Distance Q-Nary Codes," *Inform. Theory, IEEE Trans. on*, vol. 10, no. 2, pp. 116–118, Apr. 1964.

[SJB11] V. Sidorenko, L. Jiang, and M. Bossert, "Skew-Feedback Shift-Register Synthesis and Decoding Interleaved Gabidulin Codes," *Inform. Theory, IEEE Trans. on*, vol. 57, no. 2, pp. 621–632, Feb. 2011.

[SJJT86] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor, Eds., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. Piscataway, NJ, USA: IEEE Press, 1986.

[SJM+95] S. Sakata, J. Justesen, Y. Madelung, H. E. Jensen, and T. Hoholdt, "Fast Decoding of Algebraic-Geometric Codes up to the Designed Minimum Distance," *Inform. Theory, IEEE Trans. on*, vol. 41, no. 6, pp. 1672–1677, Nov. 1995.

[SK09] D. Silva and F. Kschischang, "Fast encoding and decoding of Gabidulin codes," in *Inform. Theory, IEEE Int. Symp. on*, Jun. 2009, pp. 2858–2862.

[SKHN75] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *Inform. and Control*, vol. 27, no. 1, pp. 87–99, 1975.

[SKK08] D. Silva, F. Kschischang, and R. Koetter, "A Rank-Metric Approach to Error Control in Random Network Coding," *Inform. Theory, IEEE Trans. on*, vol. 54, no. 9, pp. 3951–3967, Sep. 2008.

[SRB11] V. Sidorenko, G. Richter, and M. Bossert, "Linearized Shift-Register Synthesis," *Inform. Theory, IEEE Trans. on*, vol. 57, no. 9, pp. 6025–6032, Sep. 2011.

[SS71] A. Schönhage and V. Strassen, "Schnelle Multiplikation großer Zahlen," *Computing*, vol. 7, no. 3-4, pp. 281–292, 1971.

[SS11] V. Sidorenko and G. Schmidt, "A Linear Algebraic Approach to Multisequence Shift-Register Synthesis," *Probl. of Inform. Transm.*, vol. 47, no. 2, pp. 149–165, Jun. 2011.

[SSB07] G. Schmidt, V. Sidorenko, and M. Bossert, "Enhancing the Correcting Radius of Interleaved Reed-Solomon Decoding using Syndrome Extension Techniques," in *Inform. Theory, IEEE Int. Symp. on*. IEEE, Jun. 2007, pp. 1341–1345.

[SSB09] ——, "Collaborative Decoding of Interleaved Reed-Solomon Codes and Concatenated Code Designs," *Inform. Theory, IEEE Trans. on*, vol. 55, no. 7, pp. 2991–3012, 2009.

[SSB10] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Syndrome Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance Based on Shift-Register Synthesis," *Inform. Theory, IEEE Trans. on*, vol. 56, no. 10, pp. 5245–5252, Oct. 2010.

[SSG+05] V. Sidorenko, G. Schmidt, E. Gabidulin, M. Bossert, and V. Afanassiev, "On Polyalphabetic Block Codes," in *Inform. Theory Workshop, IEEE*. IEEE, 2005, pp. 4 pp.–.

[Sti88] H. Stichtenoth, "A Note on Hermitian Codes over GF(q^2)," *Inform. Theory, IEEE Trans. on*, vol. 34, no. 5, pp. 1345–1348, Sep. 1988.

[Sud97] M. Sudan, "Decoding of Reed-Solomon Codes beyond the Error-Correction Bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180–193, Mar. 1997.

[Tie87] H. J. Tiersma, "Remarks on Codes from Hermitian Curves (Corresp.)," *Inform. Theory, IEEE Trans. on*, vol. 33, no. 4, pp. 605–609, Jul. 1987.

[WB86] L. Welch and E. Berlekamp, "Error correction for algebraic block codes," Dec. 30 1986, US Patent 4,633,470.

[WB94] S. B. Wicker and V. Bhargava, *Reed-Solomon Codes and Their Applications*. Piscataway, NJ, USA: IEEE Press, 1994.

[Wu08] Y. Wu, "New List Decoding Algorithms for Reed-Solomon and BCH Codes," *Inform. Theory, IEEE Trans. on*, vol. 54, no. 8, pp. 3611–3630, 2008.

[WZAS13] A. Wachter-Zeh, V. Afanassiev, and V. Sidorenko, "Fast decoding of Gabidulin codes," *Designs, Codes and Cryptography*, vol. 66, no. 1-3, pp. 57–73, 2013.

[WZZB12] A. Wachter-Zeh, A. Zeh, and M. Bossert, "Decoding interleaved Reed-Solomon codes beyond their joint error-correcting capability," *Designs, Codes and Cryptography*, pp. 1–21, Jul. 2012.

[YB92] T. Yaghoobian and I. Blake, "Hermitian codes as generalized Reed-Solomon codes," *Designs, Codes and Cryptography*, vol. 2, no. 1, pp. 5–17, Mar. 1992.

[ZW11] A. Zeh and A. Wachter, "Fast Multi-Sequence Shift-Register Synthesis with the Euclidean Algorithm," *Advances in Mathematics of Commun.*, vol. 5, no. 4, pp. 667–680, Nov. 2011.

Publications containing parts of this thesis:

[KL13] S. Kampf and W. Li, "Decoding Interleaved Reed-Solomon and Hermitian Codes with Generalized Divisions," in *Systems, Commun. and Coding (SCC), Proc. 9th Int. ITG Conf. on.* VDE, Jan. 2013, pp. 1–6.

[Li12] W. Li, "On Syndrome Decoding of Chinese Remainder Codes," in *The 13th Int. Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, Jun. 2012.

[LNS14] W. Li, J. S. R. Nielsen, and V. Sidorenko, "On Decoding of Interleaved Chinese Remainder Codes (extended abstract)," in *The 21st Int. Symp. on Mathematical Theory of Networks and Systems (MTNS)*, Jul. 2014.

[LS12] W. Li and V. Sidorenko, "On the Error-Erasure-Decoder of the Chinese Remainder Codes," in *Probl. of Redundancy in Inform. and Control Systems, XIII Int. Symp. on.* IEEE, 2012, pp. 37–40.

[LSC13] W. Li, V. Sidorenko, and D. Chen, "On Transform-Domain Decoding of Gabidulin Codes," in *Int. Workshop Coding Cryptogrphy (WCC)*, Apr. 2013, pp. 33–56.

[LSN13] W. Li, V. Sidorenko, and J. S. R. Nielsen, "On Decoding Interleaved Chinese Remainder Codes," in *Inform. Theory, IEEE Int. Symp. on.* IEEE, Jul. 2013, pp. 1052–1056.

[LSS14] W. Li, V. Sidorenko, and D. Silva, "On transform-domain error and erasure correction by Gabidulin codes," *Designs, Codes and Cryptography*, vol. 73, no. 2, pp. 571–586, 2014.

[LSW14] W. Li, V. Sidorenko, and X. Wang, "Efficient Burst Error Correction by Hermitian Codes," in *The 21st Int. Symp. on Mathematical Theory of Networks and Systems (MTNS)*, Jul. 2014, pp. 354–361.

[ZL10a] A. Zeh and W. Li, "Decoding Reed-Solomon Codes up to the Sudan Radius with the Euclidean Algorithm," in *Inform. Theory and its Applications, Int. Symp. on (ISITA)*, Oct. 2010.

[ZL10b] ——, "On Reformulated Multi-Sequence Problems," in *The 12th Int. Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, Sep. 2010.

Wenhui Li | Curriculum Vitae

Personal Data

Date of Birth: 21.08.1984, Jinan, China

Nationality: Chinese

Career

Institute of Communications Engineering, Ulm University — **Ulm, Germany**
Teaching assistant (doctoral candidate) — *08.2010 – 10.2014*

Education

Ulm Univesity — **Ulm, Germany**
Degree "Master of Science" (M.Sc.) obtained — *18.02.2011*

Ulm University — **Ulm, Germany**
Master student — *03.2008 – 06.2010*
majoring Communications Technology

Shandong Univesity — **Jinan, China**
Degree "Bachelor of Engineering"(B.Eng.) obtained — *01.07.2007*

Shandong University — **Jinan, China**
Bachelor student — *09.2003 – 06.2007*
majoring Communications Engineering

High school attached to Shandong Normal University — **Jinan, China**
09.2000 – 06.2003

Middle school attached to Shandong University — **Jinan, China**
09.1997 – 06.2000

Hong Jia Lou primary school — **Jinan, China**
09.1991 – 06.1997

Experience

Exercise tutor
in course "Channel Coding" — *WS 2012/13, WS 2013/14*
Institute of Communications Engineering, Ulm University

Exercise tutor
in course"Multiuser Communications and MIMO Systems" — *SS 2012*
Institute of Communications Engineering, Ulm University

Lab Supervisor
in lab "Communications Technology" — *WS 2010/2011, SS 2011, WS 2011/2012, SS 2013*
Institute of Communications Engineering, Ulm University

Publication

Journal Article

[1] W. Li, V. Sidorenko, and D. Silva, **On transform-domain error and erasure correction by Gabidulin codes**, in *Designs, Codes and Cryptography*, vol. 73, no. 2, pp. 571-586, 2014.

Conference Papers

[2] W. Li, J. S. R. Nielsen, and V. Sidorenko, **On Decoding of interleaved Chinese remainder codes** (extended abstract), in *the 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Jul. 2014.

[3] W. Li, X. Wang, and V. Sidorenko, **Efficient burst error correction by Hermitian codes**, in *the 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Jul. 2014, pp. 354–361.

[4] W. Li, V. Sidorenko, and J. S. R. Nielsen, **On decoding interleaved Chinese remainder codes**, in *IEEE International Symposium on Information Theory (ISIT)*, IEEE, Jul. 2013, pp. 1052–1056.

[5] W. Li, V. Sidorenko, and D. Chen, **On transform-domain decoding of Gabidulin codes**, in *International Workshop on Coding and Cryptography (WCC)*, Apr. 2013, pp. 33–56.

[6] S. Kampf and W. Li, **Decoding interleaved Reed–Solomon and Hermitian codes with generalized divisions**, in *the 9th International ITG Conference on Systems, Communications and Coding (SCC)*, VDE, Jan. 2013, pp. 1–6.

[7] W. Li and V. Sidorenko, **On the error-erasure-decoder of the Chinese remainder codes**, in *the XIII International Symposium "Problems of Redundancy in Information and Control Systems*, IEEE, 2012, pp. 37–40.

[8] W. Li, **On syndrome decoding of Chinese remainder codes**, in *the 13th International Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, Jun. 2012.

[9] A. Zeh and W. Li, **Decoding Reed–Solomon codes up to the Sudan radius with the Euclidean algorithm**, In *International Symposium on Information Theory and its Applications (ISITA)*, Oct. 2010, pp. 986–990.

[10] A. Zeh and W. Li, **On reformulated multi-sequence problems**, in *the 12th International Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, Sep. 2010.