Technische Universität München
Coding for Communications and Data Storage
Prof. Dr.-Ing. Antonia Wachter-Zeh

Master's Thesis

Decoding of Interleaved Goppa Codes and Their Applications in Code-based Cryptosystem

Vorgelegt von:
Hedongliang Liu

München, Dec. 2018

Betreut von:
M.Sc. Lukas Holzbaur
Dr.-Ing. Sven Puchinger

Master's Thesis am
Coding for Communications and Data Storage (COD)
der Technischen Universität München (TUM)
Titel : Decoding of Interleaved Goppa Codes and Their Applications in
Code-based Cryptosystem
Autor : Hedongliang Liu

Hedongliang Liu
Adelheidstr. 15
80798 München
lia.liu@tum.de

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

München, 12.12.2018
....

Ort, Datum (Hedongliang Liu)

Contents

List of Figures

List of Algorithms

Notations

r	Degree of Goppa polynomial
$q\|r$	q divides r
$\mathbb{F}_q$	Finite field of size q
$\mathbf{a} \in \mathbb{F}_q^n$	A vector of length n over $\mathbb{F}_q$
$\mathrm{wt}(\mathbf{a})$	Hamming weight of vector $\mathbf{a}$, i.e., number of non-zeros entries of $\mathbf{a}$
$\mathbf{a}_{\mathcal{I}}$	A vector composed of the positions of $\mathbf{a}$ indexed by the indexes in set $\mathcal{I}$
$\mathrm{supp}(\mathbf{a})$	A set of indexes of non-zero positions of vector $\mathbf{a}$
$\mathbf{A} \in \mathbb{F}_q^{a \times b}$	A matrix with a rows and b columns over $\mathbb{F}_q$
$\mathrm{wt}(\mathbf{A})$	Number of non-zero columns of $\mathbf{A}$
$\mathbf{A}_{\mathcal{I}}$	A matrix composed of the columns of $\mathbf{A}$ indexed by the indexes in set $\mathcal{I}$
$\mathcal{C}[n,k,d]_q$	A linear code $\mathcal{C}$ of length n, dimension k and minimum distance d over $\mathbb{F}_q$
$\mathcal{GRS}(\mathcal{L},\mathcal{V})$	A generalized Reed-Solomon code with code locator set $\mathcal{L}$ and column multipler set $\mathcal{V}$
$\mathrm{supp}(\mathcal{C})$	A set of indexes of non-zero positions of all codewords $\mathbf{c} \in \mathcal{C}$
$\Gamma(\mathcal{L},g)$	A Goppa code defined by code locator set $\mathcal{L}$ and Goppa polynomial g
$\mathcal{I}\Gamma(\ell,\mathcal{L},g)$	An ℓ-interleaved Goppa code
t_{Int}	Error correction capability of an ℓ-interleaved Goppa code

Abbreviations

BMD	Bounded Minimum Distance
EEA	Extended Euclidean Algorithm
EEP	Error Evaluator Polynomial
ELP	Error Locator Polynomial
FLWCW	Finding Low-Weight CodeWords
IGC	Interleaved Goppa Codes
IRS	Interleaved Reed-Solomon
ISD	Information Set Decoding
LHS	Left-Hand Side
LSE	Linear System of Equations
RHS	Right-Hand Side
RS	Reed-Solomon
RSA	Rivest-Shamir-Adleman, a public-key cryptosystem name after is inventors
SL	Security Level
WF	Work Factor

Abstract

Post-quantum cryptography is of high interest in recent years since some algorithms based on quantum computing already threaten the security of classical public-key cryptosystems, such as RSA. From a long-term security point of view, cryptosystems should stay secure against quantum-computing attacks. Code-based cryptosystems are among the promising candidates. The McEliece system with Goppa codes, which was proposed in 1970s, is the most famous code-based cryptosystem and still stays secure today. The structure of Goppa codes can be well-hidden in McEliece system so that attacking a McEliece system with Goppa codes is similar to the problem of doing nearest distance decoding a generic linear codes, which is NP-complete.

In this thesis, we investigate the minimum distance, wild property and decoding algorithms of Goppa codes; we present a *collaborative decoding* for *interleaved Goppa codes* (IGC). The collaborative decoding increases the decoding radius beyond half of the minimum distance. We integrate the wild property into IGC and show that we can collaboratively correct more errors for binary Goppa code than the Patterson decoder, which was known as the best decoder for binary Goppa codes. We revisit the McEliece system and investigate some well-known existing attacks. A new variant of the McEliece system, *IntMcEliece*, using wild IGC, is proposed as another output of this thesis. Some instances of parameter selection and key size are presented at the end of this thesis.

Acknowledgements

I would like to express my sincere appreciation to my supervisors, Lukas Holzbaur and Sven Puchinger, for their insightful and excellent guidance, their patience, their continuous support throughout all stages of my Master's thesis and their detailed comments on the drafts of this thesis. I am very grateful for the precious time they have spent for me. We have had about 20 meetings in person or via Skype about my thesis since I started this May and I didn't count all the scattered small conversations and the occassions when I just visited their offices by chance with some minor problems that occured along the way. Not only their patience and engagement, but also their passion for research and their innovative ideas, motivated me to devote myself more to my thesis. Without their guidance and help throughout the research and writing process, I would not have been able to reach these results and gain so much from this experience. Special thanks go to Lukas for offering me this topic that combines algebraic coding and cryptography - it does interest me a lot and has extended the area of my study.

I would also like to express my sincere gratitude to Prof. Antonia Wachter-Zeh who introduced me to the field of algebraic coding. I learned the concept of "finite fields" for the first time in her lecture *Channel Coding.* Had it not been for her great teaching, I would not have applied to do my research internship and Master's thesis both in this field. I would further like to thank her for her careful concerns of the progress of my thesis, for her encouragement and constructive advice when I encountered some difficulties and for the finantial support for me to go to the summer school in Turku, Finnland and the ACCT conference in Kaliningrad, Russia, which made these 7 months an even more unforgettable memory.

I want to thank Sergey Bezzateev and Vladimir Sidorenko for the fruitful discussions at the early stages of my thesis.

My sincere gratitude also goes to all the colleagues working on the 2nd to 4th floors in the building N4. They have organized a lot of valuable workshops, seminars and fantastic extra curricular activities, which immensely enriched my Master's experience at TUM.

Special thanks go to Lukas, Sven, Julian Renner, Andreas Lenz and Patrick Schulte for talking to me in German patiently from time to time to help me practice. *Dafür bin ich euch zutiefst dankbar!*

I started working in the student lab N2406 since last October and have spent more than half of my Master's time there. I want to say thanks to the students whom I have met in that room and shared interesting conversations with during my stay, especially to Wafa Labidi and Thomas Wiegart, who have listened to me a lot and given me lots of encouragement and help throughout the year.

Many thanks to Lisa, Elfie, Shiying, Keyue, Hanbo, Yining and all my friends in distance or in neighbour, for their kindness, friendship and understanding. Thank you for filling up my life with joy and pleasure all the time.

Last but definitely not least, I want to express my gratitude, even though words cannot express all of it, to my mother who unconditionally loves me and supports me, and to my father who has always been enlightening and inspiring me in my heart in every decision that I have made for myself.

1 Introduction

1.1 Cryptosystem

Cryptography or *cryptology* is a prolonged practice and study of how to achieve *secure communication* without any outside sources interfering, reading, or altering the transmitted message. The security goals include *data confidentiality, integrity, authentication* and *non-repudiation.* We focus on data confidentiality in this thesis. *Cryptosystems* are methods or protocols used by two parties, often referred as **Alice** and **Bob** (which in real life are typically digital devices, e.g., computers, mobile phones, smart cards, etc.), that prevent a third party **Eve** from reading their private message.
The cryptosystems can be classified into *symmetric* cryptosystems and *asymmetric* cryptosystems according to whether Alice and Bob share the same secret key.
The widely-used symmetric cryptosystems nowadays are DES, AES, etc.
In asymmetric cryptosystems, also known as *public-key* cryptosystems, each party has two keys: a *public key* and a *private key.* Suppose Alice publishes her public key so that everyone can access it in order to encrypt a secret message then send to her. Only Alice who possesses the secret key can decrypt the ciphertext to the secret message. A public-key system should be constructed such that the calculation of the secret key is computationally infeasible from the public key. The secrecy of the private key usually relies on certain intractable mathematical problems. For instance, the well-known RSA cryptosystem [RSA78] is based on *the large prime factorization problem.* The *Diffie-Hellman key exchange protocol* [Mer78] and *ElGamal encryption* [Elg85] are based on *the discrete logarithm problem.* Moreover, the *Elliptic-curve cryptography* relies on *the inability to compute the multiplicand points given the origin and product on an elliptic curve.*

Post-Quantum Cryptosystems

In recent years, quantum computing [Ben80] and the concept of quantum computers [Deu85] lead technology into a new era of machines' computational ability. Though the development of quantum computers is still in its infancy, both practical and theoretical research on quantum computing is prosperous. Some algorithms based on quantum

computing can potentially break most conventional cryptosystems deployed in practice nowadays. For instance, Shor's algorithms [Sho94] are quantum algorithms that solve prime factorization and discrete logarithm problems in polynomial-time. Shor introduced a similar algorithm which can quickly find the multiplicands on an elliptic curve. All the previously mentioned public-key cryptosystems are therefore broken by Shor's algorithms. Not only asymmetric, but symmetric cryptosystems also become "victims" of quantum computing. Grover's quantum mechanical algorithm [Gro96] can find a certain item in a size-N database by using $\sqrt{N}$ quantum queries. This reduces the security level of AES and SHA by a factor of 2 [BL17].
Facing the threat of Shor's and Grover's quantum computing algorithms, *post-quantum cryptosystems* should be deployed before large-scale quantum computers arrive. The National Institute of Standards and Technology (NIST) addressed to the public and announced the launch of a Post-Quantum Cryptography (PQC) bidder competition, which is scheduled for adoption in 2020 - 2022 [Moo17]. The submission phase finished in November, 2017. Works within the competition are conducted mainly in 5 different directions:

- Code-based Cryptography;
- Lattice-based Cryptography;
- Multivariate Cryptography;
- Hash-based Signatures;
- Isogeny-based Signatures.

Thereinto, roughly 1/4 studies were conducted in the area of code-based cryptography [KKL$^+$18]. Quantum computers do not seem to give any significant improvements in attacking code-based systems.

1.2 Related Works

The first code-based public-key cryptosystem was introduced in 1978 by McEliece [JM78]. The public key specifies a random binary Goppa code. The system provides faster encryption and decryption schemes because of efficient encoding and decoding methods of Goppa codes. However, the key size is much larger (e.g. several hundred KB for 128 bits sevurity level) than classical public-key cryptosystems (e.g. RSA needs $<$ 1 KB for 128 bits security level), which is a severe drawback of the McEliece system, but in turn motivates a lot of research on reducing the key size. Some works proposed to use other

classes of codes as alternatives to Goppa codes, such as Reed-Solomon (RS) codes (broken [MSOS92]), quasi-dyadic (QD) Goppa codes [MB09] (broken [Per12]), quasi-cyclic (QC)-LDPC/MDPC codes [MTSB13] (broken [DK18]). Unbroken proposals with smaller key size include McEliece system using twisted RS codes [BBPR18], *Gabidulin-Paramanov-Tretjakov* (GPT) system [GPT91] using rank-metric codes or twisted rank-metric codes [PRWZ18] and a repair [WZPR18] of the *Faure-Loidreau Public-Key System* [FL06]. A new variant of the McEliece system, *"Interleaved McEliece"*, with *interleaved Goppa codes* (IGC) was proposed by Elleuch, Wachter-Zeh and Zeh in [EWZZ18]. The new variant increases the error correction capability, however faces a decoding attack that may severely reduce its security.
The McEliece system based on Goppa codes is still unbroken after hundreds of attacking and defending researches over 40 years.
Goppa codes [Gop70, Gop71] form a large class of algebraic error-correcting codes, named after their inventor, Valery Denisovich Goppa. E. Berlekamp summarized the properties of Goppa codes in [Ber73], before the original papers were translated into English.
Goppa codes are described in terms of a *Goppa polynomial* $g(x)$ of degree r and a *code locator set* $\mathcal{L}$. Goppa codes form a subclass of *alternant codes* [MS78, Ch. 12], which are subfield subcodes of generalized RS codes [RS60]. Therefore, a Goppa code of length n over $\mathbb{F}_q$ contains the codewords, which are elements in $\mathbb{F}_q^n$, of the corresponding RS code of length n over $\mathbb{F}_{q^m}$. This gives an essential fact that decoders of RS codes can be used to decode Goppa codes. Irreducible Goppa codes, where $g(x)$ is an irreducible polynomial, are considered exceptionally good codes in that they meet the Gilbert-Varshamov bound.
Decoding algorithms of Goppa codes have been persistently investigated from 1970s until today. Sugiyama et al. showed an algebraic decoding algorithm of solving the key equations by Euclid's algorithm in [SKHN75]. Patterson introduced a new algorithm in [Pat75] with an extra "key equation degree reduction" step. This algorithm decodes up to r errors for binary Goppa codes, while syndrome-based decoders can only decode up to $\left\lfloor \frac{r}{2} \right\rfloor$ errors. Barreto et al. [BML13] came up with a probabilistic algorithm which generalized Patterson's algorithm over any prime field $\mathbb{F}_p$ to increase the decoding radius of Goppa codes from $\left\lfloor \frac{r}{2} \right\rfloor$ to $\left\lfloor \frac{2}{p} r \right\rfloor$. Moreover, several list decoding approaches [ABC11, Ber11, BHNW13] were proposed in order to decode at the radius over the half of the minimum distance.

1.3 Contributions and Outline

There are three main contributions of this thesis:

- a probabilistic decoding algorithm for square-free *interleaved Goppa codes* (IGC) that can decode over half of the minimum distance. For binary square-free Goppa codes, the proposed decoding algorithm can correct more errors than Patterson's decoder.

- implemention of a `GoppaCode` class and several decoder classes for Goppa codes in *SageMath*, which may be useful for further researches on Goppa codes.
- propose *IntMcEliece* as a repair of the "Interleaved McEliece" [EWZZ18]. The repair effectively defends the system against Stern's attack, which may dramatically reduce the security level of the "Interleaved McEliece"..

This thesis is organized as follows.
In **Chapter 2** we first recall the definition of Goppa codes and discuss two special classes of Goppa codes and their properties. Then we review some known *bounded minimum distance* (BMD) decoding algorithms for Goppa codes, including RS syndrome-based decoder, Patterson decoder for binary square-free Goppa codes and p-ary Patterson decoder for non-binary square-free Goppa codes.
In **Chapter 3** we recap the structure of the McEliece system and *Wild McEliece*. Several attacks that decrease the security level of McEliece system are described as well.
In **Chapter 4** we give the definiton of interleaved Goppa codes and introduce a collaborative decoding algorithm that can decode beyond half of the minimum distance. The algorithm is probabilistic. We investigate the failure probability via simulations and compare the results with the bound on failure probability of decoding interleaved Reed-Solomon codes. At the end of this chapter, we present a `GoppaCode` class and several decoder classes in *SageMath* that are implenmented by the author.
In **Chapter 5** we introduce *IntMcEliece*, which is a repair of the Interleaved McEliece from [EWZZ18]. This repair increase the error correction capability comparing to the previous system by integrating the wild Goppa codes and defends itself against the *finding low-weight codewords attack* by an "error encoding scheme" in encryption. The advantage of IntMcEliece is shown by comparing its key size with that of wild McEliece at 80, 128, 256 bits security.

2 Goppa Codes

In this chapter, we first recall the definition of Goppa codes and show two special Goppa codes classes and their properties. Then we review some known BMD decoding algorithms for Goppa codes.

2.1 Definition

Definition 2.1.1. *Let q be a prime power and m, n, r be some integers such that $rm \leq n \leq q^m$. Fix a code locator set $\mathcal{L} = \{\alpha_0, \dots, \alpha_{n-1}\}$ containing n distinct elements from $\mathbb{F}_{q^m}$ and a Goppa polynimial $g(x) \in \mathbb{F}_{q^m}[x]$ of degree r such that $g(\alpha_i) \neq 0, \forall \alpha_i \in \mathcal{L}$. Denote a rational function*

$$R_{\mathbf{c}}(x) = \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \tag{2.1}$$

with any vector $\mathbf{c} = (c_0, \dots, c_{n-1})$.

A Goppa code $\Gamma(\mathcal{L}, g)$ is defined by

$$\Gamma(\mathcal{L}, g) = \left\{ \ \mathbf{c} \mid R_{\mathbf{c}}(x) \equiv 0 \mod g(x), \quad \forall \mathbf{c} \in \mathbb{F}_q^n \right\}. \tag{2.2}$$

If $g(x)$ has no multiple irreducible factors then $\Gamma(\mathcal{L}, g)$ is called a *square-free* or *separable* Goppa code. In addition, if $g(x)$ is an irreducible polynomial then $\Gamma(\mathcal{L}, g)$ is called an *irreducible* Goppa code.

A linear code can be also represented by its parity check matrix. In the following theorem we show the parity check matrix of a Goppa code.

Theorem 2.1.1 (Parity Check Matrix over $\mathbb{F}_{q^m}$)**.** *The parity check matrix of a Goppa* *code* $\Gamma(\mathcal{L}, g)$ *is*

$$\mathbf{H} = \mathbf{CXY}$$

$$= \begin{bmatrix} g_r & & & \\ g_{r-1} & g_r & & \\ \vdots & \vdots & \ddots & \\ g_1 & g_2 & \cdots & g_r \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{bmatrix} \begin{bmatrix} g(\alpha_1)^{-1} & & & \\ & g(\alpha_2)^{-1} & & \\ & & \ddots & \\ & & & g(\alpha_n)^{-1} \end{bmatrix}. \tag{2.3}$$

$\tilde{\mathbf{H}} = \mathbf{XY}$ *is an equivalent parity check matrices of* $\Gamma(\mathcal{L}, g)$.

Proof. The inverse of $(x - \alpha_i)$ exists in $\mathbb{F}_{q^m}/g(x)$ (since α_i is not a root of $g(x)$, so that $\gcd(x - \alpha_i, g(x)) = 1$), which is

$$(x - \alpha_i)^{-1} \equiv -\frac{g(x) - g(\alpha_i)}{x - \alpha_i} g(\alpha_i)^{-1} \mod g(x),$$

since

$$(x - \alpha_i)^{-1}(x - \alpha_i) + (g(x) - g(\alpha_i))g(\alpha_i)^{-1} \equiv 1 \mod g(x).$$

Therefore $\mathbf{c}$ is a codeword of $\Gamma(\mathcal{L}, g)$ iff

$$\sum_{i=0}^{n-1} c_i \frac{g(x) - g(\alpha_i)}{x - \alpha_i} g(\alpha_i)^{-1} = 0 \tag{2.4}$$

as a polynomial of x (without $\mod g(x)$). We can write $g(x) = \sum_{j=0}^{r} g_j x^j$, with $g_j \in \mathbb{F}_{q^m}$ and $g_r \neq 0$. Then

$$\begin{aligned} \frac{g(x) - g(\alpha_i)}{x - \alpha_i} &= \frac{\sum_{j=0}^{r} g_j x^j - \sum_{j=0}^{r} g_j \alpha_i^j}{x - \alpha_i} \\ &= \frac{g_r(x^r - \alpha_i^r) + \cdots + g_1(x - \alpha_i)}{x - \alpha_i} \\ &= g_r(x^{r-1} + x^{r-2}\alpha_i + \cdots + \alpha_i^{r-1}) + \cdots + g_2(x + \alpha_i) + g_1 \\ &= g_r x^{r-1} + (g_{r-1} + \alpha_i g_r)x^{r-2} + \cdots + (g_1 + \cdots + g_r \alpha_i^{r-1}) \end{aligned}$$

Equating the coefficients of $x^{r-1}, \ldots, 1$ to zero in (2.4) we can see that $\mathbf{c}$ is a codeword of $\Gamma(\mathcal{L}, g)$ iff $\mathbf{c}\mathbf{H}^T = \mathbf{0}$, where

$$\begin{aligned}\mathbf{H} &= \begin{bmatrix} g_r g(\alpha_1)^{-1} & \cdots & g_r g(\alpha_n)^{-1} \\ (g_{r-1} + \alpha_1 g_r) g(\alpha_1)^{-1} & \cdots & (g_{r-1} + \alpha_n g_r) g(\alpha_n)^{-1} \\ \vdots & \ddots & \vdots \\ (g_1 + \cdots + g_r \alpha_1^{r-1}) g(\alpha_1)^{-1} & \cdots & (g_1 + \cdots + g_r \alpha_n^{r-1}) g(\alpha_n)^{-1} \end{bmatrix} \\ &= \mathbf{CXY}.\end{aligned}$$

Since $\mathbf{C}$ is a lower-triangular matrix with non-vanishing diagonal elements, it is invertible. Thus $\mathbf{H} = \mathbf{CXY}$ and $\tilde{\mathbf{H}} = \mathbf{XY}$ both define the same $\Gamma(\mathcal{L}, g)$. □

Parity Check Matrix over $\mathbb{F}_q$ Goppa codes are codes over $\mathbb{F}_q$. A parity check matrix $\mathbf{H}_{\text{Goppa}}$ with elements from $\mathbb{F}_q$ is obtained by replacing each entry (an element from $\mathbb{F}_{q^m}$) of $\mathbf{H}$ or $\tilde{\mathbf{H}}$ by its vector representation over $\mathbb{F}_q$.

Example 1 (Obtaining $\mathbf{H}_{\text{Goppa}}$ from $\mathbf{H}$)**.** *Consider a binary Goppa code, i.e. $q = 2$, of length $n = 8$ and $m = 3$. The parity check matrix over $\mathbb{F}_{2^3}$ is given as*

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha^2 & \alpha^4 & \alpha^2 & \alpha & \alpha & \alpha^4 & 1 \\ 0 & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^5 & \alpha^6 & \alpha^3 & 1 \end{bmatrix}, \tag{2.5}$$

where α is a primitive element of $\mathbb{F}_{2^3}$ and the primitive polynomial of $\mathbb{F}_{2^3}$ is $x^3 + x + 1$. Hence a corresponding parity check matrix over $\mathbb{F}_2$ is

$$\mathbf{H}_{\text{Goppa}} = \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hdashline 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{array}\right]. \tag{2.6}$$

2.2 Properties

Theorem 2.2.1. *A Goppa code $\Gamma(\mathcal{L}, g)$ defined by Definition 2.1.1 is a linear code over $\mathbb{F}_q$. The characteristic parameters of a Goppa code $\Gamma(\mathcal{L}, g)$ are:*

- *length $n = |\mathcal{L}|$,*
- *dimension $k \geq n - mr, r = \deg g(x)$,*
- *minimum distance $d \geq r + 1$.*

Proof. Assume two codewords $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$ of $\Gamma(\mathcal{L}, g)$. It holds that $R_{\mathbf{c}^{(i)}}(x) \equiv 0 \mod g(x),\ i = 1, 2$ according to Definition 2.1.1. For any vector $\mathbf{c}' = a\mathbf{c}^{(1)} + b\mathbf{c}^{(2)},\ a, b \in \mathbb{F}_q$,

$$R_{\mathbf{c}'}(x) = \sum_{i=0}^{n-1} \frac{ac_i^{(1)} + bc_i^{(2)}}{x - \alpha_i} = a\sum_{i=0}^{n-1} \frac{c_i^{(1)}}{x - \alpha_i} + b\sum_{i=0}^{n-1} \frac{c_i^{(2)}}{x - \alpha_i} = aR_{\mathbf{c}^{(1)}}(x) + bR_{\mathbf{c}^{(i)}}(x).$$

It follows that $R_{\mathbf{c}'}(x) \equiv 0 \mod g(x)$ and therefore $\mathbf{c}'$ is also a codeword of $\Gamma(\mathcal{L}, g)$.
The length is given by definition.
The redundancy $n - k$ of a code is the number of linearly independent rows of its parity-check matrix $\mathbf{H}_{\text{Goppa}}$. Since $\mathbf{H}_{\text{Goppa}}$ has rm rows, the redundancy $n - k \leq rm$. Thus, the dimension is $k \geq n - rm$.
Suppose $\mathbf{c}$ is a nonzero codeword of $\Gamma(\mathcal{L}, g)$ with weight $\leq r$. Then $\tilde{\mathbf{H}}\mathbf{c}^T = \mathbf{XYc}^T = \mathbf{0}$. Set $\mathbf{b}^T = \mathbf{Yc}^T$ since $\mathbf{Y}$ is diagonal and invertible. Therefore $\mathbf{Xb}^T = \mathbf{0}$, which is impossible since $\mathbf{X}$ is Vandermonde and therefore any r columns are linearly independent. □

Theorem 2.2.2. *A Goppa code $\Gamma(\mathcal{L}, g)$ is the restriction to $\mathbb{F}_q$ of a generalized Reed-Solomon code $\mathcal{GRS}(\mathcal{L}, \mathcal{V})$ of dimension $n - r$, where $\mathcal{L}$ is the set of evaluation points and $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of column multipliers of $\mathcal{GRS}$ with*

$$v_i = \frac{g(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)}, i = 1, \ldots, n. \tag{2.7}$$

Proof. (i) Denote by $\cdot|\mathbb{F}_q$ the restriction to $\mathbb{F}_q$. Take $\mathbf{u} \in \mathcal{GRS}(\mathcal{L}, \mathcal{V})|\mathbb{F}_q$. Then

$$u_i = v_i f(\alpha_i) = \frac{f(\alpha_i)g(\alpha_i)}{\prod_{j \neq i}(\alpha_i - \alpha_j)},$$

where $f(x)$ is a polynomial of degree $< n - r$. Thus

$$\sum_{i=0}^{n-1} \frac{u_i}{x - \alpha_i} = \frac{1}{\prod_{i=0}^{n-1}(x - \alpha_i)} \sum_{i=0}^{n-1} f(\alpha_i)g(\alpha_i) \prod_{j \neq i} \frac{(x - \alpha_j)}{\alpha_i - \alpha_j}.$$

Let

$$N(x) = \sum_{i=0}^{n-1} f(\alpha_i)g(\alpha_i) \prod_{j \neq i} \frac{(x - \alpha_j)}{\alpha_i - \alpha_j}.$$

Then $N(\alpha_i) = f(\alpha_i)g(\alpha_i)$ for $i = 0, \ldots, n-1$. Also $\deg N(x) \leq n - 1$ and $\deg f(x)g(x) \leq n - 1$. Since the polynomial $N(x) - f(x)g(x)$ is determined by its values at n points,

$N(x) = f(x)g(x)$. Therefore

$$\sum_{i=0}^{n-1} \frac{u_i}{x-\alpha_i} = \frac{f(x)g(x)}{\prod_{i=0}^{n-1}(x-\alpha_i)}$$

and hence $\mathbf{u} \in \Gamma(\mathcal{L}, g)$. Thus

$$\Gamma(\mathcal{L}, g) \supset \mathcal{GRS}(\mathcal{L}, \mathcal{V})|\mathbb{F}_q.$$

(ii) The converse is similar. □

Other properties: there exist long Goppa codes that meet Gilbert-Varshamov bound [MS78, Theorem 3, Ch. 12]; extended Goppa codes are cyclic [BM73, TZ75].

2.3 Special Cases of Goppa Codes

2.3.1 Binary Square-free Goppa Codes

A *binary square-free Goppa code* is defined in Definition 2.1.1 with restrictions that $q = 2$ and $g(x)$ has no multiple irreducible factors.

Theorem 2.3.1. *A binary square-free Goppa code* $\Gamma(L, g)$ *has minimum distance*

$$d \geq 2r + 1,$$

where r *is the degree of its Goppa polynomial* $g(x)$*.*

Proof. We recap the proof from [MS78, Ch. 12, p. 341].
Let $\mathbf{c} = (c_0, \ldots, c_{n-1})$ be a codeword of weight w in $\Gamma(L, g)$, with $c_{p_1} = \cdots = c_{p_w} = 1$ (here $\{p_1, \ldots, p_w\} \subseteq \{0, \ldots, n-1\}$ denote the nonzero positions in the codeword), and define

$$f_{\mathbf{c}}(x) = \prod_{i=1}^{w}(x - \alpha_{p_i}).$$

The formal derivative of $f_{\mathbf{c}}$ is

$$f'_{\mathbf{c}}(x) = \sum_{i=1}^{w}\prod_{j\neq i}(x - \alpha_{p_j})$$

Then $R_{\mathbf{c}}(x)$ from (2.1) becomes

$$R_{\mathbf{c}}(x) = \sum_{i=1}^{w} \frac{1}{x - \alpha_{p_i}} = \frac{\sum_{i=1}^{w} \prod_{j \neq i} (x - \alpha_{p_j})}{\prod_{i=1}^{w} (x - \alpha_{p_i})} = \frac{f'_{\mathbf{c}}(x)}{f_{\mathbf{c}}(x)}. \tag{2.8}$$

Since the α_i are distinct from the definition of Γ, $f_{\mathbf{c}}(x)$ and $f'_{\mathbf{c}}(x)$ have no common factors so that (2.8) is in the simplest form. Since $g(\alpha_i) \neq 0$, $f_{\mathbf{c}}(x)$ and $g(x)$ are relatively prime. So from (2.8)

$$R_{\mathbf{c}}(x) \equiv 0 \mod g(x) \iff g(x) | f'_{\mathbf{c}}(x).$$

For $f_{\mathbf{c}}(x), f'_{\mathbf{c}}(x) \in \mathbb{F}_{2^m}[x]$, $f'_{\mathbf{c}}(x)$ contains only even powers and is a *perfect square* polynomial. For example, let α be a primitive element of $\mathbb{F}_{2^3}$, $f(x) = \alpha^3 x^5 + \alpha^2 x^2 + \alpha^6 x \in \mathbb{F}_{2^3}[x]$, then the formal derivative of $f(x)$ is $f'(x) = \alpha^3 x^4 + \alpha^6 = (\alpha^5 x^2 + \alpha^3)^2$. Let $\bar{g}(x)$ be the lowest degree perfect square which is divisible by $g(x)$. Then

$$g(x) | f'_{\mathbf{c}}(x) \iff \bar{g}(x) | f'_{\mathbf{c}}(x).$$

We can deduce that

$$\begin{aligned} \mathbf{c} \in \Gamma(L, g) &\iff R_{\mathbf{c}}(x) \equiv 0 \mod g(x) \\ &\iff \bar{g}(x) | f'_{\mathbf{c}}(x). \end{aligned}$$

Since Goppa codes are linear codes, the minimum distance

$$d = \min_{\substack{\mathbf{c} \in \Gamma(L,g) \\ \mathbf{c} \neq \mathbf{0}}} \operatorname{wt}(\mathbf{c}) = \min_{\substack{\mathbf{c} \in \Gamma(L,g) \\ \mathbf{c} \neq \mathbf{0}}} \deg f_{\mathbf{c}}(x) = \min_{\substack{\mathbf{c} \in \Gamma(L,g) \\ \mathbf{c} \neq \mathbf{0}}} \deg f'_{\mathbf{c}}(x) + 1.$$

If $\mathbf{c} \neq \mathbf{0}$, $\deg f'_{\mathbf{c}}(x) \geq \deg \bar{g}(x)$. Hence,

$$d \geq \deg \bar{g}(x) + 1.$$

Suppose $g(x)$ is has no multiple irreducible factors, i.e., $\Gamma(L, g)$ is a square-free binary Goppa code, so that $\bar{g}(x) = g(x)^2$. Then the minimum distance d of $\Gamma(L, g)$

$$d \geq 2 \deg g(x) + 1$$

$\square$

2.3.2 Wild Goppa Codes

The *wild Goppa codes* were analyzed in 1976 by Sugiyama, Kasahara, Hirasawa and Namekawa [SKHN76] and suggested to be used in *wild McEliece Cryptosystem* in 2011 by Bernstein, Lange and Peters [BLP11b].

Definition 2.3.1 (Wild Goppa Codes)**.** *Let q be a prime power and m, n be integers such that $n \leq q^m$. Let $\mathcal{L}$ be a set of n distinct elements of $\mathbb{F}_{q^m}$ and $g(x)$ be a monic* ***square-free*** *polynomial in $\mathbb{F}_{q^m}[x]$ such that $g(\alpha) \neq 0, \forall \alpha \in \mathcal{L}$. Goppa codes $\Gamma(\mathcal{L}, g^q)$ and $\Gamma(\mathcal{L}, g^{q-1})$ are called* wild Goppa codes*.*

The following theorem formally states that given a monic square-free polynomial $g(x)$, the Goppa codes $\Gamma(\mathcal{L}, g^q)$ and $\Gamma(\mathcal{L}, g^{q-1})$ over $\mathbb{F}_q$ are the same. The proof of this special property of wild Goppa codes has already appeared in [SKHN76, Wir88] and also been modified in [BLP11b].

Theorem 2.3.2 (Wild Goppa Codes Parameters)**.** *Let $r = \deg(g^{q-1})$. The wild Goppa codes $\Gamma(\mathcal{L}, g^q)$ and $\Gamma(\mathcal{L}, g^{q-1})$ are the same codes. These codes are q-ary codes of*

- *length n,*
- *dimension $k \geq n - rm$, and*
- *minimum distance $d \geq \frac{q}{q-1} r + 1$.*

Proof. Here we recap the proof shown in [BLP11b].
If $\sum\limits_i \frac{c_i}{x-\alpha_i} = 0$ in $\mathbb{F}_{q^m}[x]/g^q$ then certainly $\sum\limits_i \frac{c_i}{x-\alpha_i} = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.
Conversely, consider any $\mathbf{c} \in \mathbb{F}_q^n$ such that $\sum\limits_i \frac{c_i}{x-\alpha_i} = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$. Find an extension $\mathbb{K}$ of $\mathbb{F}_{q^m}$ so that g splits into linear factors in $\mathbb{K}[x]$. Then $\sum\limits_i \frac{c_i}{x-\alpha_i} = 0$ in $\mathbb{K}_{q^m}[x]/g^{q-1}$, so $\sum\limits_i \frac{c_i}{x-\alpha_i} = 0$ in $\mathbb{K}_{q^m}[x]/(x-a)^{q-1}$ for each factor $(x-a)$ of g. The elementary series expansion

$$\frac{1}{x-b_i} = -\frac{1}{(b_i-a)} - \frac{x-a}{(b_i-a)^2} - \frac{(x-a)^2}{(b_i-a)^3} - \cdots$$

then implies

$$\sum_i \frac{c_i}{(b_i-a)} + (x-a)\sum_i \frac{c_i}{(b_i-a)^2} + (x-a)^2 \sum_i \frac{c_i}{(b_i-a)^3} + \cdots = 0$$

in $\mathbb{K}_{q^m}[x]/(x-a)^{q-1}$; i.e., $\sum\limits_i \frac{c_i}{(b_i-a)} = 0$, $\sum\limits_i \frac{c_i}{(b_i-a)^2} = 0$, $\cdots$, $\sum\limits_i \frac{c_i}{(b_i-a)^{q-1}} = 0$. Now take the q-th power of equation $\sum\limits_i \frac{c_i}{(b_i-a)} = 0$, and use the fact that $c_i \in \mathbb{F}_q$ so that $c_i^q = c_i$, to obtain $\sum\limits_i \frac{c_i}{(b_i-a)^q} = 0$. Work backwards to see that $\sum\limits_i \frac{c_i}{(b_i-a)} = 0$ in $\mathbb{K}_{q^m}[x]/(x-a)^q$.

By hypothesis g is the product of its distinct linear factors $(x-a)$. Therefore g^q is the product of the coprime polynomials $(x-a)^q$, and $\sum_i \frac{c_i}{(b_i-a)} = 0$ in $\mathbb{K}_{q^m}[x]/(x-a)^q$; i.e., $\sum_i \frac{c_i}{(b_i-a)} = 0$ in $\mathbb{F}_{q^m}[x]/(x-a)^q$.
The parameters follows from Theorem 2.2.1. □

2.4 Bounded Minimum Distance Decoders of Goppa Codes

Bounded minimum distance (BMD) decoders can decode up to half of the minimum distance d. Such decoders guarantee to give a unique codeword when the number of errors is $t \leq \lfloor \frac{d-1}{2} \rfloor$. Goppa codes are linear codes. Syndrome decoding is an efficient method to decode linear codes because of their linear structures. We mainly discuss the syndrome-based decoders for Goppa codes.

2.4.1 RS Syndrome-based Decoder

As stated in Theorem 2.2.2, Goppa codes are subfield subcodes of generalized RS codes. We therefore can use the decoders of RS codes to decode Goppa codes.
Uniquely decoding RS codes was initialized by [Pet60, GZ61] and involves works from [Chi64, Ber68, Mas69, For66, SKHN75]. A detailed overview of their contributions can be found in [Czy11, Ch. 4].

Definition 2.4.1 (RS Syndrome)**.** *Given a $\mathcal{GRS}(\mathcal{L},\mathcal{V})$ and a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} \in \mathcal{GRS}$, the* syndrome polynomial *$S(x)$ is given by*

$$S(x) = \sum_{i=0}^{n} \frac{r_i v_i}{1-\alpha_i x} \mod x^{d-1}, \tag{2.9}$$

where r_i is the i-th entry of $\mathbf{r}$ and $\alpha_i \in \mathcal{L}, v_i \in \mathcal{V}$.
Alternatively, given the parity check matrix $\mathbf{H}_{GRS}$ of $\mathcal{GRS}$, the coefficients of $S(x)$ can be computed by

$$S(x) = \sum_{i=0}^{d-2} s_i x^i,$$

where the s_i are the entries of the syndrome vector $\mathbf{s} = \mathbf{r}\mathbf{H}_{GRS}^T$.

Definition 2.4.2 (RS Key Equation)**.** *Assume $\mathcal{E}$ is the set of error positions, namely, $\mathcal{E} = \{i | e_i \neq 0\}$. Define the* error locator polynomial *(ELP) as*

$$\Lambda(x) := \prod_{i\in\mathcal{E}} (1-\alpha_i x)$$

and the error evaluator polynomial *(EEP) as*

$$\Omega(x) := \sum_{i\in\mathcal{E}} e_i v_j \prod_{j\in\mathcal{E}\setminus\{i\}} (1-\alpha_j x).$$

The GRS key equation *consists of the following equations:*

$$\Omega(x) \equiv \Lambda(x)S(x) \mod x^{d-1} \tag{2.10}$$

$$\gcd(\Lambda(x),\Omega(x)) = 1$$

$$\deg\Omega(x) < \deg\Lambda(x) \le \frac{d-1}{2}. \tag{2.11}$$

Algorithm 2.1: RS Syndrome-based Decoder

```
Data: A received word $\mathbf{r}=\mathbf{c}+\mathbf{e}$ with $\mathbf{c}\in\Gamma(\mathcal{L},g)$ and the super code $\mathcal{GRS}(\mathcal{L},\mathcal{V})$
      (see Theorem 2.2.2) of $\Gamma(\mathcal{L},g)$
Result: Correct codeword $\mathbf{c}\in\Gamma(\mathcal{L},g)$
1 Calculate syndrome polynomial $S(x)$;
2 Solve the RS key equation to obtain ELP $\Lambda(x)$ and EEP $\Omega(x)$;
3 Determine the error locations and value by Forney's formula:
      $e_i = \begin{cases} -\frac{\alpha_i}{v_i}\cdot\frac{\Omega(\alpha_i^{-1})}{\Lambda'(\alpha_i^{-1})} & \text{if } \Lambda(\alpha_i^{-1})=0 \\ 0 & \text{otherwise} \end{cases}, i=0,1,\dots,n-1,$
  where $\lambda'(x)$ is the formal derivative of the polynomial $\lambda(x)$;
4 Return $\mathbf{c}=\mathbf{r}-\mathbf{e}$.
```

Algorithm 2.1 summarizes the syndrome-based decoding algorithm of generalized RS codes from [Rot06, Ch. 6]. Step 2 can be done by solving a *linear system of equations* (LSE), which is constructed by equating the coefficients of $1,\dots,x^{d-2}$ from left-hand-side (LHS) to those from right-hand-side (RHS) of (2.10). The resulting algorithm is known as *Peterson-Gorenstein-Zierler algorithm* ([Rot06, Sec. 6.3.1]), whose time complexity is cubic in d. More efficient alternatives are the *extended Euclidean algorithm* ([Rot06, Sec. 6.4]) and the *Berlekamp-Massey algorithm* ([Rot06, Sec. 6.7]), whose complexities are quadradic in d. All these algorithms give the unique solution of $\Lambda(x),\Omega(x)$ that satisfies the key equation.

Decoding Radius of RS Syndrome-based Decoder

Theorem 2.4.1 (Decoding Radius of RS Syndrome-based Decoder)**.** *For a Goppa code $\Gamma(\mathcal{L}, g)$ with $\deg(g) = r$, Algorithm 2.1 returns the correct codeword if*

$$wt(\mathbf{e}) \leq \left\lfloor \frac{r}{2} \right\rfloor .$$

Proof. The Goppa code has minimum distance $d = r + 1$ (Theorem 2.2.1). From (2.11) we can see that $\deg(\Lambda)$ is at most $\frac{r}{2}$. The degree of $\Lambda(x)$ is defined by the number of errors $\mathrm{wt}(\mathbf{e})$, which is an integer. Thus, at most $\lfloor \frac{r}{2} \rfloor$ error can be corrected by solving the GRS key equation. □

Corollary 2.4.1. *For a wild Goppa code $\Gamma(\mathcal{L}, g^{q-1})$ with $\deg(g^{q-1}) = r$,* Algorithm 2.1 *returns the correct codeword if*

$$\mathrm{wt}(\mathbf{e}) \leq \left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor .$$

Proof. Follows from Theorem 2.3.2 and Therem 2.4.1. □

2.4.2 Patterson Decoder for Binary Squary-free Goppa codes

In the summary [Ber73] from Berlekamp, the syndrome and key equation for decoding Goppa codes are defined slightly different from those of RS codes.

Definition 2.4.3 (Goppa Syndrome)**.** *Given a Goppa code $\Gamma(\mathcal{L}, g)$ and a received word $\mathbf{r} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \Gamma(\mathcal{L}, g)$, the syndrome polynomial $s(x)$ is defined as*

$$s(x) := \sum_{i=0}^{n-1} \frac{e_i}{x - \alpha_i} \mod g(x) \tag{2.12}$$

and can be calculated from the received word $\mathbf{r}$ by

$$s(x) = \sum_{i=0}^{n-1} \frac{r_i}{x - \alpha_i} \mod g(x). \tag{2.13}$$

Definition 2.4.4 (Goppa Key Equation)**.** *Assume $\mathcal{E}$ is the set of error positions, namely, $\mathcal{E} = \{i | e_i \neq 0\}$. Define ELP $\sigma(x)$ and EEP $\eta(x)$ respectively as*

$$\sigma(x) := \prod_{i \in \mathcal{E}} (x - \alpha_i)$$

Algorithm 2.2: Patterson Decoder

```
Data: a binary square-free Goppa code $\Gamma(\mathcal{L}, g)$ and received word $\mathbf{r}$
Result: Correct codeword $\mathbf{c} \in \Gamma(\mathcal{L}, g)$
1 Calculate syndrome polynomial $s(x)$;
2 if $\frac{1}{s(x)} \mod g(x) = x$ then
3     set $\sigma(x) = x$ and go to step 7;
4 Calculate $v(x) = \sqrt{\frac{1}{s(x)} + x} \mod g(x)$;
5 Find $a_0(x), a_1(x)$ s.t.
      $$a_0(x) \equiv v(x)\ a_1(x) \mod g(x) \qquad (2.15)$$
  using extended Euclidean algorithm with restrictions
      $$\deg(a_i) \leq \left\lfloor \frac{r-i}{2} \right\rfloor, i = 0, 1; \qquad (2.16)$$
6 Compute $\sigma(x)$ by
      $$\sigma(x) = a_0(x)^2 + x a_1(x)^2; \qquad (2.17)$$
7 Return the $\mathbf{c}$ where
      $$c_i = \begin{cases} r_i + 1 & \text{if } \sigma(\alpha_i) = 0 \\ r_i & \text{otherwise} \end{cases}, i = 0, \ldots, n-1.$$
```

$$\eta(x) := \sum_{i \in \mathcal{E}} e_i \prod_{j \in \mathcal{E} \setminus \{i\}} (x - \alpha_j).$$

The Goppa key equation *consists of the following equations:*

$$\eta(x) \equiv \sigma(x) s(x) \mod g(x), \tag{2.14}$$

$$\gcd(\sigma(x), \eta(x)) = 1,$$

$$\deg \eta(x) < \deg \sigma(x) \leq \frac{r}{2}.$$

Patterson [Pat75] proposed an algorithm for binary square-free Goppa codes based on the Goppa key equation, which is summarize in Algorithm 2.2.

Theorem 2.4.2. Algorithm 2.2 *is correct for binary square-free Goppa codes.*

Proof. As a proof, we recap the derivation of the algorithm.

Consider a binary square-free Goppa code $\Gamma(\mathcal{L}, g)$. Since the error value could only be 1 in binary codes, the EEP $\eta(x)$ is equal to $\sigma'(x)$, the formal derivative of $\sigma(x)$. Therefore

(2.14) becomes

$$\sigma'(x) \equiv \sigma(x)s(x) \mod g(x). \tag{2.18}$$

Being a polynomial in characteristic 2, $\sigma(x)$ in module $g(x)$ can be written as (2.17) for some $a_0(x), a_1(x)$ with degree restrictions (2.16) and hence

$$\sigma'(x) = 2a_0(x)a_0'(x) + a_1(x)^2 + 2xa_1(x)a_1'(x) = a_1(x)^2. \tag{2.19}$$

Substituting $\sigma(x)$ and $\sigma(x)'$ in (2.18) by (2.17) and (2.19) respectively, we obtain

$$a_1(x)^2 \equiv (a_0(x)^2 + xa_1(x)^2)s(x) \mod g(x),$$

whence

$$a_0(x) \equiv \sqrt{\frac{1}{s(x)} + x}\; a_1(x) \mod g(x) \tag{2.20}$$

Let $v(x) \equiv \sqrt{\frac{1}{s(x)} + x} \mod g(x)$ we obtain (2.15).
If $\frac{1}{s(x)} = x$, $\sigma(x) = x$ is the solution, since (2.18) directly holds in this case. Otherwise, there surely exists a unique nonzero polynomial $v(x)$ if $g(x)$ is square-free (referring to Theorem 2.4.1 for $p = 2$).
Solving (2.15) for $a_0(x)$ and $a_1(x)$ is equivalent to the problem of finding two polynomials $a_1(x)$ and $\beta(x)$ such that $\gcd(v(x), g(x)) = a_1(x)v(x) + \beta(x)g(x)$, where $a_0(x) = \gcd(v(x), g(x))$. This problem can be solved by extended Euclidean algorithm with the degree restrictions (2.16). $\square$

Since (2.18) holds only when errors are binary vectors and $v(x)$ exists if and only if $g(x)$ is square-free, we can conclude that Patterson decoder *only* works for binary square-free Goppa codes.

Remark. *A binary square-free Goppa code $\Gamma(\mathcal{L}, g)$ is equivalent to $\Gamma(\mathcal{L}, g^2)$ (see Section 2.3.2 wild Goppa codes). We can also decode $\Gamma(\mathcal{L}, g)$ by applying RS syndrome-based decoder on $\Gamma(\mathcal{L}, g^2)$ but with larger computational costs.*

Decoding Radius of Patterson Decoder

Theorem 2.4.3 (Decoding Radius of Patterson Decoder)**.** *For a binary square-free Goppa code $\Gamma(\mathcal{L}, g)$ with $\deg(g) = r$,* Algorithm 2.2 *returns the correct codeword if*

$$wt(\mathbf{e}) \leq r.$$

Proof. From the definition of ELP $\sigma(x)$, we can see that the number of correctable errors

$\mathrm{wt}(\mathbf{e}) = \deg(\sigma(x))$. In Patterson decoder, since $\sigma(x)$ is calculated from $a_0(x)$ and $a_1(x)$ by (2.17) and degrees of $a_0(x)$ and $a_1(x)$ are restricted by (2.16). It follows that

$$\begin{aligned}\deg(\sigma(x)) &= \max\{2 \cdot \deg(a_0),\ 2 \cdot \deg(a_1) + 1\} \\ &\leq \max\{2 \cdot \left\lfloor \frac{r}{2} \right\rfloor,\ 2 \cdot \left\lfloor \frac{r-1}{2} \right\rfloor + 1\} \\ &= r\end{aligned}$$

□

2.4.3 p-ary Patterson Decoder for Squary-free Goppa codes

A nondeterministic decoding algorithm for square-free Goppa codes $\Gamma(\mathcal{L}, g)$ over $\mathbb{F}_p$ is proposed by Barreto et al. in [BML13], where p is a prime. We summerize the algorithm in Algorithm 2.3 and refer it as *p-ary Patterson Decoder*.

The derivation of this algorithm is similar to that of Patterson decoder and is given in [BML13]. Here we show that the p-ary Patterson decoder works only for square-free Goppa codes.

Lemma 2.4.1. *Let p be a prime and m be an integer. Given a polynomial $g(x) \in \mathbb{F}_{p^m}[x]$, for any polynomial $h(x) \in \mathbb{F}_{p^m}[x]$, $\sqrt[p]{h(x)}$ exist in $\mathbb{F}_{p^m}[x]/g(x)$ if and only if $g(x)$ is square-free.*

Proof. Being a polynomial in characteristic p, we can write $h(x)$ by

$$h(x) = h_0(x)^p + xh_1(x)^p + x^2h_2(x)^p + \cdots + x^{p-1}h_{p-1}(x)^p.$$

Then,

$$\sqrt[p]{h(x)} = h_0(x) + \sqrt[p]{x}h_1(x) + (\sqrt[p]{x})^2h_2(x) + \cdots + (\sqrt[p]{x})^{p-1}h_{p-1}(x) \mod g(x). \quad (2.21)$$

We can write

$$g(x) = g_0(x)^p + xg_1(x)^p \quad (2.22)$$

since it is also a polynomial in characteristic p. Therefore,

$$\begin{aligned}\sqrt[p]{x} = -\frac{g_0(x)}{g_1(x)} \mod g(x) \text{ exists} &\iff g_1(x)^{-1} \mod g(x) \quad \text{exists} \\ &\iff \gcd(g(x), g_1(x)) = 1.\end{aligned}$$

Algorithm 2.3: p-ary Patterson Decoder

```
Data: A Goppa code $\Gamma(\mathcal{L}, g)$ over $\mathbb{F}_p$ with $g(x)$ being square-free and a received
      codeword $\mathbf{r}$
Result: a list of corrected codewords $\mathbf{c} \in \Gamma(L, g)$ ($\varnothing$ upon failure)
1  Calculate the syndrome polynomial $s(x)$
2  $S \leftarrow \varnothing$
3  foreach $\phi \leftarrow 1$ to $p - 1$ do
4      # guess the correct scale factor $\phi$
5      foreach $k \leftarrow 1$ to $p - 1$ do
6          Calculate $u_k(x) = x^k - \phi \frac{kx^{k-1}}{s(x)} \mod g(x)$;
7          Calculate $v_k(x) = \sqrt[p]{u_k(x)} \mod g(x)$;
8      Build the lattice $\Lambda(A_\phi)$ whose basis $A_\phi$ is
           $\begin{bmatrix} g(x) & & & \\ -v_1(x) & 1 & & \\ \vdots & & \ddots & \\ -v_{p-1} & & & 1 \end{bmatrix};$
9      Reduce the basis of $\Lambda(A_\phi)$;
10     foreach $i \leftarrow 1$ to $p$ do
11         Let $a$ denote the $i$-th row of the reduced basis of $\Lambda(A_\phi)$;
12         foreach $j \leftarrow 0$ to $p - 1$ do
13             if $\deg(a_j) > \lfloor \frac{(t-j)}{p} \rfloor$ then
14                 try next $i$; # not a solution
15         Calculate $\sigma_\phi(x) = \sum_j x^j a_j(x)^p$;
16         Compute the set $\mathcal{J} \subset \mathcal{L}$ such that $\sigma_\phi(\alpha_j) = 0$;
17         $\mathbf{e} \leftarrow \mathbf{0}$ of length $n$;
18         foreach $j \in J$ do
19             Compute the multiplicity $\mu_j$ of $\alpha_j$;
20             $e_j \leftarrow \phi\mu_j$;
21         if $\sum_{i=0}^{n-1} \frac{e_i}{x-\alpha_i} \mod g(x) \equiv 0$ then
22             $S \leftarrow S \cup \{\mathbf{r} - \mathbf{e}\}$;
23 Return $S$.
```

From (2.22), the formal derivative of $g(x)$ is

$$g'(x) = pg_0(x)^{p-1} + g_1(x)^p + pxg_1(x)^{p-1} = g_1(x)^p,$$

since we work in characteristic p. Because of $\gcd(g_1(x), g_1(x)^p) = g_1(x)$,

$$\begin{aligned}\sqrt[p]{x} = -\frac{g_0(x)}{g_1(x)} \mod g(x) \text{ exists } &\iff \gcd(g(x), g_1(x)^p) = 1 \\ &\iff \gcd(g(x), g'(x)) = 1.\end{aligned}$$

It is easy to see that

$$\gcd(g(x), g'(x)) = 1 \iff g(x) \text{ is square-free.}$$

Once we obtain $\sqrt[p]{x}$, $\sqrt[p]{h(x)}$ can be calculated from (2.21). □

It follows from Lemma 2.4.1 that Step 7 can be done only if $g(x)$ is square free. Thus p-ary Patterson decoder (Algorithm 2.3) works only for square-free Goppa codes.
In step 7, if $g(x)$ is square-free but not irreducible, $s(x)$ may not be invertible modulo $g(x)$ but is invertible modulo each of the irreducible factors of $g(x)$. In this case, $v_k(x)$ can be calculated by modulo each of irreducible factors of $g(x)$ and then recovered via the *Chinese Remainder Theorem.*
In step 9, *Mulders and Stojohann's algorithm* [MS03] can be used to do the lattice reduction to find candidate solutions a_k.

Decoding Radius of p-ary Patterson Decoder

It is shown in [BML13] that without knowing the error magnitudes distribution, the maximum number of errors that can be corrected by the p-ary Patterson decoder is $\left\lfloor \frac{2}{p} r \right\rfloor$. However, this is not a guaranteed decoding radius. By choosing the error magnitudes to be equal, up to r errors can be corrected with high probability.

3 Code-based Cryptosystem - McEliece System

This chapter recaps the structure of the McEliece System and Wild McEliece. Several attacks that effectively decrease the security level of the McEliece system are also covered.

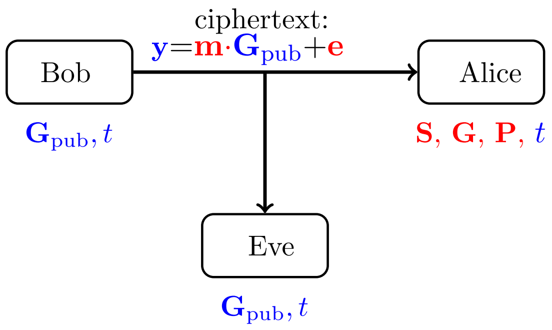
3.1 System Description

The McEliece cryptosystem is a code-based public-key system proposed by Robert J. McEliece in 1978 [JM78]. It is one of the promising candidates for post-quantum cryptography.
The McEliece cryptosystem works like this:

- **Key generation (Alice)**
  - Choose a t-error-correcting linear $[n, k]$ code with generator matrix $\mathbf{G}$,
  - Generate a random-looking matrix $\mathbf{G}_{\text{pub}}$ by $\mathbf{G}_{\text{pub}} = \mathbf{SGP}$, where $\mathbf{S}$ is a random full-rank scramble matrix and $\mathbf{P}$ is a random permutation matrix,
  - Publish the public key $(\mathbf{G}_{\text{pub}}, t)$ and keep the secret key $(\mathbf{G}, \mathbf{S}, \mathbf{P})$ secret.
- **Encryption (Bob)**
  Generate the ciphertext $\mathbf{y}=\mathbf{m}\cdot\mathbf{G}_{\text{pub}}+\mathbf{e}$, where $\mathbf{m}$ is a plaintext vector and $\mathbf{e}$ is an error vector of weight t.
- **Decryption (Alice)**
  - Inverse permutation: $\mathbf{y}' = \mathbf{y}\mathbf{P}^{-1}$;
  - Correct errors: $\mathbf{m}' = \mathcal{D}(\mathbf{y}')$, where $\mathcal{D}(\cdot)$ is an efficient decoding algorithm of the secret code correcting t errors;
  - Inverse scramble: $\mathbf{m} = \mathbf{m}'\mathbf{S}^{-1}$.

Notice that the inverse of a permutation matrix is again a permutation matrix and right-multiplying a permutation matrix to a vector does not change the weight of the vector, i.e., $\text{wt}(\mathbf{e}) = \text{wt}(\mathbf{e}\mathbf{P}^{-1})$.

Eve can not reveal the plaintext from the ciphertext efficiently since $\mathbf{G}_{\text{pub}}$ generates a obfuscated code and decoding a random-looking code is a hard problem.
Figure 3.1 gives an illustration of the principle of the McEliece cryptosystem, where the red letters $\mathbf{m}, \mathbf{e}, \mathbf{G}, \mathbf{S}, \mathbf{P}$ should be kept secret and the blue letters $\mathbf{G}_{\text{pub}}, t, \mathbf{y}$ are accessible by public.


ciphertext:
$\mathbf{y}=\mathbf{m}\cdot\mathbf{G}_{\text{pub}}+\mathbf{e}$
Bob
Alice
$\mathbf{G}_{\text{pub}}, t$
$\mathbf{S}, \mathbf{G}, \mathbf{P}, t$
Eve
$\mathbf{G}_{\text{pub}}, t$

Credit: *[EWZZ18]*

Figure 3.1: Principle of the McEliece cryptosystem.

Work Factor & Security Level

In principle every computational cryptographic method is breakable, just a matter of time and effort. **Work factor** (WF) is the number of bit operations that a certain adversarial algorithm needs to do in order to break a cipher.
Security level (SL) is a measure of the strength that a cryptographic algorithm achieves. Denote the work factor of the *fastest* attack by W. The security level of a cryptosystem is defined by $SL = \log_2(W)$ and is usually represented in "bits", that is, n-bits security means that the fastest attack would have to perform 2^n bit operations to break the cipher.

3.2 Attacks

As we can see from Figure 3.1, Eve can access to both $\mathbf{G}_{\text{pub}}$ and the ciphertext $\mathbf{y}$. Eve has two ways to retrieve the secret plaintext $\mathbf{m}$ from $\mathbf{y}$:

- find out the secret code from the public code, i.e., retrieve $\mathbf{G}$ from $\mathbf{G}_{\text{pub}}$;
- decode $\mathbf{y}$ without knowning an efficient decoding algorithm for the public code.

Attacks of the first type are called *structural attacks*. These attacks are customized for specific code classes, such as *Sidelnikov-Shestakov attack* [MSOS92] on RS codes. If the secret generator matrix $\mathbf{G}$ or an equivalent efficiently recognizable representation of the

underlying secret code can be retrieved in sub-exponential time, this code should not be used in the McEliece system. The structural attack on Goppa codes can be only done by testing the equivalence between all Goppa codes and the code generated by $\mathbf{G}$. The complexity is exponential to the field order m and the degree r of the Goppa polynomial. Thus, Goppa codes are secure codes against structural attack.
The second type is called *decoding attack*. In the following we will describe how to correct errors in a random-looking code without any obvious structure.

3.2.1 Information Set Decoding Attack

The *information set decoding* (ISD) attack relies on the "encoding-and-comparison" decoding algorithm proposed by Eugene Prange in 1962 [Pra62]. The idea is to find an information set $\mathcal{I}$ of coordinates with $|\mathcal{I}| = k$. Denote by $\mathbf{y}_\mathcal{I}$ the positions indexed by $\mathcal{I}$ in $\mathbf{y}$ and by $\mathbf{G}_{\text{pub},\mathcal{I}}$ the columns indexed by $\mathcal{I}$ in $\mathbf{G}_{\text{pub}}$. The ISD algorithm hopes that $\mathbf{y}_\mathcal{I}$ contains only error-free positions. The original message can then be computed by $\mathbf{m} = \mathbf{y}_\mathcal{I} \cdot \mathbf{G}^{-1}_{\text{pub},\mathcal{I}}$.

Algorithm 3.1: Information Set Decoding

```
Data: Generator matrix $\mathbf{G}_{\text{pub}}$ of size $k \times n$, ciphertext $\mathbf{y}$, number of errors $t$
Result: Secret plaintext $\mathbf{m}$
1 Choose an $\mathcal{I} \subset \{0,\dots,n-1\}$, $|\mathcal{I}| = k$ that has never been chosen
2 if $\text{wt}\Big(\big(\mathbf{y}_\mathcal{I} \cdot \mathbf{G}^{-1}_{pub,\mathcal{I}}\big) \cdot \mathbf{G}_{pub} - \mathbf{y}\Big) = t$ then
3     Return $\mathbf{m} = \mathbf{y}_\mathcal{I} \cdot \mathbf{G}^{-1}_{\text{pub},\mathcal{I}}$
4 else
5     Repeat from step 1
```

Theorem 3.2.1 (Work Factor of ISD)**.** *The work factor of* Algorithm 3.1 *on an McEliece system with binary codes and parameters* (n, k, t) *is*

$$W_{ISD}(n,k,t) = k^3 \cdot \frac{\binom{n}{k}}{\binom{n-t}{k}}.$$

Example 2. *The security level of original parameters ($n = 1024, k = 524, t = 50$) porposed by McEliece against ISD attack is 80.71 bits.*

The ISD algorithm is the best method to decode an arbitrary linear code. The first analysis was done by McEliece in [JM78]. Several improvements were done by Lee and Brickell in [LB88], by Leon in [Leo88] and by Stern in [Ste89].

The ISD attack on McEliece system can be reduced into a another problem: *finding low-weight codewords (FLWCW) problem.* Stern's algorithm [Ste89] is originally an algorithm for finding low-weight codewords. We do not show in detail how to find the low weight codewords. For the interest, please refer to [Ste89, Pet10]. We just give the complexity of Stern's algorithm in terms of number of bit operations in Lemma 3.2.1.

Lemma 3.2.1 (Bit Operations of Stern's algorithm [Ste89])**.** *The number of bit operations of finding a codeword of weight t of an $[n,k,t]$ code by Stern's algorithm is*

$$\mathit{FLWCW}(n,k,t) = \frac{\binom{n}{k}}{\binom{t}{2p}\binom{n-t}{k-2p}} \cdot \frac{4^p}{\binom{2p}{p}} \cdot \frac{\binom{n-k}{\lambda}}{\binom{n-k-t+2p}{\lambda}} \cdot \left(\frac{1}{2}(n-k)^3 + k(n-k)^3 + 2\lambda p\binom{k/2}{p} + \frac{2p(n-k)\binom{k/2}{p}^2}{2^\lambda}\right),$$

where p, λ are parameters to optimize the complexity. Usually p is a small positive integer and $\lambda = \left\lceil \log_2 \binom{k'/2}{p} \right\rceil$ [Pet10].

Algorithm 3.2 shows the idea of using Stern's algorithm to attack the McEliece system.

Algorithm 3.2: Stern's Attack

```
Data: Generator matrix $\mathbf{G}_{\text{pub}}$ of size $k \times n$, ciphertext $\mathbf{y}$, number of errors $t$
Result: Secret plaintext $\mathbf{m}$
1 Append $\mathbf{y}$ to $\mathbf{G}_{\text{pub}}$ to form a new generator matrix $\mathbf{G}' = \begin{pmatrix}\mathbf{G}_{\text{pub}} \\ \mathbf{y}\end{pmatrix}$ of an $[n, k+1, t]$
  code $\mathcal{C}$';
2 Find the lowest-weight codeword $\mathbf{e}$ in the code $\mathcal{C}$' by Stern's algorithm [Ste89];
3 Set $\mathcal{I} \subset \{0, 1, \ldots, n-1\} \backslash \text{supp}(\mathbf{e})$;
4 Return $\mathbf{m} = \mathbf{y}_{\mathcal{I}} \cdot \mathbf{G}^{-1}_{\text{pub},\mathcal{I}}$.
```

Corollary 3.2.1 (Work Factor of Stern's Attack)**.** *The work factor of* Algorithm 3.2 *on an McEliece system with parameters (n,k,t) is*

$$W_S(n,k,t) = k^3 + \mathit{FLWCW}(n, k+1, t).$$

Proof. k^3 comes from step 4 and the other component comes from step 2. □

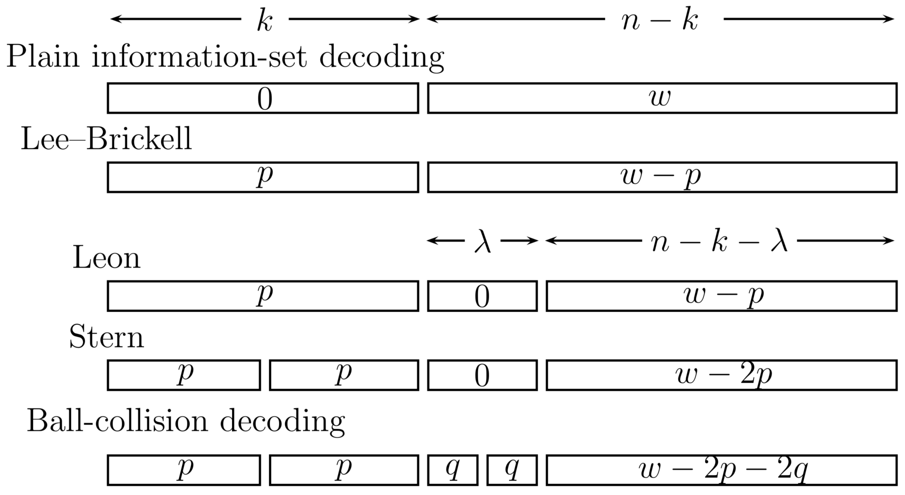
Example 3. *The security level of original parameters ($n = 1024, k = 524, t = 50$) against Stern's attack is 67.95 bits ($p = 4$).*

Bernstein, Lange and Peters improve Stern's attack in [BLP08]. This improvement takes only $2^{60.5}$ bit operations on the original McEliece parameters ($n = 1024, k = 524, t = 50$).

Finiasz and Sendrier presented in [FS09] "a lower bound on the effective work factor of exsiting and coming attacks", which computed $2^{59.9}$ as a bound for the original McEliece parameters. Bernstein, Lange and Peters introduced *ball-collision attack* in [BLP11a], where they came up with a new simpler lower bound on the work factor of their ball-collision attack:

$$W \geq \min \left\{ \frac{1}{2} \binom{n}{t} \binom{n-k}{t-p}^{-1} \binom{k}{p}^{-\frac{1}{2}} \ : \ 0 \leq p \leq \min\{t, k\} \right\}$$

The ball collision is actually a further improvement of Stern's attack. Figure 3.2 shows the principles of the improvements of ISD attack: allowing errors to appear in the information set. Further, we may refer all these attacks shown in Figure 3.2 as ISD-based attacks and Algorithm 3.1 as naive ISD algorithm.



Credit: *[BLP11a]*

Figure 3.2: Improvements of ISD attack. There are w errors in total. ISD allows no errors in the information set. The improvements in turn allows more errors in different segments to give better work factor of the algorithm.

The aforementioned attacks only concern about $\mathbb{F}_2$. Peters generalized the Lee-Brickell's, Stern's algorithms over arbitrary fields $\mathbb{F}_q$ in [Pet10]. Pradeep investigated in [Pra18] the work factor of the generalized algorithms from [Pet10]. We conclude the results of [Pra18] by Lemma 3.2.2.

Lemma 3.2.2 (Work Factor of ISD Attacks in $\mathbb{F}_q$)**.** *For general field size q, the work factor increases by a factor of $(\log_2 q)^2$ for ISD and more than $(\log_2 q)^2$ for its improvements.*

Following from Lemma 3.2.2, we suggest that the bound on the work factor should also increase by a factor of $(\log_2 q)^2$ for general q, as stated in Theorem 3.2.2.

Theorem 3.2.2 (Lower Bound on Work Factor)**.** *The work factor W of ISD-based algorithms on an McEliece system with parameters (n, k, t) over $\mathbb{F}_q$ is*

$$W \geq W_B(n,k,t) = \min \left\{ \frac{1}{2} \binom{n}{t} \binom{n-k}{t-p}^{-1} \binom{k}{p}^{-\frac{1}{2}} (\log_2 q)^2 \ : \ 0 \leq p \leq \min\{t,k\} \right\}$$

Remark. *This bound is rather heuristic because the bound for binary codes given in [BLP11a] is not proven, either. But it is in a very simple form, easy to compute and lower-bounds the best known attack: ball-collision attack. So we recommend to use this bound as a security measure for selecting the parameters of the McEliece system.*

Further, we will use this formula to evaluate the security level of the McEliece system with Goppa codes.

Example 4. *The security level of original parameters ($n = 1024, k = 524, t = 50$) against this bound is 49.69 bits ($p = 5$).*

3.3 Wild McEliece

Wild McEliece is a variant of the McEliece system using wild Goppa codes (see Section 2.3.2) to reduce the key size of the McEliece system. It was proposed by Bernstein, Lange and Peters in [BLP11b].
The wild Goppa codes $\Gamma(\mathcal{L}, g^{q-1})$ over $\mathbb{F}_q$ can be uniquely decoded by the RS syndrome-based decoder (Algorithm 2.1) or a newly proposed *Goppa Key Equation Decoder* (Algorithm 4.1) in Section 4.3.1 when $t \leq \left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor$ errors happen, where $r = \deg(g^{q-1})$. Note that the improvement of decoding radius in wild McEliece is only effective for non-binary Goppa codes, since the binary square-free Goppa codes that the original McEliece uses which are wild Goppa codes by default.

3.4 Attacks on Wild McEliece

3.4.1 Structural Attack

Polynomial Search Attack

There are approximately q^{mr}/r monic irreducible polynomials g of degree r in $\mathbb{F}_{q^m}[x]$, and therefore approximately q^{mr}/r choices of g^{q-1}. One can marginally expand the space

of polynomials by considering more general square-free polynomials g, but here we focus on irreducible polynomials for simplicity.
An attacker can try to guess the Goppa polynomial g^{q-1} and then apply Sendriers *support-splitting algorithm* [Sen00] to compute the support $\mathcal{L}$.

Square Code Operation

In 2013, Faugère, Umaña, Otmani, Perret and Tillich showed in [FnO+13] that a *square code operation*:

$$\mathcal{C} = \mathcal{C}_1 \star \mathcal{C}_2 := \{\mathbf{u} \star \mathbf{v} := (u_1v_1, u_2v_2, \dots, u_nv_n),\ \forall \mathbf{u} \in \mathcal{C}_1, \mathbf{v} \in \mathcal{C}_2\}$$

can be used to distinguish high rate Goppa codes from a random code because the dimension of the square of the dual is much smaller than that of a ramdom code. The worst case possibility is that this distinguisher somehow allows an inversion attack faster than decoding attacks. However, the distinguisher stops working at 8 errors for $n = 1024$ (where orinial McEliece parameter has already 50 errors) and at 20 errors for $n = 8192$ (where for such long length codes, we usually choose the Goppa codes that can decode more than 100 errors).
In a recent paper [COT17], Couvreur, Otmani and Tillich introduced a polynomial-time attack on wild Goppa codes over quadratic extensions. It turns out that the wild Goppa codes with $m = 2$ can be distinguished from random codes by observing that the square code of some of their shortenings have a small dimension compared to squares of random codes of the same dimension.

3.4.2 Parameters Selection Against Structural Attacks

We suggest the following restrictions on selecting parameters of wild Goppa parameters to defense wild McEliece and its possible variants against the structural attacks mentioned above:

- Keep q^{mr}/r extremly large, e.g., $\geq 2^{128}$, so that guessing g^{q-1} has neglegible chance of success;
- Keep $\mathcal{L}$ noticeably smaller than $\mathbb{F}_{q^m}$, so that the support-splitting algorithm takes a guess of $\mathcal{L}$ as another input along with g;
- Keep m large (at least > 2) and keep q s.t. q^m is sufficiently large, e.g., ≥ 2048;
- Keep code rate $R = n/k$ noticeably away from 0 and 1 (optional, since the high-rate code distinguisher stops working when too many errors occurs).

3.4.3 Decoding Attack

The most effective attack on the original McEliece cryptosystem is the ISD attack. As a generic decoding method, ISD does not rely on any particular code structure. ISD-based attacks also appear to be the top threat to wild McEliece and determine its security level. The exact complexity of ISD is not easy to state concisely, since there are a lot of improvements. We rely on the bound in Theorem 3.2.2 to evaluate the security level of wild McEliece with parameters n, k, t, q.
Note that the number of errors t is at most $\left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor$ in a $\Gamma(\mathcal{L}, g^{q-1})$ wild McEliece with $r = \deg(g^{q-1})$, while being at most $\left\lfloor \frac{r}{2} \right\rfloor$ in a $\Gamma(\mathcal{L}, g)$ McEliece with $r = \deg(g)$. The increasing of error positions results in a better security level of wild McEliece. Or alternatively, for the same security level, wild McEliece needs smaller key size than the McEliece system.

4 Interleaved Goppa Codes

In this chapter we give the definiton of interleaved Goppa codes and introduce a probabilistic decoding algorithm that can decode more than half of the minimum distance errors. The failure probability is investigated by simulations with several codes. At the end of this chapter, a `GoppaCode` class and several decoder classes implenmented in *SageMath* are presented.

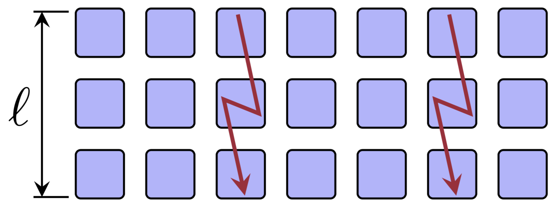
4.1 Definition

Definition 4.1.1 (Interleaved Goppa Codes)**.** *Let* $\Gamma(\mathcal{L}, g)$ *denote a Goppa code as defined in Definition 2.1.1. An* ℓ-interleaved Goppa code *(IGC) is denoted by* $\mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ *and defined by*

$$\mathcal{I}\Gamma(\mathcal{L}, g, \ell) = \left\{ \begin{pmatrix} \mathbf{c}^{(1)} \\ \mathbf{c}^{(2)} \\ \vdots \\ \mathbf{c}^{(\ell)} \end{pmatrix}, \quad \forall \mathbf{c}^{(i)} \in \Gamma(\mathcal{L}, g), i = 1, \ldots, \ell \right\}.$$

4.2 Burst Errors

Burst errors are error patterns that corrupt continuous positions in data streams. If we transmit the codewords of IGC column by column as data streams, then retrieve the received streams into $\ell \times n$ arrays, the burst errors can be seen as error matrices that corrupt columns of the codewords of $\mathcal{I}\Gamma$, as illustrated in Figure 4.1. This can be also seen as ℓ Goppa codewords being corrupted at the same positions. Hence, a collaborative decoding strategy can be applied, which locates the errors jointly in all words instead of locating them independently in the several words. This allows to uniquely locate up to t errors, in many cases even if t is greater than the half the minimum distance of Γ.



Credit: *[WZ17]*

Figure 4.1: Illustration of burst errors on interleaved codes.

Consider an $\mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ over $\mathbb{F}_q$. Assume we observe a received word

$$\mathbf{R} = \begin{pmatrix} \mathbf{r}^{(1)} \\ \mathbf{r}^{(2)} \\ \vdots \\ \mathbf{r}^{(\ell)} \end{pmatrix} = \begin{pmatrix} \mathbf{c}^{(1)} \\ \mathbf{c}^{(2)} \\ \vdots \\ \mathbf{c}^{(\ell)} \end{pmatrix} + \begin{pmatrix} \mathbf{e}^{(1)} \\ \mathbf{e}^{(2)} \\ \vdots \\ \mathbf{e}^{(\ell)} \end{pmatrix} = \mathbf{C} + \mathbf{E},$$

where $\mathbf{C} \in \mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ and $\mathbf{E} \in \mathbb{F}_q^{\ell \times n}$. We define $\mathrm{wt}(\mathbf{E})$ by the number of non-zero columns of $\mathbf{E}$.

4.3 Decoding of Interleaved Goppa codes

4.3.1 Goppa Key Equation Decoder

Before we explain the concept of *collaboratively decoding* for IGC, we first introduce another BMD decoder of Goppa codes, the *Goppa Key Equation Decoder*, besides the ones in Chapter 2. The collaboratively decoding is based on this decoder.
We follow the definitions of Goppa syndrome polynomial $s(x)$, ELP $\sigma(x)$, EEP $\eta(x)$ and Goppa key equation from Definition 2.4.3 and 2.4.4. Unlike Patterson decoder doing a "polynomial splitting" on $\sigma(x)$, we will directly solve the Goppa key equation.
Solving the key equation is equivalent to solving a LSE with coefficients of $\sigma(x)$, $\eta(x)$ and $g(x)$.
In the following, we will show how to set up and solve the LSE.

Setting up LSE for Goppa Key Equation

We can write the polynomials with their coefficients explicitly by $g(x) = \sum\limits_{i=0}^{r} g_i x^i$, $s(x) = \sum\limits_{i=0}^{r-1} s_i x^i$, $\sigma(x) = \sum\limits_{i=0}^{t} \sigma_i x^i$ and $\eta(x) = \sum\limits_{i=0}^{t-1} \eta_i x^i$.

Then $\sigma(x)s(x)$ (without mod $g(x)$) can be written as

$$\begin{aligned}\sigma(x)s(x) =&(\sigma_0 s_0) + (\sigma_1 s_0 + \sigma_0 s_1)x + \cdots \\ &+ (\sigma_t s_{r-t-1} + \sigma_{t-1} s_{r-t} + \cdots + \sigma_0 s_{r-1})x^{r-1} \\ &+ (\sigma_t s_{r-t} + \cdots + \sigma_1 s_{r-1})x^r + \cdots \\ &+ (\sigma_t s_{r-1})x^{r+t-1}.\end{aligned} \tag{4.1}$$

The coefficients of x^i in (4.1) can be expressed by the multiplication of a matrix $\mathbf{M}$ with entries s_i and a vector $\underline{\sigma} = (\sigma_0, \ldots, \sigma_t)$:

$$\mathbf{M} \cdot \underline{\sigma}^T = \left[\begin{array}{ccccc} s_0 & & & & \\ s_1 & s_0 & & 0 & \\ \vdots & \vdots & \ddots & & \\ s_{t-1} & s_{t-2} & \cdots & s_0 & \\ \hdashline s_t & s_{t-1} & \cdots & s_1 & s_0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ s_{r-1} & s_{r-2} & \cdots & s_{r-t} & s_{r-t-1} \\ & s_{r-1} & \cdots & s_{r-t+1} & s_{r-t} \\ & & \ddots & \ddots & \vdots \\ & & & s_{r-1} & s_{r-2} \\ & & & & s_{r-1} \end{array}\right] \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} t \cdot \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_t \end{bmatrix}. \tag{4.2}$$

Taking mod $g(x)$ into account, the $x^r, \cdots, x^{r+t-1}$ terms in (4.1) should disappear and their coefficients will appear in the coefficients of $1, \cdots, x^{t-1}$ with some modification according to $g(x)$. Identically, the last t rows of $\mathbf{M}$ in (4.2) should be modified into the upper part of $\mathbf{M}$.

Notice that $b \cdot x^i \mod g(x) = b \cdot (x^i \mod g(x))$ where $b \in \mathbb{F}_{q^m}$ is a constant. We pre-calculate the residues of $x^{i+r} \mod g(x)$ for $0 \leq i \leq t-1$ and store the coefficients as column in a *look-up table* (LUT) $\mathbf{A}$, which is a $r \times t$ matrix. The entry A_{ij} is the coefficient of x^i in the residue of $x^{j+r} \mod g(x)$, for $0 \leq i \leq r-1, 0 \leq j \leq t$.

Let $\mathbf{M}' \cdot \underline{\sigma}^T$ contain the coefficients of x^i in $\sigma(x)s(x) \mod g(x)$, where $\mathbf{M}'$ is the modified upper r rows of $\mathbf{M}$ and is of size $r \times (t+1)$. Each entry M'_{ij} of $\mathbf{M}'$ is calculated from $\mathbf{M}$ and the LUT $\mathbf{A}$ by the following rule:

$$M'_{ij} = M_{ij} + \sum_{l=0}^{t-1} A_{il} \cdot M_{l+r,j}, \tag{4.3}$$

for $0 \leq i \leq r-1, 0 \leq j \leq t$.
We denote the upper r rows of $\mathbf{M}$ by $\mathbf{M}_{\text{up}}$ and lower t rows by $\mathbf{M}_{\text{lo}}$. Then equivalently, $\mathbf{M}'$ can be calculated by

$$\mathbf{M}' = \mathbf{M}_{\text{up}} + \mathbf{A} \cdot \mathbf{M}_{\text{lo}}$$

Then, equating the coefficients in (2.14) we get the following LSE for unknowns σ_i and η_i:

$$\mathbf{M}' \cdot \underline{\sigma}^T = \begin{bmatrix} M'_{00} & M'_{01} & \cdots & M'_{0,t-1} \\ M'_{10} & M'_{11} & \cdots & M'_{1,t-1} \\ \vdots & \vdots & \cdots & \vdots \\ M'_{t-1,0} & M'_{t-1,1} & \cdots & M'_{t-1,t-1} \\ \hdashline M'_{t,0} & M'_{t,1} & \cdots & M'_{t,t-1} \\ \vdots & \vdots & \cdots & \vdots \\ M'_{r-1,0} & M'_{r-1,1} & \cdots & M'_{r-1,t-1} \end{bmatrix} \cdot \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_t \end{bmatrix} = \begin{bmatrix} \eta_0 \\ \eta_1 \\ \vdots \\ \eta_{t-1} \\ \hdashline 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} r-t \end{matrix} \tag{4.4}$$

Given the LSE (4.4), we first solve the lower $r-t$ equations to obtain the σ_i. It can be seen that $\sigma_0 = \sigma_1 = \cdots = \sigma_t = 0$ is a trivial solution. In order to get a non-trivial solution, we set $\sigma_0 = 1$ and obtain the following LSE

$$\underbrace{\begin{bmatrix} M'_{t,1} & \cdots & M'_{t,t-1} \\ \vdots & \cdots & \vdots \\ M'_{r-1,1} & \cdots & M'_{r-1,t-1} \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_t \end{bmatrix} = - \begin{bmatrix} M'_{t,0} \\ \vdots \\ M'_{r-1,0} \end{bmatrix} \tag{4.5}$$

Once the $\sigma(x)$ is known, the decoding is almost done. Algorithm 4.1 summarizes all steps of our Goppa key equation decoder.

Remark. *For binary Goppa codes, we only execute the first step 1 - 4, since as long as we know the error positions, we can correct the error by flipping the bits at the error positions.*

Remark (t is known)**.** *The readers may notice that generally in a decoding problem the number of errors is not known to the decoder. Since we consider the application in McEliece system, where t is a system parameter known to Alice who decodes the received words, we regard t as a known algorithm input. Otherwise, t can be also searched by the algorithm itself but we do not consider this step here.*

Remark (Euclid's Algorithm)**.** *Instead of solving the LSE by Gaussian elimination, the key equation can be solved more efficiently by the extended Euclidean algorithm (EEA).*

Algorithm 4.1: Goppa Key Equation Decoder

```
Data: $\Gamma(\mathcal{L}, g)$, received word $\mathbf{r}$ and the number of errors $t = \mathrm{wt}(\mathbf{e})$
Result: Correct codeword $\mathbf{c} \in \Gamma(\mathcal{L}, g)$
1 Calculate the syndrome polynomial $s(x)$ given $\mathbf{r}$ by (2.13);
2 Setting up the LSE (4.5) according to (4.2) - (4.4);
3 Do Gaussian Elimination to obtain the solution $(\sigma_1, \dots, \sigma_t)$ of (4.5);
4 Calculate ELP by $\sigma(x) = \sigma_t x^t + \cdots + \sigma_1 x + 1$;
5 Calculate EEP by $\eta(x) \equiv \sigma(x)s(x) \mod g(x)$;
6 Determine the error vector $\mathbf{e}$ by
    $$e_i = \begin{cases} -\dfrac{\eta(\alpha_i)}{\sigma'(\alpha_i)} & \text{if } \sigma(\alpha_i) = 0 \\ 0 & \text{otherwise} \end{cases}, \quad i = 0, 1, \dots, n-1,$$
  where $\sigma'(x)$ is the formal derivative of $\sigma(x)$.
7 Return $\mathbf{c} = \mathbf{r} - \mathbf{e}$
```

EEA was used by Sugiyama et. al. in [SKHN75] to solve the key equation. They also showed the algorithm gives the unique solution.

Decoding Radius of Goppa Key Equation Decoder

Theorem 4.3.1 (Decoding Radius of Goppa Key Equation Decoder)**.** *For a Goppa code $\Gamma(\mathcal{L}, g)$ with $\deg(g) = r$,* Algorithm 4.1 *returns the unique and correct codeword if*

$$wt(\mathbf{e}) \leq \left\lfloor \frac{r}{2} \right\rfloor.$$

Proof. From the LSE (4.5) we see that we have t unknowns and $r - t$ equations. To have a unique solution we need that $\mathrm{rank}(\mathbf{S}) = t$ in (4.5). This is only possible when the number of unknowns is less than the number of linearly independent equations, i.e., $t \leq r - t$. □

Corollary 4.3.1. *For a wild Goppa code $\Gamma(\mathcal{L}, g^{q-1})$ with $\deg(g^{q-1}) = r$,* Algorithm 4.1 *returns the correct codeword if*

$$wt(\mathbf{e}) \leq \left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor.$$

Proof. Follows from Theorem 2.3.2 and Therem 4.3.1. □

4.3.2 Collaborative Decoding

Assume a codeword $\mathbf{C} \in \mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ is corrupted by an error $\mathbf{E}$ with $\mathrm{wt}(\mathbf{E}) = t$ and we receive $\mathbf{R} = \mathbf{C} + \mathbf{E}$. As shown in Section 4.3.1, each row $\mathbf{r}^{(i)}$ of $\mathbf{R}$ gives a key equation

$$\eta^{(i)}(x) \equiv \sigma(x)s^{(i)}(x) \mod g(x), \tag{4.6}$$

hence each also gives an LSE

$$\underbrace{\begin{bmatrix} {M'_{t,1}}^{(i)} & \cdots & {M'_{t,t-1}}^{(i)} \\ \vdots & \cdots & \vdots \\ {M'_{r-1,1}}^{(i)} & \cdots & {M'_{r-1,t-1}}^{(i)} \end{bmatrix}}_{\mathbf{S}^{(i)}} \cdot \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_t \end{bmatrix} = - \underbrace{\begin{bmatrix} {M'_{t,0}}^{(i)} \\ \vdots \\ {M'_{r-1,0}}^{(i)} \end{bmatrix}}_{\mathbf{T}^{(i)}} . \tag{4.7}$$

Note that $\sigma(x)$ is the same for all syndromes since every row has the same error positions. For collaborative decoding we concatenate the LSEs from each row in the following way to build a new LSE

$$\underbrace{\begin{bmatrix} \mathbf{S}^{(1)} \\ \mathbf{S}^{(2)} \\ \vdots \\ \mathbf{S}^{(\ell)} \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_t \end{bmatrix} = - \underbrace{\begin{bmatrix} \mathbf{T}^{(1)} \\ \mathbf{T}^{(2)} \\ \vdots \\ \mathbf{T}^{(\ell)} \end{bmatrix}}_{\mathbf{T}} . \tag{4.8}$$

Solving the key equation (4.6) with $i = 1, \dots, \ell$ for $\sigma(x)$ can be seen as a *Multi-sequence Linear Feedback Shift-Register (MLFSR) synthesis problem.* J. S. R. Nielsen proposed in [Nie13] using *Module Minimisation* to solve this problem, which is a quasi-linear methods with complexity $\mathcal{O}(\ell^3 r \log(r) \log(\log(r)))$, while solving the LSE (4.8) has complexity $\mathcal{O}(\ell^3 r^3)$.
Algorithm 4.2 summarizes the procedure of collaboratively decoding for interleaved Goppa codes.

Decoding Radius of Goppa Key Equation Decoder

Theorem 4.3.2 (Guaranteed Decoding Radius of Collaboratively Decoding)**.** *For an $\mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ with* $\deg(g) = r$*,* Algorithm 4.2 *guarantees to return a unique and correct codeword only if*

$$wt(\mathbf{E}) \leq \left\lfloor \frac{r}{2} \right\rfloor .$$

Proof. If $\mathrm{wt}(\mathbf{E}) \leq \lfloor \frac{r}{2} \rfloor$, collaborative decoding can be done by decoding each row of $\mathbf{R}$ seperately, which gives the unique and correct codeword, following from Theorem 4.3.1.

Algorithm 4.2: Collaboratively Decoding

```
Data: $\mathcal{I}\Gamma(\mathcal{L}, g, \ell)$, received word $\mathbf{R}$ and the number of errors $t$
Result: Correct codeword $\mathbf{C} \in \mathcal{I}\Gamma$ or decoding failure
1  foreach $\mathbf{r}^{(i)}$ do
2      Calculate the syndrome polynomial $\mathbf{s}^{(i)}(x)$;
3      Compute $\mathbf{S}^{(i)}$ and $\mathbf{T}^{(i)}$ in (4.7) according to (4.2) - (4.4);
4  Construct the LSE (4.8);
5  Do Gaussian Elimination to obtain the solution $(\sigma_1, \dots, \sigma_t)$ of (4.8);
6  Calculate ELP by $\sigma(x) = \sigma_t x^t + \cdots + \sigma_1 x + 1$;
7  foreach $\mathbf{s}^{(i)}(x)$ do
8      Calculate EEPs by $\eta^{(i)}(x) \equiv \sigma(x) s^{(i)}(x) \mod g(x)$;
9      Determine the error vector $\mathbf{e}^{(i)}$ by
           $e_i^{(i)} = \begin{cases} -\frac{\eta^{(i)}(\alpha_i)}{\sigma'(\alpha_i)} & \text{if } \sigma(\alpha_i) = 0 \\ 0 & \text{otherwise} \end{cases}, \quad i = 0, 1, \dots, n-1,$
       where $\sigma'(x)$ is the formal derivative of $\sigma(x)$.
10 Return
           $\mathbf{C} = \begin{pmatrix} \mathbf{r}^{(1)} - \mathbf{e}^{(1)} \\ \vdots \\ \mathbf{r}^{(\ell)} - \mathbf{e}^{(\ell)} \end{pmatrix}.$
```

□

Theorem 4.3.3 (Maximum Decoding Radius of Collaboratively Decoding)**.** *For an $\mathcal{I}\Gamma(\mathcal{L}, g, \ell)$ with $\deg(g) = r$, the maximum decoding radius of* Algorithm 4.2 *is*

$$wt(\mathbf{E}) \leq t_{\max} = \left\lfloor \frac{\ell}{\ell+1} \cdot r \right\rfloor. \tag{4.9}$$

Proof. From (4.7), we can see each $\mathbf{S}^{(i)}$ gives $r - t$ rows. To find a solution from the LSE (4.8) with t unknowns, the matrix $\mathbf{S}$ must at least have t rows. Thus, $t \leq \ell(r - t)$. However, we can find a unique solution of the LSE (4.8) only if $\operatorname{rank}(\mathbf{S}) = t$. This is not guaranteed when $\operatorname{wt}(\mathbf{E}) > \lfloor \frac{r}{2} \rfloor$. □

Corollary 4.3.2. *Consider a wild IGC $\mathcal{I}\Gamma(\mathcal{L}, g^{q-1}, \ell)$, where $\deg(g^{q-1}) = r$. The maximum decoding radius of* Algorithm 4.2 *for $\mathcal{I}\Gamma(\mathcal{L}, g^{q-1}, \ell)$ is*

$$wt(\mathbf{E}) =\leq t_{\max} = \left\lfloor \frac{\ell}{\ell+1} \cdot \frac{q}{q-1} \cdot r \right\rfloor.$$

Proof. Follows from Theorem 2.3.2 and Theorem 4.3.3 □

4.4 Probability of Decoding Failure

We can see from Theorem 4.3.2 and 4.3.3 that when $\left\lfloor \frac{r}{2} \right\rfloor < t = \mathrm{wt}(\mathbf{E}) \leq \left\lfloor \frac{\ell}{\ell+1} \cdot r \right\rfloor$, the decoder can not guarantee to return a correct codeword. When $\mathrm{rank}(\mathbf{S}) < t$, the decoder may give a wrong solution or multiple solutions. We define both consequences as decoding failures.

In following we present a bound on failure probability of decoding interleaved Goppa codes and show some simulation results of the failure probability of our decoder when decoding wild IGC for $\left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor < t \leq \left\lfloor \frac{\ell}{\ell+1} \cdot \frac{q}{q-1} \cdot r \right\rfloor$.

4.4.1 Bounds on Probability of Decoding Failure

In [SSB09], Schmidt, Sidorenko and Bossert came up with an upper bound on the failure probability of decoding *interleaved Reed-Solomon* (IRS) codes, in which a codeword is composed of ℓ codewords of an RS code. Since Goppa codes are subspace subcodes of RS codes, we will investigate whether the bound of decoding IRS codes is also the bound for decoding IGC.

Theorem 4.4.1 (Failure Probability of Decoding IRS Codes)**.** *Assuming that all error matrices $\mathbf{E} \in \mathbb{F}_{q^m}^{\ell \times n}$ with t non-zero columns are equally likely, the failure probability of collaboratively decoding interleaved Reed-Solomon codes can be bounded from above by*

$$P_f(\ell, t) \leq \left(\frac{(q^m)^\ell - \frac{1}{q^m}}{(q^m)^\ell - 1} \right)^t \cdot \frac{(q^m)^{-(\ell+1)(t_{\max}-t)}}{q^m - 1}, \tag{4.10}$$

where $t_{\max} = \frac{\ell}{\ell+1} \cdot r$.

From the simulation results in Figure 4.2 - 4.7 we can see that the failure probability of decoding IGC when the error matrices are over $\mathbb{F}_q$ violates the bound for IRS codes.

Sidorenko derived a failure probability of decoding interleaved alternant code in [Sid18].

Theorem 4.4.2 (Failure Probability of Decoding Interleaved Alternant Codes)**.** *Assuming that all error matrices $\mathbf{E} \in \mathbb{F}_q^{\ell \times n}$ with t non-zero columns are equally likely, the failure probability of decoding an interleaved $[n, n-\mu r]_q$, $1 \leq \mu \leq m$ alternant code can be upper-bounded by*

$$P_f(\ell, t) \leq \left(\frac{q^\ell - \frac{1}{q^m}}{q^\ell - 1} \right)^t \cdot \frac{q^{-(\ell\mu+m)(t_{\max}-t)}}{q^m - 1},$$

where $t_{\max} = \frac{\ell\mu}{\ell\mu+m} r$.

Since the effect of μ on the failure probability is not clear, we plotted all the bounds calculated by $\mu = 1, \ldots, m$ in Figure 4.2 - 4.7 to verify the correctness of the bound in Theorem 4.4.2.

4.4.2 Simulation Results

The simulations are done with wild IGC. Algorithm 4.3 shows how the simulations were operated.

Algorithm 4.3: Simulation Procedure

```
Data: Code parameters $(r, m, q, \ell)$ s.t. $q-1|r$
Result: Decoding Failure Probability
1 Construct an $\mathcal{I}\Gamma(\mathcal{L}, g^{q-1}, \ell)$, where $\mathcal{L} = \mathbb{F}_{q^m}\backslash\{0\}$ and $\deg(g^{q-1}) = r$;
2 Calculate $t_{\min} = \left\lfloor \frac{q}{q-1} \cdot \frac{r}{2} \right\rfloor$ and $t_{\max} = \left\lfloor \frac{\ell}{\ell+1} \cdot \frac{q}{q-1} \cdot r \right\rfloor$;
3 foreach $t \leftarrow t_{\min}$ to $t_{\max}$ do
4     for $i \leftarrow 0$ to iterations do
5         Generate an error matrix $\mathbf{E}$ with $t$ non-zero columns;
6         Decode $\mathbf{R} = \mathbf{C} + \mathbf{E}$ ($\mathbf{C} \in \mathcal{I}\Gamma$) using Algorithm 4.2 and obtain $\hat{\mathbf{C}}$;
7         if $\hat{\mathbf{C}} \neq \mathbf{C}$ then
8             fail+ = 1
9     Calculate $P_f(\ell, t) = \frac{\texttt{fail}}{\texttt{iterations}}$.
```

In order to investigate the effects of error value and error pattern on the failure probability, we chose four kinds of error matrices:

- $\mathbf{E} \in \mathbb{F}_q^{\ell \times n}$ with t non-zero columns;
- $\mathbf{E} \in \mathbb{F}_q^{\ell \times n}$ with t non-zero columns and being full-rank, i.e., $\text{rank}(\mathbf{E}) = \min\{\ell, t\}$;
- $\mathbf{E} \in \mathbb{F}_{q^m}^{\ell \times n}$ with t non-zero columns;
- $\mathbf{E} \in \mathbb{F}_{q^m}^{\ell \times n}$ with t non-zero columns and being full-rank.

We executed the simulations with parameters $(r, m, q) = (6, 7, 2)$ and $(r, m, q) = (14, 4, 3)$ for $\ell = 2, \ldots, 10$. Since the simulation iterations are 2123, it returns 0 when the failure probability is lower than 4.7×10^{-4}. We chose the results with non-zero experimental failure probability to present here, that are, $(r, m, q) = (6, 7, 2)$ for $\ell = 2, 3, 5$ and $(r, m, q) = (14, 4, 3)$ for $\ell = 2, 3, 6$.
Figure 4.2, 4.3 and 4.4 present the simulation results of $(r, m, q) = (6, 7, 2)$ for $l = 2$, $l = 3$ and $l = 5$ respectively. The first four entries in the label box represents the experimental failure probability. The red curve "th. IRS (q^m)" is calculated by the theoretical bound

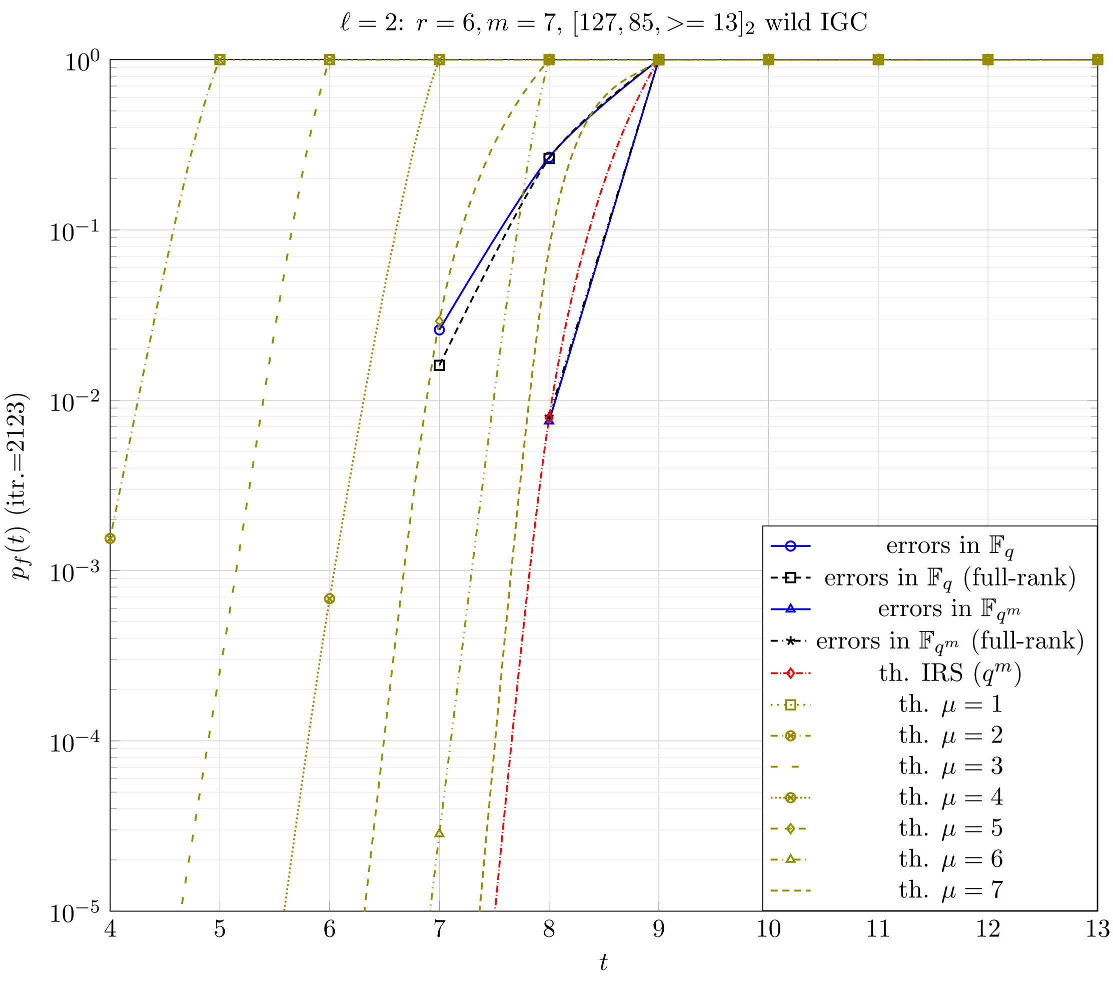


Figure 4.2: Failure probability of collaboratively decoding $[127, 85, >= 13]_2$ wild IGC for $l = 2$. The red curve "th. IRS (q^m)" is calculated from Theorem 4.4.1. The last 7 olive curves are drawn according Theorem 4.4.2.

inTheorem 4.4.1. The olive curves (the last 7 labels) are drawn according Theorem 4.4.2.

Figure 4.5, 4.6 and 4.7 present the simulation results of $(r, m, q) = (14, 4, 3)$ for $l = 2$, $l = 3$ and $l = 6$ respectively.

From the figures we can see that restricting the errors to be full-rank slightly reduces the failure probability. If the error value is in the extension field $\mathbb{F}_{q^m}$, the failure probability reduces significantly to the failure probability of decoding IRS codes. This indicates that the value of errors has more impact on the rank of $\mathbf{S}$ in (4.8).

It can be seen that neither of the bound presented in Theorem 4.4.1 or 4.4.2 matches the simulation results. A new bound of decoding IGC should be developed, which is left for

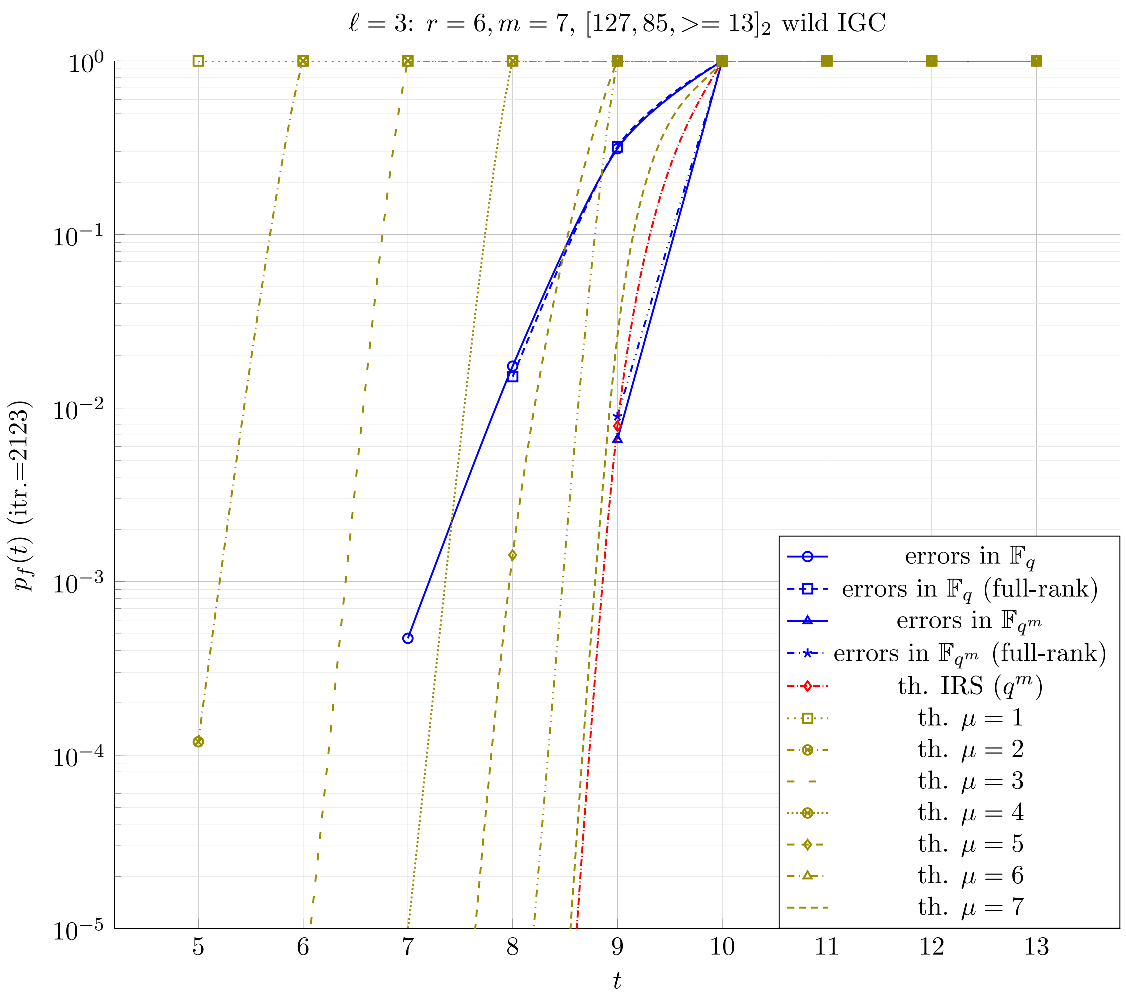


Figure 4.3: Failure probability of collaboratively decoding $[127, 85, >= 13]_2$ wild IGC for $l = 3$.

work in future.

4.5 Implementations of Goppa Codes and Decoders In SageMath

4.5.1 `GoppaCode` Class

A class of Goppa codes was implemented in SageMath [The18] (referred as Sage). Sage is a free open-source mathematics software system. There are abundant libaries supporting research in algebra, finite fields, coding theory and related areas.

The class `GoppaCode` is defined in `Goppa_code.py` which is appended in Appendix (only

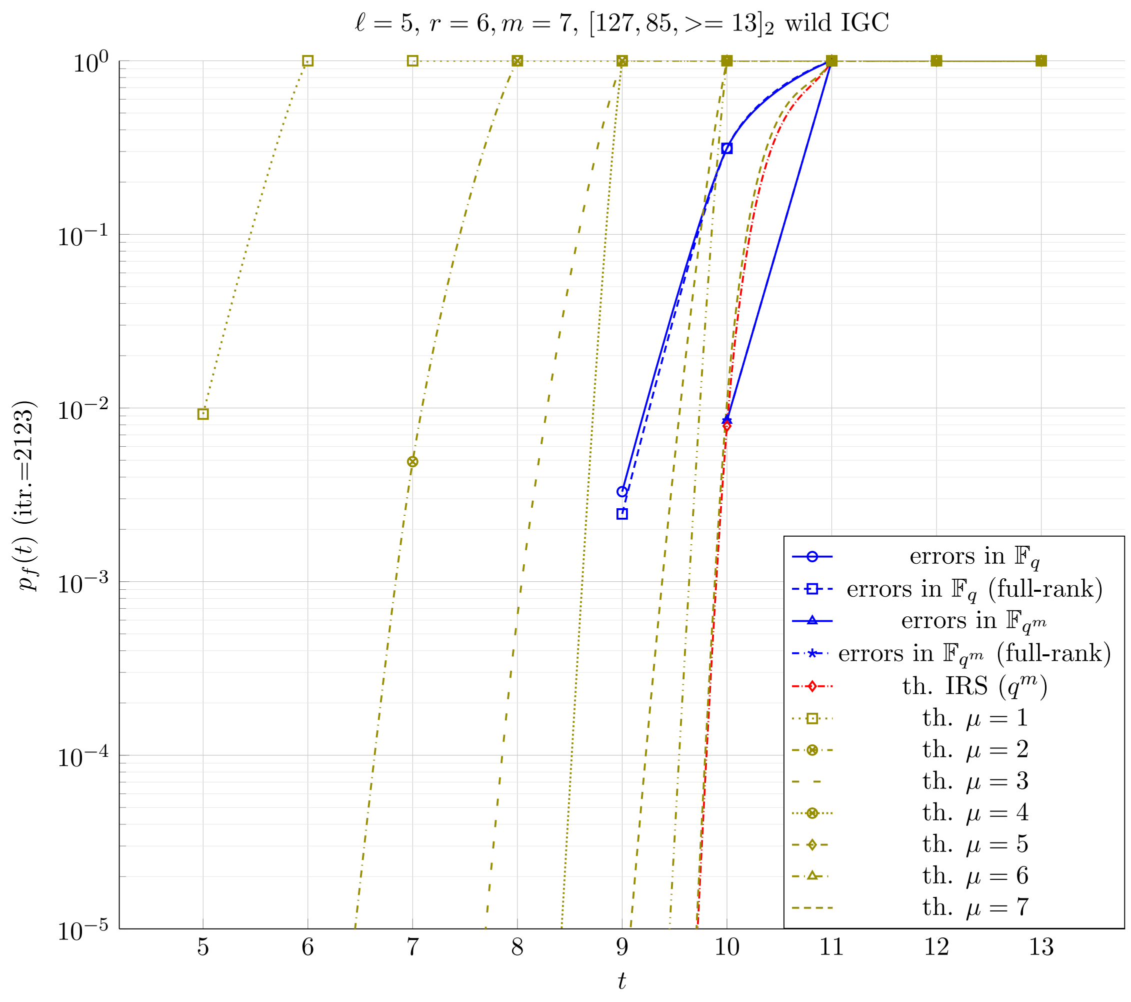


Figure 4.4: Failure probability of collaboratively decoding $[127, 85, >= 13]_2$ wild IGC for $l = 5$.

in electronic version) (the source code can be also found in our public reposity `https://bitbucket.org/LiaLiu0128/decodinginterleavedgoppacodes/src/master/`). A Goppa code object can be initialized by a tuple $(\mathcal{L}, g)$. For example, by

```
L = GF(2^3).list()
x = polygen(GF(2^3))
g = x^2 + x + 1

GC = GoppaCode(L, g)
```

an $[8, 2, 5]$ binary Goppa code object is constructed in Sage. In this example $g(x) = x^2+x+1$ is an irreducible polynomial over $\mathbb{F}_{2^3}$. There is an advantage to use an irreducible

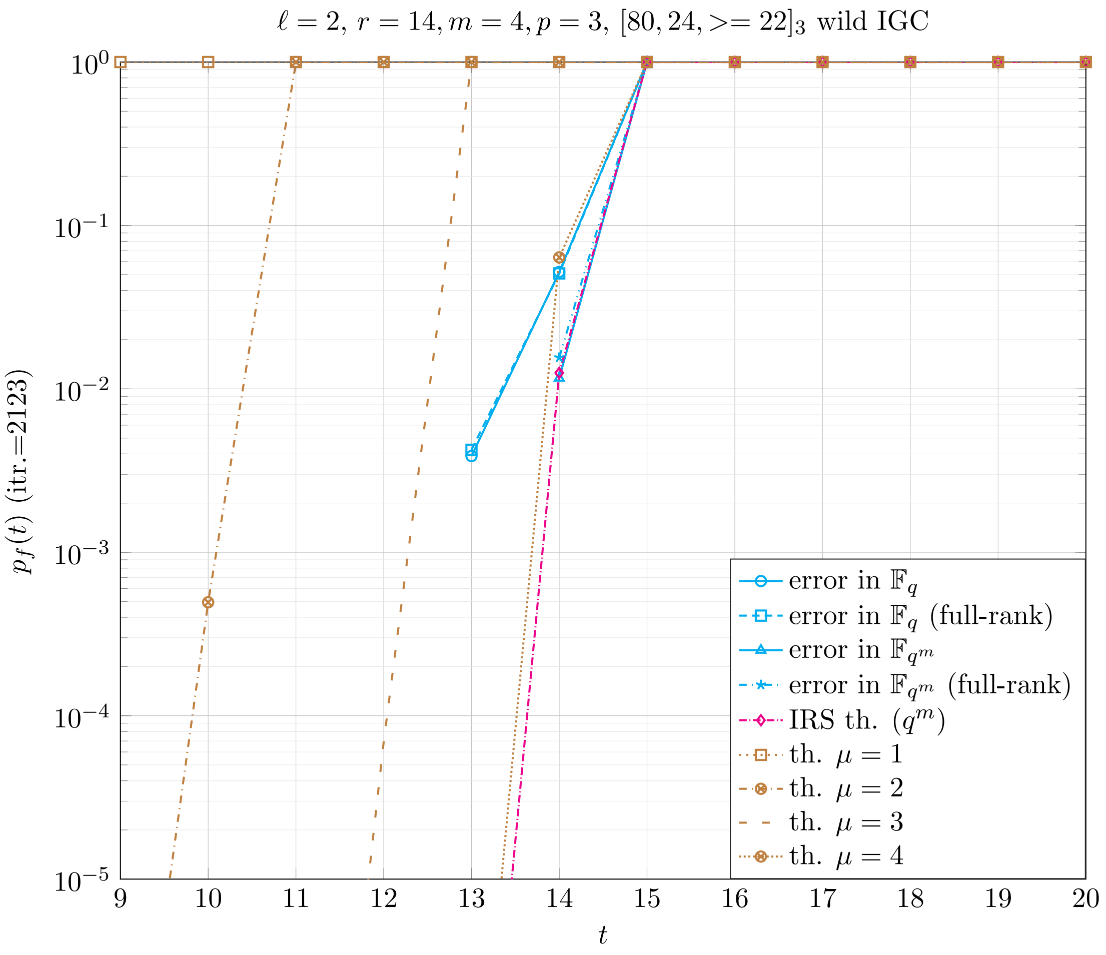


Figure 4.5: Failure probability of collaboratively decoding $[80, 24, >= 22]_3$ wild IGC for $l = 2$.

polynomial over $\mathbb{F}_{q^m}$ to construct Goppa codes. Because no root of the irreducible polynomial lies in $\mathbb{F}_{q^m}$, we can then take $\mathcal{L} = \mathbb{F}_{q^m}$ and obtain an $[n = q^m, k \geq q^m - rm, d \geq r+1]$ Goppa code over $\mathbb{F}_q$.

The parameters, parity check matrices $\tilde{\mathbf{H}}$ and $\mathbf{H}_{\text{Goppa}}$ (in (2.3)) and the $\mathcal{GRS}$ supercode can be accessed by calling the attributes of the object.

```
# continue the previous example
n = GC.length()
k = GC.dimension()
d = GC.minimum_distance()
H_ext = GC.parity_check_matrix_ext()
H_base = GC.parity_check_matrix()
```

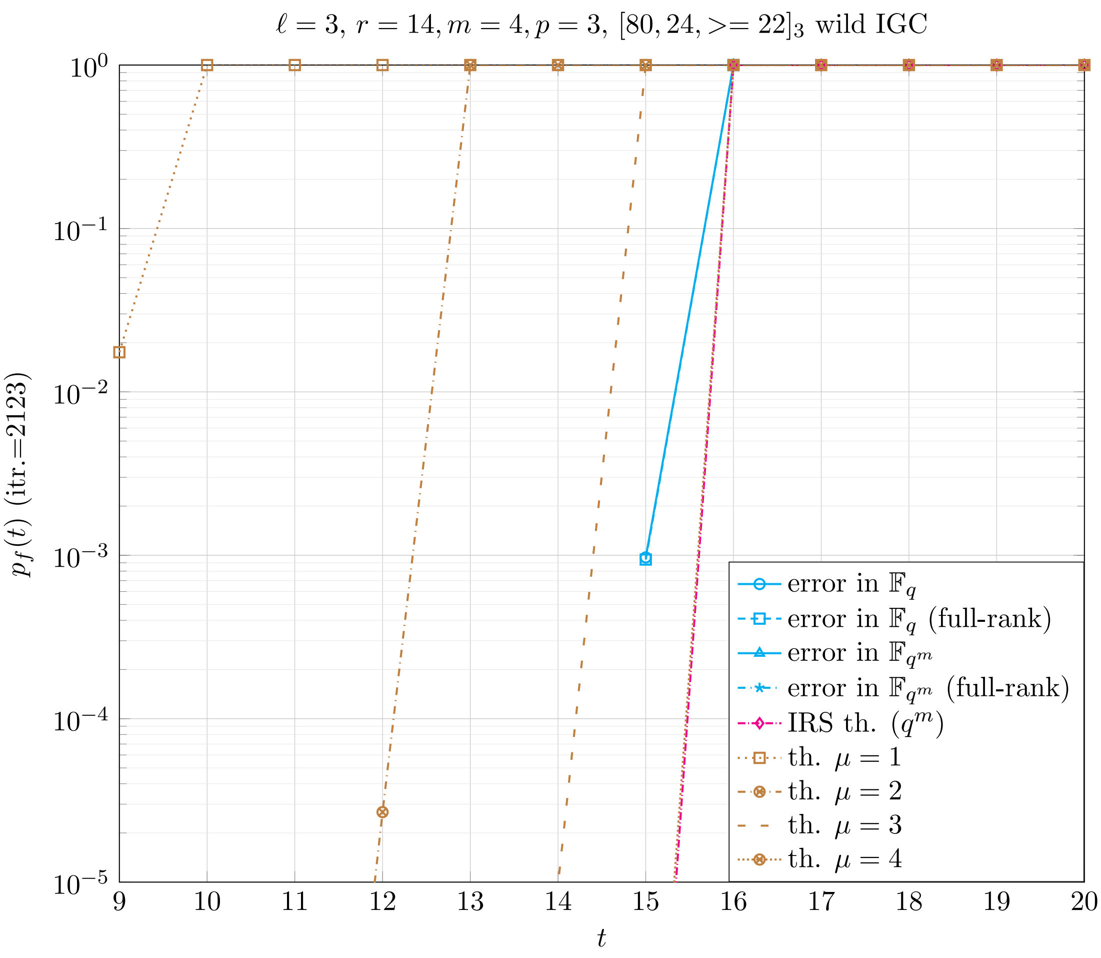


Figure 4.6: Failure probability of collaboratively decoding $[80, 24, >= 22]_3$ wild IGC for $l = 3$.

```
RS = GC._RS()
```

The results of `H_ext` and `H_base` are shown in (2.5) and (2.6) respectively.
The reason of including the $\mathcal{GRS}$ supercode as an attribute of GoppaCode class is that we can then directly use the well-developed decoders of the *GeneralizedReedSolomonCode* class in Sage.

4.5.2 Decoder Classes

Several decoder classes have been implemented in Sage by the author as tools for investigating decoding algorithms of Goppa codes and IGC in `Goppa_code.py`. These decoder classes are designed for the `GoppaCode` class. Some of them corresponds to the decoding algorithms explained in previous sections. For convenience of usage, we give the mappings

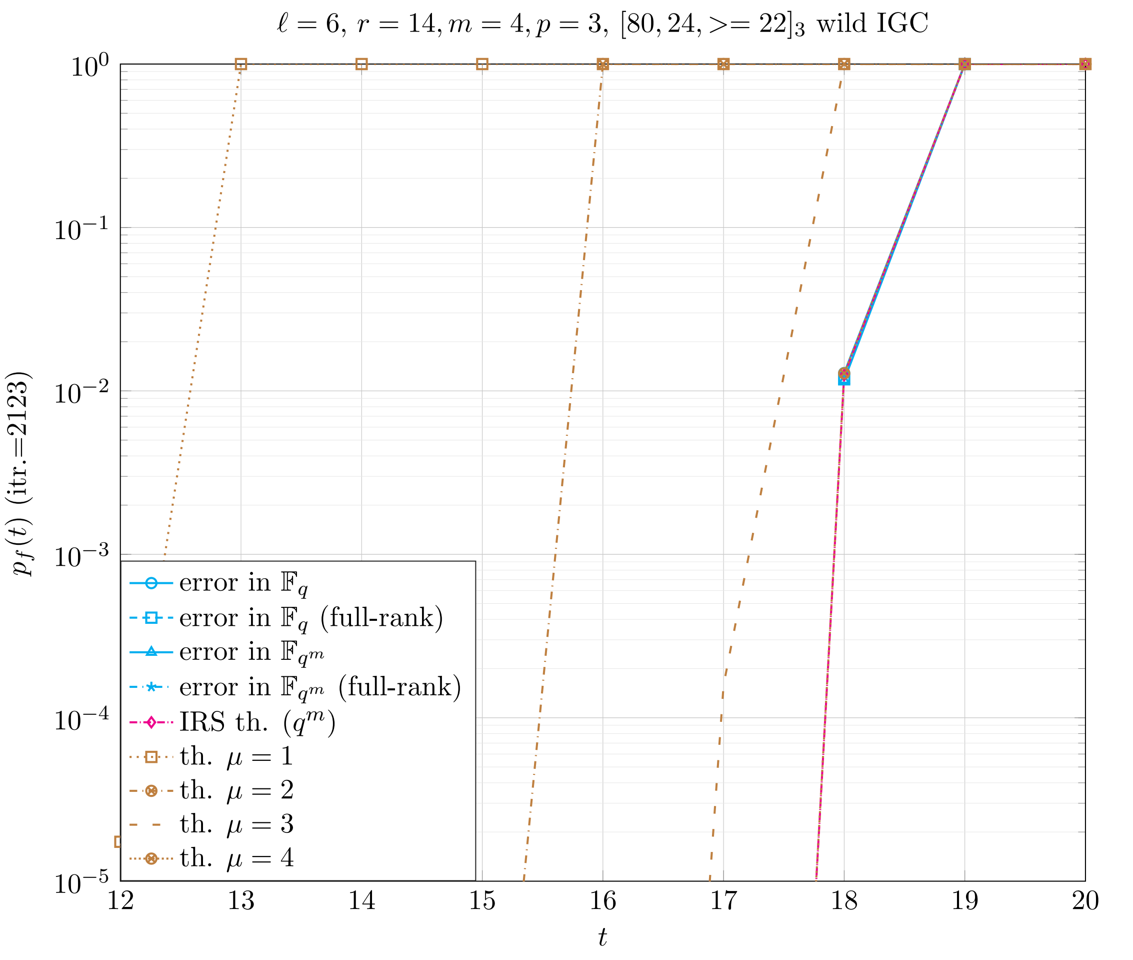


Figure 4.7: Failure probability of collaboratively decoding $[80, 24, >= 22]_3$ wild IGC for $l = 6$.

between the decoder classes and the corresponding sections.

1. `RSSyndromeDecoder` - GRS Syndrome-based Decoder (Section 2.4.1)
2. `GoppaSyndromeDecoder` - Goppa Key Equation Decoder (Section 4.3.1)
3. `PattersonDecoder` - Patterson Decoder (Section 2.4.2)
4. `GeneralizedPattersonDecoder` - p-ary Patterson Decoder (Section 2.4.3)
5. `RSSuperCodeDecoder`[1]
6. `IntGoppaSyndromeDecoder` - Collaboratively Decoding (Section 4.3.2)

[1]RS supercode decoder makes use of the existing decoder of the supper `GeneralizedReedSolomonCode` to decode `GoppaCode`.

7. `IRSDecoder`[2]

The `RSSyndromeDecoder` employs *Peterson-Gorenstein-Zierler algorithm* for $\Lambda(x)$ and the `GoppaSyndromeDecoder` sets up LSE (4.2)-(4.4) for $\sigma(x)$. Besides, we retain an alternative approach, so-called *lattice reduction*, for both of them. Lattice reduction works as the following: put the GRS syndrome polynomial $S(x)$ (2.9) or Goppa syndrome polynomial $s(x)$ (2.13) in a polynomial matrix

$$\begin{bmatrix} 1 & S(x) \\ 0 & x^{d-1} \end{bmatrix} \text{ or } \begin{bmatrix} 1 & s(x) \\ 0 & g(x) \end{bmatrix}.$$

Then, bring the lattice spanned by the rows of the matrix above into its *weak Popov form* via the functions in *Johan_lib.py* in Appendix (only in electronic version) (this can be also found in our public repository). *Johan_lib.py* is part of the *codinglib* [Nie17] implemented by Johan S. R. Nielsen. The left-most entry of the row, whose *leading position*[3] is 0, is the desired $\Lambda(x)$ or $\sigma(x)$. For further interests in lattice reduction and its application in decoding algorithms, please refer to [MS03, Nie13].
A good implementation of Patterson decoder has been done by Roering in [Roe13], from which several functions are used in the `PattersonDecoder` class.
The source code of the decoders is also contained in *GoppaCode.py.*
In the following some examples of how to use the decoders are given.

```
# given a Goppa code GC, a received word r and the number of errors t
c1 = GC.decode_to_code((r,t), "RSSyndromeDecoder")
c2 = GC.decode_to_code((r,t), "GoppaSyndromeDecoder")
c3 = GC.decode_to_code(r, "PattersonDecoder") # default decoder
c4 = GC.decode_to_code(r, "GeneralizedPattersonDecoder")
# c4 is a list of possible codewords
c5 = GC.decode_to_code(r, "RSSuperCodeDecoder")
```

The default decoder of `GoppaCode` is `PattersonDecoder`, namely, `GC.decode_to_code(r)` is equivalent to `GC.decode_to_code(r, "PattersonDecoder")`.
`IntGoppaSyndromeDecoder` and `IRSDecoder` are two decoders implemented for collaboratively decoding interleaved Goppa codes. For a received word `R = C + E`, where each row of `C` is a codeword of Goppa code `GC` and `E` has `t` non-zero column, decoding of `R` can be done by

```
C1 = GC.decode_to_code((R,t), "IntGoppaSyndromeDecoder")
```

[2]IRS decoder deploys the Algorithm 4.2 but uses RS syndrome $S(x)$, ELP $\Lambda(x)$ and EEP $\Omega(x)$.
[3]The index of the entry that has the greatest degree.

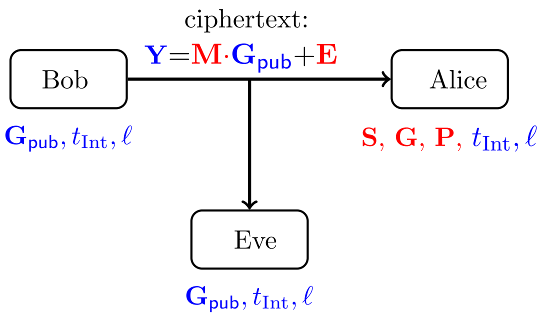
```
C2 = GC.decode_to_code((R,t), "IRSDecoder").
```

5 Interleaved McEliece System

A new variant of McEliece system with *interleaved Goppa codes* (IGC) was proposed by Elleuch, Wachter-Zeh and Zeh in [EWZZ18]. In this chapter we modify the vairant with wild Goppa codes. The new system will be denoted by *IntMcEliece* in further use. We first give a formal system discription for the setup and then discuss the attacks faced by IntMcEliece. Finally we show the key size of the new IntMcEliece at 80, 128, 256-bits security compared to wild McEliece.

5.1 IntMcEliece: System Description

Our proporsed system is an instantiation of the McEliece public-key cryptosystem using a wild ℓ-interleaved Goppa code $\mathcal{I}\Gamma(\mathcal{L}, g^{q-1}, \ell)$ over $\mathbb{F}_q$ with parameters $[n, k \geq n - mr, d \geq \frac{q}{q-1} \cdot r + 1]_q$.
Figure 5.1 illustrates the IntMcEliece.


ciphertext:
$\mathbf{Y}=\mathbf{M}\cdot\mathbf{G}_{\text{pub}}+\mathbf{E}$
Bob
Alice
Eve
$\mathbf{G}_{\text{pub}}, t_{\text{Int}}, \ell$
$\mathbf{S}, \mathbf{G}, \mathbf{P}, t_{\text{Int}}, \ell$
$\mathbf{G}_{\text{pub}}, t_{\text{Int}}, \ell$

Credit: *[EWZZ18]*

Figure 5.1: Key and information distribution of IntMcEliece.

Similar to McEliece system, the IntMcEliece consists of three part: **Key Generation**, **Encryption** and **Decryption**.
First, the key generation is done by Alice.

Key Generation (Alice):

- Choose parameters:

- A prime power q;
- Integers m, n, r such that $q-1|r$, $rm \leq n \leq q^m$ and $\frac{n-rm}{n}$ is noticeably distinct from 0 and 1;
- Locators set $\mathcal{L} = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\} \subseteq \mathbb{F}_{q^m}$ (i.e. n distinct elements from $\mathbb{F}_{q^m}$);
- Irreducible monic polynomial $g(x) \in \mathbb{F}_{q^m}[x]$ of degree $\frac{r}{q-1}$ such that $g(\alpha_i) \neq 0, \forall \alpha_i \in \mathcal{L}$;
- Integer ℓ and calculate $t_{\text{Int}} = \left\lfloor \frac{\ell}{\ell+1} \cdot \frac{q}{q-1} \cdot r \right\rfloor$;

• Generate the following matrices:
  - $\mathbf{S} \in \mathbb{F}_q^{k \times k}$: random non-singular matrix, called *scrambler matrix*;
  - $\mathbf{G} \in \mathbb{F}_q^{k \times n}$: generator matrix of $\Gamma(\mathcal{L}, g^{q-1})$;
  - $\mathbf{P} \in \mathbb{F}_q^{n \times n}$: random permutation matrix;

• Compute $\mathbf{G}_{\text{pub}} = \mathbf{SGP}$;

• Define the *key pair*:
  - Public key: $(\mathbf{G}_{\text{pub}}, t_{\text{Int}}, \ell)$;
  - Private key: $(\mathbf{S}, \mathbf{G}, \mathbf{P})$.

Compared to the variant of McEliece in [EWZZ18], which firstly proposed to use IGC as the secret code, IntMcEliece releases the restriction $q > 2$ in choosing parameters. The restriction was because for $q = 2$ the decoder of interleaved codes cannot decode more errors than the Patterson decoder which can correct up to r errors uniquely for binary Goppa codes. In this new IntMcEliece, we consider wild Goppa codes. Binary square-free Goppa codes are actually also wild Goppa codes so that we can collaboratively decode binary interleaved Goppa code more than Patterson decoder with high probability.

Second, Bob is responsible for the encryption. Rather than encoding one message and adding one random error vector of weight t, in the new system Bob needs to encode ℓ messges and add ℓ error vectors whose non-zero positions form a basis of an "error code".

Encryption (Bob):

Input: $\mathbf{G}_{\text{pub}}$, t_{Int}, ℓ
Output: Ciphertext $\mathbf{Y} \in \mathbb{F}_q^{\ell \times n}$

- Form a matrix $\mathbf{M} \in \mathbb{F}_q^{\ell \times k}$ with plaintexts;
- Generate error matrix:
  - Choose a code $\mathcal{C}_{\text{E}}$ of length t_{Int}, dimension $\geq \ell$ over $\mathbb{F}_q$ s.t. its minimum distance d_{E} is the largest among all possibilities;
  - Choose a basis of $\mathcal{C}_{\text{E}}$ to form a matrix $\mathbf{E}_{\text{basis}} \in \mathbb{F}_q^{\ell \times t_{\text{Int}}}$;
  - Generate a zero matrix $\mathbf{E} \in \mathbb{F}_q^{\ell \times n}$;
  - Randomly replace t_{Int} columns of $\mathbf{E}$ by the columns of $\mathbf{E}_{\text{basis}}$;
- Encryption: $\mathbf{Y} = \mathbf{M}\mathbf{G}_{\text{pub}} + \mathbf{E}$.

In our new IntMcEliece, Bob has to generate the errors from a basis of a code rather than randomly choosing a matrix with t_{Int} non-zero columns. This is to prevent the security level from being reduced too much by Stern's attack (see Algorithm 5.2).

At last, the decryption is executed by Alice.

Decryption (Alice):
Input: $\mathbf{S}, \mathbf{G}, \mathbf{P}$, ciphertexts $\mathbf{Y} \in \mathbb{F}_q^{\ell \times n}$
Output: Plaintexts $\mathbf{M} \in \mathbb{F}_q^{\ell \times k}$;

- Inverse permutation: $\mathbf{R} = \mathbf{Y}\mathbf{P}^{-1} = \mathbf{M}\mathbf{S}\mathbf{G} + \mathbf{E}\mathbf{P}^{-1}$;
- Decoding: $\mathbf{M}' = \mathcal{D}(\mathbf{R})$, where $\mathcal{D}(\cdot)$ is an efficient decoding algorithm for the ℓ-interleaved Goppa code up to t_{Int} errors;
- Inverse scramble $\mathbf{M} = \mathbf{M}'\mathbf{S}^{-1}$.

One can use the collaboratively decoding (see Section 4.3.2) as $\mathcal{D}(\cdot)$. However, as investigated in Section 4.4, it is possible that decoding fails, even if the error matrix is full-rank. Nevertheless, in the simulation results presented in Section 4.4, the full-rank error matrices are arbitrary full-rank matrices. Whether using a basis of a code could further reduce the failure probability of collaboratively decoding is left for future work.

5.2 Attacks

5.2.1 Structural Attack

Metzner-Kapturowski Attack

Metzner and Kapturowski presented an algorithm in [MK90] that decodes efficiently *any interleaved code* if the interleaving order ℓ is equal to the number of error positions t_{Int} and the error matrix $\mathbf{E}$ has full rank, just by applying Gaussian elimination on the product of the parity-check matrix and the corrupted codeword.
This principle can therefore be applied as follows to break our system if $\ell \geq t_{\text{Int}}$ and $\mathbf{E}$ is full-rank:

- Calculate $\mathbf{H}_{\text{pub}}$ from $\mathbf{G}_{\text{pub}}$ such that $\mathbf{H}_{\text{pub}}$ has full rank and $\mathbf{G}_{\text{pub}} \cdot \mathbf{H}_{\text{pub}}^T = \mathbf{0}$ (i.e., $\mathbf{H}_{\text{pub}}$ is a parity-check matrix of the public code generated by $\mathbf{G}_{\text{pub}}$);
- Consider the ℓ ciphertexts $\mathbf{Y} \in \mathbb{F}_q^{\ell \times n}$. The rows $\mathbf{y}^{(i)}$ of $\mathbf{Y}$ are codewords of the public code corrupted by rows of a full-rank burst error.
- Decode all rows $\mathbf{y}^{(i)}$ with $\mathbf{H}_{\text{pub}}$ according to the algorithm from [MK90] which then provides the ℓ secret messages in cubic time.

This general decoding principle therefore provides an attack to our system if $\ell \geq t_{\text{Int}}$ and $\text{rank}(\mathbf{E}) = t_{\text{Int}}$. From a high-level point of view, decoding a "random" interleaved code is an easy problem for larger interleaving orders and if the burst errors has full rank.
Thus, for parameter selection, to avoid this attack, we should choose $\ell < t_{\text{Int}}$ and should keep ℓ "not too close" to t_{Int}. Because according to [MK90], it might be possible to search the whole solution space of the decoder from if $l < t_{\text{Int}}$. Straight-forward, searching the solution space would provide an attack of complexity $\mathcal{O}(n^3 q^{m(t_{\text{Int}}-\ell)})$ which does not decrease the security level if ℓ is significantly less than t_{Int}.
Fortunately, the error matrix $\mathbf{E}$ in our system cannot be full-rank if $\ell = t_{\text{Int}}$, due to the fact that the non zero columns of $\mathbf{E}$ come from a basis of a $[t_{\text{Int}}, \ell, d_{\text{E}}]_q$. If $\ell = t_{\text{Int}}$, $d_{\text{E}} \leq 1$, whence $\text{rank}(\mathbf{E}) = d_{\text{E}} \leq 1$. And we desire larger d_{E} to keep the security level large (see Section 5.2.2), so we would have to keep ℓ small to find a d_{E} as large as possible.
The Metzner-Kapturowski Attack is not a threat to our IntMcEliece.

Attacks on Wild Goppa Codes

Since we deploy wild Goppa codes in IntMcEliece and the $\mathbf{G}_{\text{pub}}$ is also part of the public key of a wild McEliece, the attacks against wild McEliece, *polynomial search attack* and *square code operation* mentioned in Section 3.4, can be also applied on IntMcEliece.

We stress that the selection of parameters are crucial to defend our system from these structural attacks. One should also follow the restrictions given in Section 3.4 when selecting the paramters.

5.2.2 Decoding Attack

Besides structural attacks, what eavesdroppers may want to do is to reveal plaintext from ciphertext without knowing any structure of the codes. Implementing this is equivalent to decode a random code. Information set decoding (ISD) is the most efficient algorithm decoding an arbitrary linear code.

Naive ISD Attack on IntMcEliece

The naive ISD attack against IntMcEliece works similarly as in Section 3.2.1 against the original McEliece system. The difference is substituting t by t_{Int}.

Algorithm 5.1: Naive ISD on IntMcEliece

```
Data: Generator matrix $\mathbf{G}_{\text{pub}} \in \mathbb{F}_q^{k \times n}$, ciphertext $\mathbf{Y}$, number of errors $t_{\text{Int}}$
Result: Secret plaintext $\mathbf{M}$
1 Choose an $\mathcal{I} \subset \{0, \dots, n-1\}$, $|\mathcal{I}| = k$ that has never been chosen;
2 Choose any row $\mathbf{y}$ of $\mathbf{Y}$;
3 if $\mathrm{wt}\Big(\Big(\mathbf{y}_{\mathcal{I}} \cdot \mathbf{G}_{\text{pub},\mathcal{I}}^{-1}\Big) \cdot \mathbf{G}_{\text{pub}} - \mathbf{y}\Big) = t_{\text{Int}}$ then
4     Return $\mathbf{M} = \mathbf{Y}_{\mathcal{I}} \cdot \mathbf{G}_{\text{pub},\mathcal{I}}^{-1}$.
5 else
6     Repeat from step 1;
```

The work factor of naive ISD on IntMcEliece is

$$W_{\text{ISD}} = k^3 \cdot \frac{\binom{n}{k}}{\binom{n-t_{\text{Int}}}{k}},$$

followed from Theorem 3.2.1.

Note that since the error vector in $\mathbf{y}$ is of weight more than half distance, it is possible that there are multiple sets $\mathcal{I}$ that fulfill the condition at step 3. One may need to check whether the $\mathcal{I}$ fulfills the condisiton for the other rows, which may increase the work factor of this naive ISD. Nevertheless, we assume the worst case for us that getting a wrong $\mathcal{I}$ is of low probability.

Finding Low-Weight Codewords (FLWCW) Attack

The naive ISD attack can be reduced to the FLWCW problem. Stern's algorithm [Ste89] is an algorithm for FLWCW.

Algorithm 5.2 shows how to use Stern's algorithm to attack IntMcEliece.

```
Algorithm 5.2: Stern's Attack on IntMcEliece
Data: Generator matrix $\mathbf{G}_{\text{pub}}$ of size $k \times n$, ciphertext $\mathbf{Y}$, number of errors $t_{\text{Int}}$
Result: Secret plaintext $\mathbf{M}$
1 Append $\mathbf{Y}$ to $\mathbf{G}_{\text{pub}}$ to form a new generator matrix $\mathbf{G}_E$ of an $[n, k+\ell, d_{\text{E}}]$ code $\mathcal{C}$;
      $$\mathbf{G}' = \begin{pmatrix} \mathbf{G}_{\text{pub}} \\ \mathbf{Y} \end{pmatrix}$$
2 $\mathcal{E} \leftarrow \varnothing$;
3 while $|\mathcal{E}| < t_{Int}$ do
4     Find a low-weight code $\mathbf{e}$ in the code $\mathcal{C}$ by Stern's algorithm [Ste89];
5     $\mathcal{E} \leftarrow \mathcal{E} \cup \text{supp}(\mathbf{e})$;
6 Set $\mathcal{I} = \{0, 1, \dots, n-1\} \backslash \mathcal{E}$;
7 Return $\mathbf{M} = \mathbf{Y}_{\mathcal{I}} \cdot \mathbf{G}^{-1}_{\text{pub},\mathcal{I}}$.
```

A *low-weight* codeword in step 4 means $d_{\text{E}} \leq \text{wt}(\mathbf{e}) < t_{\text{Int}}$.

Theorem 5.2.1 (Work Factor of Stern's Attack on IntMcEliece)**.** *The work factor of Algorithm 5.2 on an IntMcEliece with parameters $(n, k, t_{Int}, \ell, d_E)$ is*

$$W_{S,Int}(n, k, t_{Int}, \ell, d_E) \geq k^3 + FLWCW(n, k+\ell, d_E)$$

where $FLWCW(\cdot)$ is given in Lemma 3.2.1.

Remark. *The work factor is given with a lower bound rather than exact value is because when d_E error positions are revealed, there are still $t_{Int} - d_E$ error positions left to be found out. But the $t_{Int} - d_E$ is relatively much less than d_E, thus it shows much less effect on work factor than d_E. We omit the effort for $t_{Int} - d_E$ positions for the consideration of the worst case. Moreover it is also possible that the Stern's algorithm can give multiple low-weight codewords at one time.*

Subcode Attack

Tillich pointed out in [Til18] that another code $\mathcal{C}'$ spanned by the error matrix $\mathbf{E} \in \mathbb{F}_q^{n\times\ell}$ is a subcode of $\mathcal{C}$. The code $\mathcal{C}'$ is an $[n, \ell, d_{\text{E}}]_q$ code with $\text{supp}(\mathcal{C}') = t_{\text{Int}}$, where $\text{supp}(\mathcal{C}') = \{i \quad \text{s. t.} \quad c_i \neq 0,\ i = 0, \dots, n-1,\ \forall \mathbf{c} \in \mathcal{C}'\}$.

Definition 5.2.1 (Support of Subcode Problem). *Given a code $\mathcal{C}[n, k+\ell, t_{Int}]$, knowing that there is subcode $\mathcal{C}'$ of $\mathcal{C}$ such that $\dim(\mathcal{C}') = \ell$ and supp$(\mathcal{C}') = t_{Int}$, one wants to recover the support of $\mathcal{C}'$.*

Thus, finding low-weight codewords problem can be reduced to the *support of subcode problem* as discussed. This problem has been handled by Otmani and Tillich for the binary case in [OT11]. In fact, the critical problem concerned in Otmani and Tillich's attack is still finding low-weight codewords in a code. They applied Dumer's algorithm [Dum91], which is an equivalent algorithm to Stern's algorithm. So we can assume the *subcode attack* has the same work factor as the Stern's attack on IntMcEliece.
Fortunately, Otmani and Tillich's algorithm is exponential in nature and can be easily defeated when the rate of $\mathcal{C}$ is significantly larger than the rate of the subcode $\mathcal{C}'$, i.e., $\frac{k}{n} > \frac{\ell}{t_{\text{Int}}}$.

Lower Bound on the Work Factor of ISD-based Attacks on IntMcEliece

Bernstein, Lange and Peters proposed in [BLP11a] *ball-collision attack*, based on which they came up with a lower bound on the work factor of ISD-based attacks working in $\mathbb{F}_2$. A generalized lower bound for $\mathbb{F}_q$ is given in Theorem 3.2.2.
From the process of Stern's attack on IntMcEliece, one can see that the security level is determined by d_{E}, rather than t_{Int}. Thus, following from Theorem 3.2.2, Corollary 5.2.1 gives a lower-bound on the work factor of attacks on IntMcEliece.

Corollary 5.2.1 (Work Factor of Attacks on IntMcEliece). *The work factor W of ISD-based attacks on an IntMcEliece with parameters $(n, k, t_{Int}, \ell, d_E)$ over $\mathbb{F}_q$ is*

$$W \geq \left\{ \frac{1}{2}\binom{n}{d_E}\binom{n-k}{d_E-p}^{-1}\binom{k}{p}^{-\frac{1}{2}} (\log_2 q)^2 \ : \ 0 \leq p \leq \min\{d_E, k\} \right\}.$$

We will use this lower bound to calculate the security level of IntMcEliece in next section.

5.3 Analysis of IntMcEliece

5.3.1 Selection of $\mathbf{E}_{\text{basis}}$ in Encryption

In this section we will use some examples to show how to choose an "error code" $\mathcal{C}_{\text{E}}$.
Whether the "error code" scheme can improve the security level of IntMcEliece from the McEliece system depends on whether a q-ary linear codes with parameters $[n = t_{\text{Int}}, k = \ell, d = d_{\text{E}}]_q$ exists or not. As long as we fix the field size q, the degree of Goppa polynomial r and the interleaving order ℓ, we can calculate the error correction capability t_{Int} of the

ℓ-interleaved Goppa code. Then we just need to find largest minimum distance of a code over $\mathbb{F}_q$ of length t_{Int} and dimension ℓ.

Table 5.1 presents some examples of the largest possible choices of d_{E} given r and ℓ. The column t is the error correction capability of the original McEliece system with Goppa codes and t_{wild} is the error correction capability of wild McEliece. The “error code” length is calculated by $t_{\text{Int}} = \left\lfloor \frac{\ell}{\ell+1} \cdot \frac{q}{q-1} \cdot r \right\rfloor$ and the minimum distance d_{E} is found with the help of CodeTables [Gra07], which is a website providing the largest minimum distance of any linear code and the construction of such code.

q	r	t	t_{wild}	ℓ	t_{Int}	d_{E}
2	68	68	68	**2**	**90**	**60**
				3	102	58
				4	108	56
				5	113	57
				6	116	58
				7	119	59
				8	120	58
	138	138	138	**2**	**184**	**122**
				3	207	118
				4	220	116
				5	230	118
				6	236	119
				7	241	120
				8	**245**	**122**
				9	**248**	**122**
3	84	42	64	2	84	63
				3	94	64
				4	100	66
				5	105	69
				6	108	69
				7	**110**	**70**
				8	**112**	**70**
3	54	27	40	2	54	40
				3	60	41
				4	64	42
				5	67	43
				6	69	43
				7	70	43
				8	**72**	**44**
4	54	27	36	2	48	38
				3	54	40
				4	57	41
				5	**60**	**43**
				6	61	42
				7	**63**	**43**
				8	**64**	**43**
5	100	50	62	2	83	69
				3	93	75
				4	100	79
				5	104	81
				6	107	82
				7	109	82
				8	**111**	**83**
				9	112	82

Table 5.1: Examples of the error code $\mathcal{C}_{\text{E}}$ given q, r. The column t and t_{wild} are the error correction capabilities of the original McEliece system and wild McEliece respectively. ℓ is the interleaving order, t_{Int} is the capability of IntMcEliece and d_{E} is the largest minimum distance of a $[t_{\text{Int}}, \ell]_q$ code.

The bold lines are the parameters of $\mathcal{C}_{\text{E}}$ that achieve the best d_{E} for a certain pair of (q, r).

The error correction capability determines the security level of the McEliece system and its variants when the length and polynomial degree of the deployed Goppa code are fixed. But for IntMcEliece, considering the Stern's attack, it is d_{E}, rather than t_{Int}, that determines the security level of IntMcEliece.

From the table we can roughly say that with binary Goppa codes, the IntMcEliece *can not* increase the security level comparing to the original McEliece system. But for any other Goppa codes with $q > 2$, the error correction capability of IntMcEliece is larger than that of the original McEliece system or wild McEliece. Thus, we can speculate that IntMcEliece gives higher security level compared to the original or wild McEliece when $q > 2$.

If the t_{Int} is out of the capability of CodeTables, one can also use *Griesmer Bound* (for binary codes) or *Gilvert-Varshamov Bound* (for any q) to determine the existence of a linear code.

Theorem 5.3.1 (Griesmer Bound)**.** *The length of shortest binary linear code of dimension k and minimum distance d is*

$$n(k,d) \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil .$$

Theorem 5.3.2 (Gilbert-Varshamov Bound)**.** *Let n, k and d be integers satisfying $2 \leq d \leq n$ and $1 \leq k \leq n$. If*

$$\sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i < q^{n-k},$$

then there exists an $[n,k]$ linear code over $\mathbb{F}_q$ with minimum distance at least d.

5.3.2 Key Size of IntMcEliece

In this section, we will compare the $(n, k, t_{\mathrm{wild}})$ wild McEliece system and our proposed (n, k, t_{Int}) IntMcEliece in terms of the key size for some certain security level. The security level of IntMcEliece depends on the minimum distance d_{E} of the error code $\mathcal{C}_{\mathrm{E}}[t_{\mathrm{Int}}, \ell]_q$. Table 5.2 presents some parameters for typical cryptosystem security levels (SL) (80, 128, and 256 bits). The column R is the code rate. The bold font indicates the better scheme that achieves smaller key size.

For each parameter pair we compute the size of the public key in systematic form as $k \times (n-k)$. Note that we do not consider *semantic attack*, where an attacker can easily get the plaintext from the first k bits of the ciphertext. So only when proper padding and randomizing (so-called *CCA2-conversion*) are engaged, one could then use the systematic public-key to encode the randomized plaintext.

As we predicted in the last section, IntMcEliece can not achieve smaller key size than the original key size, since the minimum distance d_{E} of the error code $\mathcal{C}_{\mathrm{E}}$ is smaller than

SL [bits]	q	m	System	r	n	k	t_{wild} or $(\ell, t_{\text{Int}}, d_{\text{E}})$	R	Key size [Bytes]
80	4	6	Wild McEliece	54	1681	1357	36	0.81	109 917
			IntMcEliece		1288	964	$(5, 60, 43)$	0.75	**78 084**
	3	7	Wild McEliece	54	1653	1275	40	0.77	95 485
			IntMcEliece		1447	1069	$(8, 72, 44)$	0.74	**80 057**
128	2	12	**McEliece**	68	3250	2434	68	0.75	**248 268**
			IntMcEliece	68	3926	3094	$(2, 90, 60)$	0.79	321 776 $\star$
			IntMcEliece	80	3788	2828	$(3, 119, 68)$	0.75	339 360 $\star$
			IntMcEliece	77	3656	2732	$(2, 102, 68)$	0.75	315 546 $\star$
	3	8	wild McEliece	54	5094	4662	40	0.92	399 012
			IntMcEliece		4042	3610	$(8, 72, 44)$	0.89	**308 974**
			wild McEliece	84	2934	2262	63	0.77	310 476
			IntMcEliece		2586	1914	$(7, 110, 70)$	0.74	**254 824**
	4	6	wild McEliece	99	2493	1899	66	0.76	282 002
			IntMcEliece		1890	1296	$(8, 117, 82)$	0.69	**192 456**
	5	5	wild McEliece	100	2342	1842	62	0.79	267 312
			IntMcEliece		1593	1093	$(8, 111, 83)$	0.69	**158 617**
256	2	13	**McEliece**	138	6975	5181	138	0.74	**1 161 840**
			IntMcEliece	156	7815	5787	$(2, 207, 138)$	0.74	1 467 258 $\star$
			IntMcEliece	160	8002	5922	$(6, 274, 138)$	0.74	1 529 720 $\star$
			IntMcEliece	155	7769	5754	$(10, 281, 138)$	0.74	1 449 289 $\star$
		14	**McEliece**	138	7460	5528	138	0.74	**1 335 012**
			IntMcEliece		8929	6997	$(2, 184, 122)$	0.78	1 689 776 $\star$
	3	9	wild McEliece	84	18230	17474	63	0.96	2 769 563
			IntMcEliece		13642	12886	$(7, 110, 70)$	0.94	**1 930 052**
	4	7	wild McEliece	222	5515	3961	148	0.72	1 538 849
			IntMcEliece		4074	2520	$(9, 266, 195)$	0.62	**979 020**
	5	7	wild McEliece	100	19058	18358	62	0.96	3 729 772
			IntMcEliece		7949	7249	$(8, 111, 83)$	0.91	**1 472 770**
		6	wild McEliece	204	5413	4189	128	0.77	1 488 163
			IntMcEliece		4153	2929	$(7, 223, 156)$	0.71	**1 040 542**

Table 5.2: Key size of IntMcEliece and wild McEliece for 80, 128, 256-bits security level.

the error correction capability t of orginal McEliece system. But for any other $q > 2$, the IntMcEliece always gives smaller key size compared to wild McEliece.

For $q = 2$, instead of leaving $d_{\text{E}} < t$, we can also keep $d_{\text{E}} = t$ and increase r in IntMcEliece. The examples marked with $\star$ in Table 5.2 show the corresponding key size of these two options. We can see that by properly choosing the interleaving order, the r required for achieving a certain d_{E} could be optimized, wherein the key size could be reduced.

6 Conclusion

The McEliece system with Goppa codes, a promising candidate of post-quantum cryptosystem, has remained remarkably secure, after hundreds of papers on attacking and defending over 40 years. The encryption and decryption in the McEliece system are efficient. The biggest disadvantage of the McEliece system for implementations in practice is its key size. For fixed n and k, the secuirity level of the system depends on the error correction capability of the Goppa codes it uses. In other words, for a required security level, increasing the number of errors in the ciphertexts can effectively reduce the key size.

In this thesis, we investigated the properties of Goppa codes and wild Goppa codes. We developed a probabilistic collaborative decoder for interleaved Goppa codes (IGC), which can decode t_{Int} errors with high probability, where t_{Int} is more than half of minimum distance. For binary square-free Goppa codes, this decoder can decode more errors than the Patterson decoder. With the decoder, IGC can be used in the McEliece system to increase the number of errors that can be added. However, with more ciphertexts which are corrupted at the same positions, Stern's attack can reduce the security level dramatically if the errors are randomly generated.
We therefore proposed a new code-based public-key cryptosystem, so-called IntMcEliece, where the errors are from a basis of an "error code". The security level of IntMcEliece depends on the minimum distance d_{E} of the error code, rather than the number of error positions t_{Int}. Due to the limitation of finding a good error code over $\mathbb{F}_2$, IntMcEliece still can not beat the original McEliece to have smaller key size for a certain security level. Nevertheless, for any $q > 2$, IntMcEliece obtains much smaller key size than wild McEliece, where the wild Goppa codes are used as the secrect code.

Finally, we showed with some examples that for $q > 2$, the key size of IntMcEliece is 20% less than that of wild McEliece for a certain security level.

We close this thesis by summarizing the tasks that could be interesting topics for future study.

- Analyse the failure probability of the collaborative decoder for IGC precisely and develop a bound on the failure probability;
- Investigate whether full-rank error matrices that form a basis of a code can reduce the failure probability compared to non-full error matrices or full-rank matrices that do not a basis;
- Survey whether algorithms similar to Metzner-Kapturowski's algorithm that can decode general interleaved codes for small interleaving order but full-rank errors exist;
- Accurately analyse the work factor of ISD-based attacks on IntMcEliece and reform the bound on the work factor by taking the interleaving order ℓ into account;
- Combine other decoding methods, like list decoding, with wild Goppa codes or IGC to further increase the decoding radius.

Bibliography

[ABC11] D. Augot, M. Barbier, and A. Couvreur, "List-decoding of binary goppa codes up to the binary johnson bound," in *2011 IEEE Information Theory Workshop*, Oct 2011, pp. 229–233.

[BBPR18] P. Beelen, M. Bossert, S. Puchinger, and J. Rosenkilde, "Structural properties of twisted reed-solomon codes with applications to cryptography," *CoRR*, vol. abs/1801.07003, 2018. [Online]. Available: http://arxiv.org/abs/1801.07003

[Ben80] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *Journal of Statistical Physics*, vol. 22, pp. 563–591, May 1980.

[Ber68] E. Berlekamp, "Nonbinary bch decoding (abstr.)," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 242–242, March 1968.

[Ber73] ——, "Goppa codes," *IEEE Transactions on Information Theory*, vol. 19, no. 5, pp. 590–592, September 1973.

[Ber11] D. J. Bernstein, "List decoding for binary goppa codes," in *Coding and Cryptology*, Y. M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 62–80.

[BHNW13] P. Beelen, T. Hholdt, J. S. R. Nielsen, and Y. Wu, "On rational interpolation-based list-decoding and list-decoding binary goppa codes," *IEEE Transactions on Information Theory*, vol. 59, no. 6, pp. 3269–3281, 2013.

[BL17] D. J. Bernstein and T. Lange, "Post-quantum cryptography - dealing with the fallout of physics success," *IACR Cryptology ePrint Archive*, vol. 2017, no. 314, p. 314, 2017.

[BLP08] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the mceliece cryptosystem," in *Post-Quantum Cryptography*, J. Buchmann and J. Ding, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–46.

[BLP11a] ——, “Smaller decoding exponents: Ball-collision decoding,” in *Advances in Cryptology – CRYPTO 2011*, P. Rogaway, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 743–760.

[BLP11b] ——, “Wild mceliece,” in *Selected Areas in Cryptography*, A. Biryukov, G. Gong, and D. R. Stinson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 143–158. [Online]. Available: https://cr.yp.to/codes/wild-20100721.pdf

[BM73] E. Berlekamp and O. Moreno, “Extended double-error-correcting binary goppa codes are cyclic (corresp.),” *IEEE Transactions on Information Theory*, vol. 19, no. 6, pp. 817–818, November 1973.

[BML13] P. S. L. M. Barreto, R. Misoczki, and R. Lindner, “Decoding square-free goppa codes over fp,” *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6851–6858, 2013.

[Chi64] R. Chien, “Cyclic decoding procedures for bose- chaudhuri-hocquenghem codes,” *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, October 1964.

[COT17] A. Couvreur, A. Otmani, and J. Tillich, “Polynomial time attack on wild mceliece over quadratic extensions,” *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 404–427, 2017.

[Czy11] S. Czynszak, “Decoding algorithms of reedsolomon code,” Dissertation, School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, 10 2011. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:833161/FULLTEXT01.pdf

[Deu85] D. Deutsch, “Quantum theory, the church-turing principle and the universal quantum computer,” *Proc. Royal Society London Ser. A*, pp. 96–117, 1985.

[DK18] V. Dragoi and H. T. Kalachi, “Cryptanalysis of a public key encryption scheme based on qc-ldpc and qc-mdpc codes,” *IEEE Communications Letters*, vol. 22, no. 2, pp. 264–267, Feb 2018.

[Dum91] I. Dumer, “On minimum distance decoding of linear codes,” vol. 1, 01 1991, pp. 50–52.

[Elg85] T. Elgamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, July 1985.

[EWZZ18] M. Elleuch, A. Wachter-Zeh, and A. Zeh, “A public-key cryptosystem from interleaved goppa codes,” 2018.

[FL06] C. Faure and P. Loidreau, “A new public-key cryptosystem based on the problem of reconstructing p–polynomials,” in *Coding and Cryptography*, O. Ytrehus, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 304–315.

[FnO$^+$13] J. C. Faugère, V. G. U. na, A. Otmani, L. Perret, and J. P. Tillich, “A distinguisher for high rate mceliece cryptosystems,” *IEEE Trans. Inf. Theory,*, vol. 59, no. 1, p. 68306844, Oct. 2013.

[For66] G. Forney, “Generalized minimum distance decoding,” *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125–131, April 1966.

[FS09] M. Finiasz and N. Sendrier, “Security bounds for the design of code-based cryptosystems,” in *Advances in Cryptology – ASIACRYPT 2009*, M. Matsui, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 88–105.

[Gop70] V. Goppa, “A new class of linear error correcting codes,” *Problems of Information Transmission*, vol. 6, no. 3, pp. 207–212, 1970.

[Gop71] V. D. Goppa, “Rational representation of codes and (l,g) - codes,” *Problems of Information Transmission*, vol. 7, no. 3, pp. 223–229, 1971.

[GPT91] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, “Ideals over a non-commutative ring and their application in cryptology,” in *Advances in Cryptology — EUROCRYPT ’91*, D. W. Davies, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 482–489.

[Gra07] M. Grassl, “Bounds on the minimum distance of linear codes and quantum codes,” Online available at http://www.codetables.de, 2007, accessed on 2018-12-09.

[Gro96] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC ’96. New York, NY, USA: ACM, 1996, pp. 212–219. [Online]. Available: http://doi.acm.org/10.1145/237814.237866

[GZ61] D. Gorenstein and N. Zierler, “A class of error-correcting codes in p^m symbols,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 2, pp. 207–214, 1961. [Online]. Available: https://doi.org/10.1137/0109020

[JM78] R. J. Mceliece, "A public-key cryptosystem based on algebraic coding theory," vol. 44, 05 1978.

[KKL$^+$18] A. Kuznetsov, A. Kiian, M. Lutsenko, I. Chepurko, and S. Kavun, "Code-based cryptosystems from nist pqc," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, May 2018, pp. 282–287.

[LB88] P. J. Lee and E. F. Brickell, "An observation on the security of mceliece's public-key cryptosystem," in *Advances in Cryptology — EUROCRYPT '88*, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, and C. G. Günther, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 275–280.

[Leo88] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1354–1359, Sept 1988.

[Mas69] J. Massey, "Shift-register synthesis and bch decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, January 1969.

[MB09] R. Misoczki and P. S. L. M. Barreto, "Compact mceliece keys from goppa codes," in *Selected Areas in Cryptography*, M. J. Jacobson, V. Rijmen, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 376–392.

[Mer78] R. C. Merkle, "Secure communications over insecure channels," *Commun. ACM*, vol. 21, no. 4, pp. 294–299, Apr. 1978. [Online]. Available: http://doi.acm.org/10.1145/359460.359473

[MK90] J. J. Metzner and E. J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Transactions on Information Theory*, vol. 36, no. 4, pp. 911–917, July 1990.

[Moo17] D. Moody, "The ship has sailed: The nist post-quantum crypto "competition"," [Online], 2017. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/asiacrypt-2017-moody-pqc.pdf

[MS78] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, 1978.

[MS03] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *Journal of Symbolic Computation*, vol. 35, no. 4, pp. 377 – 401, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747717102001396

[MSOS92] V. M. SIDELNIKOV and S. O. SHESTAKOV, "On insecurity of cryptosystems based on generalized reed-solomon codes," *Discrete Mathematics and Applications*, vol. 2, pp. 439–444, 01 1992.

[MTSB13] R. Misoczki, J. Tillich, N. Sendrier, and P. S. L. M. Barreto, "Mdpc-mceliece: New mceliece variants from moderate density parity-check codes," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 2069–2073.

[Nie13] J. S. R. Nielsen, "Generalised multi-sequence shift-register synthesis using module minimisation," *CoRR*, vol. abs/1301.6529, 2013. [Online]. Available: http://arxiv.org/abs/1301.6529

[Nie17] ——, *Codinglib for Sage*, 2017. [Online]. Available: https://bitbucket.org/jsrn/codinglib/

[OT11] A. Otmani and J.-P. Tillich, "An efficient attack on all concrete kks proposals," in *Post-Quantum Cryptography*, B.-Y. Yang, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 98–116.

[Pat75] N. Patterson, "The algebraic decoding of goppa codes," *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 203–207, 1975.

[Per12] E. Persichetti, "Compact mceliece keys based on quasi-dyadic srivastava codes," *Journal of Mathematical Cryptology*, vol. 6, pp. 149–169, 10 2012.

[Pet60] W. Peterson, "Encoding and error-correction procedures for the bose-chaudhuri codes," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 459–470, September 1960.

[Pet10] C. Peters, "Information-set decoding for linear codes over fq," in *Post-Quantum Cryptography*, N. Sendrier, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 81–94.

[Pra62] E. Prange, "The use of information sets in decoding cyclic codes," *Information Theory, IRE Transactions on*, vol. 8, pp. 5 – 9, 10 1962.

[Pra18] A. Pradeep, "Information set decoding for mceliece cryptosystem over fq," Technical University of Munich, Research Internship's Report, Aug. 2018.

[PRWZ18] S. Puchinger, J. Renner, and A. Wachter-Zeh, "Twisted gabidulin codes in the gpt cryptosystem," 2018.

[Roe13] C. Roering, "Coding theory-based cryptography: Mceliece cryptosystems in sage," Thesis, College of Saint Benedict/Saint John's University, Aug. 2013.

[Rot06] R. M. Roth, *Introduction to Coding Theory.* Cambridge University Press, 2006.

[RS60] 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. [Online]. Available: http://dx.doi.org/10.1137/0108018

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[Sen00] N. Sendrier, "Finding the permutation between equivalent linear codes: the support splitting algorithm," *IEEE Transactions on Information Theory*, vol. 46, no. 4, p. 11931203, 2000.

[Sho94] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov 1994, pp. 124–134.

[Sid18] V. Sidorenko, "Failure probability for syndrome decoding interleaved alternant codes," *Personal Communication*, pp. 11–12, 2018.

[SKHN75] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Information and Control*, vol. 27, no. 1, pp. 87 – 99, 1975. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S001999587590090X

[SKHN76] ——, "Further results on goppa codes and their applications to constructing efficient binary codes," *IEEE Transactions on Information Theory*, vol. 22, no. 5, pp. 518–526, September 1976.

[SSB09] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Collaborative decoding of interleaved reed-solomon codes and concatenated code designs," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 2991–3012, July 2009.

[Ste89] J. Stern, "A method for finding codewords of small weight," in *Coding Theory and Applications*, G. Cohen and J. Wolfmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 106–113.

[The18] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 8.1.0)*, 2018, http://www.sagemath.org. [Online]. Available: http://www.sagemath.org/index.html

[Til18] J.-P. Tillich, "Personal communication," Aug. 2018.

[TZ75] K. Tzeng and K. Zimmermann, "On extending goppa codes to cyclic codes (corresp.)," *IEEE Transactions on Information Theory*, vol. 21, no. 6, pp. 712–716, November 1975.

[Wir88] M. Wirtz, "On the parameters of goppa codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1341–1343, Sept 1988.

[WZ17] A. Wachter-Zeh, "Coding theory in storage and network. chapter 1: Interleaved reed-solomon codes," Lecture Slides, Apr. 2017.

[WZPR18] A. Wachter-Zeh, S. Puchinger, and J. Renner, "Repairing the faure-loidreau public-key cryptosystem," *2018 IEEE International Symposium on Information Theory (ISIT)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/ISIT.2018.8437561