Code-based Cryptography

Kim J. E. Jääameri

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 31.12.2019

Supervisor

Prof. Camilla Hollanti

Advisor

Prof. Camilla Hollanti

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Kim J. E. Jääameri

Title Code-based Cryptography

Degree programme Master's Programme in Mathematics and Operations Research

Major Mathematics **Code of major** SCI3054

Supervisor Prof. Camilla Hollanti

Advisor Prof. Camilla Hollanti

Date 31.12.2019 **Number of pages** 44+2 **Language** English

Abstract

Large enough quantum computers could run Shor's algorithm to solve problems such as the integer factorization problem or the discrete logarithm problem in time frames unattainable by conventional computers. Popular implementations of public-key cryptography rely on brute force computations for such problems being inefficient. This threat has motivated the research of post-quantum cryptography, in which code-based cryptography, making use of error-correcting codes, is one promising research direction. The underlying hard problem in code-based cryptography is the syndrome decoding problem, which is known to be NP-complete and believed to resist brute force attack from quantum computers. The original code-based cryptosystem was suggested in 1978 by Robert McEliece and since then extensive analysis on it and later variants has been covered in the literature. Many of the proposed variants have been shown to be insecure due to lack of sufficient concealment of the code structure, which enables attackers to bypass the hard problem and exploit structural vulnerabilities instead. Twisted codes are a relatively new family of codes, with certain explicit constructions potentially being viable for implementation to code-based cryptography, since previously deployed attack are not directly applicable to them. This thesis surveys the literature on linear codes to present concepts necessary to describe twisted codes in their cryptographic environment. The feasibility of twisted codes in code-based cryptography is then discussed through experimental simulation results.

Keywords Error-correcting codes, Post-quantum cryptography, Code-based cryptography, Rank distance, Rank metric codes, Gabidulin codes, Twisted codes

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Kim J. E. Jäämeri

Työn nimi Koodipohjainen kryptografia

Koulutusohjelma Matematiikan ja operaatiotutkimuksen maisteriohjelma

Pääaine Matematiikka **Pääaineen koodi** SCI3054

Työn valvoja Prof. Camilla Hollanti

Työn ohjaaja Prof. Camilla Hollanti

Päivämäärä 31.12.2019 **Sivumäärä** 44+2 **Kieli** Englanti

Tiivistelmä

Riittävän suurella kvanttitietokoneella on teoriassa kyky hyödyntää Shorin algoritmia ja laskea nopeasti tiettyjä vaikeita ongelmia, kuten kokonaislukujen tekijöihinjako ja diskreetti logaritmi. Suositut julkisen avaimen salausmenetelmät nojaavat näiden ongelmien vaikeuteen ja ovat siksi alttiita kvanttilaskennalle. Tästä lähtökohdasta on tutkittu useita vaihtoehtoisia kvanttilaskennan kestäviä salausmenetelmiä. Yksi lupaavimmista tutkimussuunnista on virheenkorjauskoodeihin pohjautuva kryptografia, jossa perustana on laskettavuusteoriassa tunnettu NP-täydellinen ongelma: satunnaisen koodisyndrooman dekoodaus. Tämän ongelman uskotaan olevan vaativa myös kvanttitietokoneille. Robert McEliecen vuonna 1978 esittämää alkuperäistä koodipohjaista salausjärjestelmää on tutkittu paljon ja sille on esitetty lukuisia muunnelmia. Näissä puutteellinen peite esitettyjen muunnelmien koodirakenteille on osoittautunut yleiseksi haasteeksi. Salaisen koodirakenteen avulla voidaan ohittaa kryptografinen vaativa ongelma ja purkaa salattu viesti suoraan hyödyntämällä näitä rakenteellisia heikkouksia. Twist-koodit ovat tutkimuksessa suhteellisen uusi koodiluokka ja niistä on ehdotettu tiettyjä rakennelmia tarkasteltavaksi myös kryptografian suhteen. Näihin esiteltyihin rakennelmiin ei voida suoraan käyttää mitään olemassa olevia hyökkäyksiä. Tässä työssä käydään aluksi läpi alan kirjallisuutta liittyen koodipohjaiseen kryptografiaan. Tarkoituksena on kattaa riittävästi taustateoriaa, jotta voidaan kuvata sen avulla twist-koodien käyttö kryptografisessa salauksessa. Lopuksi tarkastellaa twist-koodien soveltuvuutta kokeellisten tulosten valossa.

Avainsanat Virheenkorjauskoodit, kvanttilaskennan kestävät salausmenetelmät, koodipohjainen kryptografia, rangietäisyys (*engl.* rank distance), rangimetriikkakoodit (*engl.* rank metric code), Gabidulin-koodit, twist-koodit

Preface

I would like to extend my most sincere gratitude to Prof. Camilla Hollanti from the Department of Mathematics and Systems Analysis at Aalto University for instructing and supervising my thesis. I am deeply indebted from the past year and a half, during which I have been supported by Prof. Hollanti's invaluable guidance and expertise. This thesis would not have been possible without her patience, encouragement and good nature. In addition to being constantly steered in the right direction, I had the pleasure of being introduced to her international colleagues and their leading research in the field of code-based cryptography.

In the same spirit, I would also like to thank Dr. Sven Puchinger from the Algebraic Coding Theory research group at the Technical University of Denmark; Dr. Julien Lavauzelle from Institut de Recherche Mathématique de Rennes and M.Sc. Julian Renner from the Institute for Communications and Engineering from the Technical University of Munich. Thank you for sharing your ingenious research ideas and granting me access to your SageMath libraries. Our fruitful discussions last summer were a key source of insight and inspiration for my thesis work.

I am also thankful to the faculty at Aalto University as well as my fellow students and friends. You created an outstanding learning environment and it was a great privilege to have had the opportunity to study with such brilliant people.

Finally, I must express my profound gratitude towards my family for their unfailing support throughout my years at Aalto University. My parents Kim and Marja were always understanding and reassuring. Lastly, a special thanks to my partner Elise, who has brought so much joy and enthusiasm to my master's studies.

Helsinki, 12.12.2019

Kim J. E. Jäämeri

Contents

Symbols and abbreviations

Symbols

$\boldsymbol{v}$	vector $\boldsymbol{v}$
$\mathbf{M}$	matrix $\mathbf{M}$
$\mathbf{M}^{-1}$	inverse matrix of $\mathbf{M}$
$\mathbf{M}^{\intercal}$	transpose of $\mathbf{M}$
$\mathbb{F}_q$	finite field with q elements, shortly $\mathbb{F}$
$\mathcal{C}$	linear code
$\mathcal{C}^{\perp}$	the dual code of $\mathcal{C}$
$\mathcal{D}$	decoding algorithm
$\hat{\mathbf{G}}_k$	public generator matrix of a Gabidulin code of rank k

Operators

$\sum_{i=a}^{b}$	sum over index i, going from a to b
$\boldsymbol{v}_1 \star \boldsymbol{v}_2$	component-wise product of vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$
$d_H(\boldsymbol{v}_1, \boldsymbol{v}_2)$	Hamming distance between vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$
$d_r(\boldsymbol{v}_1, \boldsymbol{v}_2)$	rank distance between vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$
$rk(\boldsymbol{v})$	rank weight of vector $\boldsymbol{v}$
$\binom{n}{k}$	binomial coefficients
$f : X \to Y$	the map f from domain set X to codomain set Y
$x \mapsto y, x \in X, y \in Y$	element-wise mapping of the element $x \in X$ to $y \in Y$
$\mathbf{M}^{[i]}$	$\mathbf{M}^{q^i}$
mod $g(x, \alpha)$	modulo w.r.t. polynomial g, which is a polynomial with variables x and α
$gdc(a, b)$	the greatest common divisor of integers a and b
$\Lambda_i(\cdot)$	Overbeck's Frobenius operator (also referred to as q-sum)

Abbreviations

RSA	Rivest–Shamir–Adleman (cryptosystem)
ECC	elliptic curve cryptography
SDP	syndrome decoding problem
NP	non-deterministic polynomial-time
GPT	Gabidulin-Paramonov-Tretjakov
XOR	exclusive or (digital logic gate)
MDS	maximum distance separable
MRD	maximum rank distance
NIST	National Institute of Standards and Technology

1 Introduction

1.1 Background

In recent decades, concern for post-quantum cryptography has given rise to a new field of research in the study of techniques for secure communication. The concern is that a sufficiently large quantum computer, if constructed, could run e.g. Shor's algorithm to break popular cryptographic systems such as RSA and ECC [52]. In other words, popular methods for cryptography have relied on the hardness of certain problems that are not, in theory, safe from being solved by means of quantum computing.

Alternatives that are believed to resist attacks by large quantum computers have been suggested in the literature [7]. Some of the most promising among these are the code-based cryptosystems using error-correcting algebraic codes. While the original purpose of such codes was to enable communication over noisy communication channels [24], the defining properties of error-correcting codes have proved equally convenient for applications in cryptography [7]. The European Commission funded study group on post-quantum cryptography has recommended this approach for long-term protection against quantum computers [2].

Code-based cryptography, originally suggested by Robert McEliece [33], is based on the known NP-hardness of the Syndrome Decoding Problem (SDP). This corresponds to decoding a general linear code. The private key decoding on the other hand is based on the linear code secretly having some structure to which an efficient decoding algorithm is known. The original scheme by McEliece uses binary Goppa codes masked as general linear codes and an algorithm by Nicholas Patterson for decoding [44].

1.2 Thesis Scope and Structure

The remainder of the thesis is structured as follows. We define some preliminary algebraic notions in Chapter 2. These are used in the general definition for linear codes, as well as some more specific code structures later on.

Chapter 3 introduces the reader to linear codes, the core of algebraic coding theory and the fundamental basis for the remainder of this thesis. Extending on the general definitions and discussion, Hamming codes are used to demonstrate a readily understood, clear-cut example and Reed-Solomon codes are subsequently presented to set the stage for the topics of Chapter 6.

The topic of Chapter 4 is the concept of implementing a rank metric for linear codes as opposed to the original Hamming metric. The chapter explores the key differences and parallels stemming from the rank metric and presents the original construction method for this family of rank metric codes.

Chapter 5 describes the cryptographic context for the algebraic linear codes covered in Chapters 3 and 4. Namely, the cryptosystem presented by McEliece as well as a similar construction by Gabidulin, Paramonov and Tretjakov (GPT) [13], designed for codes in the rank metric.

The goal in Chapter 6 is to then delve into the ongoing back and forth process of both finding new ways to attack the existing code structures as well as fixing them by adding additional structure to the code [20][21][40][42][46][30].

Last but not least, Chapter 7 concludes the thesis with closing remarks and discussion, as well as ideas for further research directions.

2 Algebraic Preliminaries

2.1 Extension Fields

Finite fields are a core preliminary for constructing linear codes and we often want to impose some further structure on them. This is especially true for rank metric codes, which are generally constructed over extension fields. Hence, extension fields will be introduced before defining linear codes. This is a well-known topic in algebra, for further reference see e.g. [25].

Definition 1. Let $\mathbb{F}_q$ be a finite field with q elements and let $f(x)$ be an irreducible polynomial of degree $m \geq 1$ over $\mathbb{F}_q[x]$. An *extension field* of degree m over $\mathbb{F}_q$ is defined as the triple $(\mathbb{F}_{q^m}, +, \cdot)$, where $\mathbb{F}_{q^m}$ is the set of all q-ary vectors of length m and $+$ and $\cdot$ are operations for vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{F}_{q^m}$ defined as:

- $\boldsymbol{v}_1 + \boldsymbol{v}_2$: component-wise addition of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$
- $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2$: the coefficients of the polynomial $\boldsymbol{v}_1(x) \cdot \boldsymbol{v}_2(x) \bmod f(x)$

An extension field can be easily generated using a *primitive element* of $\mathbb{F}_{q^m}$, an element such that the powers of that element generate all other non-zero elements. Such an element can be obtained by selecting $f(x)$ to be a primitive irreducible polynomial. Let us consider an illustrative example of an extension field over a primitive irreducible polynomial:

Example 1. Let $f(x) = x^3 + x + 1$ and set $f(\alpha) = 0 = \alpha^3 + \alpha + 1$. Then the powers of α generate the elements of the extension field $\mathbb{F}_{2^3}$ as follows:

Table 1: The powers of α

Powers of α	**Element**	**Binary vector**
α^0	1	[0 0 1]
α^1	α	[0 1 0]
α^2	α^2	[1 0 0]
α^3	$\alpha + 1$	[0 1 1]
α^4	$\alpha^2 + \alpha$	[1 1 0]
α^5	$\alpha^2 + \alpha + 1$	[1 1 1]
α^6	$\alpha^2 + 1$	[1 0 1]
$\alpha^7 = \alpha^0$	1	[0 0 1]

The final element of the extension field is the zero element, corresponding to the binary vector [0 0 0]. The addition and multiplication operations are easy to compute in this framework. For example $\alpha^4 \cdot \alpha^5 = (\alpha^2 + \alpha)(\alpha^2 + \alpha + 1) = \alpha^4 + 2\alpha^3 + 2\alpha^2 + \alpha$ and since $2 \equiv 0 \pmod 2$, this simplifies to $\alpha^4 + \alpha$ and since $\alpha^4 = \alpha^2 + \alpha$, we are left

with $\alpha^2 + 2\alpha$, which again is congruent to $\alpha^2 \pmod 2$.

2.2 Linearized Polynomials

An important class of polynomials in abstract linear algebra is defined directly from extension fields. Namely, *linearized polynomials* [36], defined as follows:

Definition 2. Let the elements from an extension field $\alpha_i \in \mathbb{F}_{q^m}$ be coefficients for monomials that are powers of q, so $x^{q^i}, i \in \{0, \ldots, n\}, n \in \mathbb{N}$. A polynomial from such components is known as a *linearized polynomial* $L(x)$ of degree n, when $\alpha_0 \neq 0$.

A straightforward construction for a linearized polynomial can be done as:

$$L(x) = \sum_{i=0}^{n} \alpha_i x^{q^i}. \tag{1}$$

One application for these is in code-based cryptography, where certain families of codes draw their structure from clever use of linearized polynomials.

2.3 The Schur Product

The Schur product (also known as the Hadamard product) is a distributive, associative and commutative operation for vectors of equal dimension [**?**].

Definition 3. For two vectors $v, w \in \mathbb{F}_q^n$ the *Schur product* is defined via component-wise multiplication as:

$$v \star w := [v_1 \cdot w_1, v_2 \cdot w_2, \ldots, v_n \cdot w_n].$$

From the above definition, we can later derive a Schur product for linear codes which has many interesting applications.

3 Linear Codes

Linear codes are a family of error-correcting codes and are an essential pre-requisite for topics to follow. A general code could technically admit to no specific structure and the only way to describe it would be to list the entire codebook. Hence, in this chapter we define the useful concept of a linear code and discuss its parameters and properties [29].

Definition 4. A q-ary *linear code* $\mathcal{C}$ of rank k and length n is a linear subspace of dimension k of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is a finite field with q elements, and q is a prime power.

Codes over the binary field $\mathbb{F}_2$ are known as *binary* codes. The vectors of $\mathcal{C}$ are called *codewords* and the *size* of the code, q^k, is the number of codewords in $\mathcal{C}$. A codeword has *Hamming weight* w_H equal to its number of non-zero coordinates. One standard metric for *distance* d between two codewords is the Hamming distance.

Definition 5. The *Hamming distance* d_H between two codewords c_1 and c_2 is the number of respective symbol positions in which the two codewords differ.

For example, the codewords [0 1 0 1 0 1] and [1 1 1 1 1 1] have a different coordinate in exactly three positions and therefore $d_H([0\ 1\ 0\ 1\ 0\ 1],[1\ 1\ 1\ 1\ 1\ 1]) = 3$. This distance between codewords is precisely the property which allows for two codewords to remain unambiguously different, and also to be observed as different codewords when we are able to correct sufficiently many of those errors via the code's error-correction capabilities.

For a linear code $\mathcal{C}$ then, the minimum (Hamming) distance $d(\mathcal{C})$ is defined as the *minimum distance* between two distinct codewords. A linear code of length n, dimension k and minimum distance d over a finite field of size q is denoted as an $[n,k,d]_q$ code.

Finally, the code $\mathcal{C}$ has *code rate* $R = \frac{k}{n}$, which is the ratio of the number of information symbols to the total number of data symbols. The $n-k$ symbols in the codeword are redundant, but allow for the desired error-correcting properties.

Definition 6. The *error-correction* parameter t is defined to be the number of errors in a linear code, up to which any number of errors may be corrected.

For linear codes over Hamming distance, the error-correction parameter t is equal to $\lfloor \frac{d_H-1}{2} \rfloor$, since this is precisely when a ball centered at any codeword $\boldsymbol{c} \in \mathcal{C}$ with radius t is restricted to being non-intersecting with other balls of radius t centered around any other $\boldsymbol{c}' \neq \boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$. Such balls are known as *Hamming spheres*. Consequently, if $\lfloor \frac{d_H-1}{2} \rfloor$ errors occurred during the transmission of a received codeword $\boldsymbol{w}$, then there exists a unique codeword $\boldsymbol{w}' \in \mathcal{C}$ with $d_H(\boldsymbol{w}, \boldsymbol{w}') \leq t$. Error-correction and detection for Hamming metric codes is based in these properties.

3.1 Generator and Parity Check Matrices

The *generator matrix* $\mathbf{G}$ of an $[n,k,d]_q$ code is a $k \times n$ matrix, in which the rows of $\mathbf{G}$ form a basis for the code. Consequently, the set of unique linear combinations of the rows in $\mathbf{G}$ are the codewords in the code. Hence, the generator matrix is a compact way to represent an otherwise extensive code. In standard form, the generator matrix is given as:

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}], \tag{2}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ matrix. From the standard form of the generator matrix, it is possible to construct the *parity check matrix* $\mathbf{H}$ for the code. Assuming $\mathbf{G}$ is in standard form, the parity check matrix is constructed as:

$$\mathbf{H} = [-\mathbf{P}^\intercal | \mathbf{I}_{n-k}], \tag{3}$$

where $\mathbf{P}^\intercal$ denotes the transpose of matrix $\mathbf{P}$. Codewords c in the code $\mathcal{C}$ must satisfy $c\mathbf{H}^\intercal = 0$. The parity check matrix also has further implications relating to the dual code, which is covered in the next chapter.

The above concepts may be visualized by a simple example.

Example 2. Let $q = 2$ and

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

We notice that $\mathbf{G}$ generates a set of codewords: $\{[0\ 0\ 0], [0\ 1\ 1], [1\ 0\ 1], [1\ 1\ 0]\}$ when multiplied by different binary vectors of length 2. For example, the codeword $[0\ 1\ 1]$ was obtained from $[0\ 1]\mathbf{G}$. Note that the sum of any two codewords is also a codeword in the binary field.

It is easy to see that each codeword differs from any other codeword in exactly 2 positions and therefore the minimum distance of the code is 2. The above observations imply that $\mathbf{G}$ is the generator matrix of a $[3, 2, 2]_2$ code. The parity check matrix for this example code would be

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

having dimensions $(n - k) \times n = 1 \times 3$. The matrix $-\mathbf{P}^\intercal$ in this example is $[1\ 1]$. It can be verified that $c\mathbf{H}^\intercal = 0\ \forall\ c \in \mathcal{C}$.

3.2 The Dual Code

The orthogonal complement of a code $\mathcal{C}$ can be described as the set of all vectors which are orthogonal to every vector in $\mathcal{C}$. This is known as the *dual code* of $\mathcal{C}$ and denoted by $\mathcal{C}^\perp$. Since $\mathcal{C}^\perp$ is also a linear subspace of the vector space $\mathbb{F}_q^n$, it is itself a linear code.

The code and the dual code are related via their parity check and generator matrices. Indeed, the parity check matrix of $\mathcal{C}$ is the generator matrix of $\mathcal{C}^\perp$ and vice versa: $\mathbf{G}(\mathcal{C}) = \mathbf{H}(\mathcal{C}^\perp)$.

3.3 The Hamming Code

Hamming codes are a popular class of binary linear codes invented by Richard W. Hamming in 1950 [35]. This class was designed to have both high minimum distance $d(\mathcal{C})$ and code rate $R(\mathcal{C})$ to produce efficient codes that still retain well-structured redundancy for error-correction.

The number of parity bits $m = n - k \geq 2$ is selected, from which the $[2^m - 1, 2^m - m - 1, 3]_2$ Hamming code is constructed. This is done by carefully choosing the error-correcting bits such that the XOR-operation over all the bit positions containing a 1 is 0.

Example 3. One of the most famous Hamming Codes is the $[7, 4, 3]_2$ code, which has the following generator matrix:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We can construct its parity check matrix as in (3):

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The design for having the highest possible code rate while maintaining a minimum distance d is characterized by the *Hamming bound*. This bound gives an important limitation to the efficiency of error-correcting codes and any code achieving the Hamming bound is said to be a *perfect code*. We now define the Hamming bound.

Definition 7. The *Hamming bound* is the limit for the maximum size $\mathcal{A}(n, d)$ of a q-ary linear code of length n and minimum distance d:

$$\mathcal{A}(n, d) \leq \frac{q^n}{\sum_{k=0}^{t} \binom{n}{k}(q-1)^k}, \tag{4}$$

where $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ is the maximum number of errors the code can correct.

Proof. Let $\mathcal{C}$ be a linear code capable of correcting $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors and let $\boldsymbol{c} \in \mathcal{C}$ be a codeword. The unambiguous error-correction then implies that balls, or *Hamming spheres* of radius t centered around each codeword $\boldsymbol{c}$ are non-intersecting. These Hamming spheres cover q-ary words of length n, which deviate from the centre at codeword $\boldsymbol{c}$ in at most t components.

These deviations for each component are chosen from $(q-1)$ possibilities and we are choosing up to t of the n components to deviate from $\boldsymbol{c}$. Hence there are a total of $\sum_{k=0}^{t} \binom{n}{k}(q-1)^k$ different possible words within each Hamming sphere.

We can then consider selecting $\mathcal{A}(n, d)$ of these Hamming spheres, that is, the maximum number of codewords in $\mathcal{C}$ and count the total number of words in the union of these spheres as $\mathcal{A}(n, d) \cdot \sum_{k=0}^{t} \binom{n}{k}(q-1)^k$. Clearly this number cannot exceed q^n and hence we have the the bound

$$\mathcal{A}(n, d) \leq \frac{q^n}{\sum_{k=0}^{t} \binom{n}{k}(q-1)^k}.$$

□

3.4 The Singleton Bound

The Singleton Bound, credited to Richard C. Singleton, gives a loose upper bound on the minimum distance of a linear code. This bound for a linear code $\mathcal{C}$ is often expressed as $d(\mathcal{C}) \leq n - k + 1$, where d, n and k are the minimum Hamming distance, the length, and dimension, respectively, of $\mathcal{C}$. A short proof for the bound goes as follows:

Proof. The generator matrix for any linear code can be expressed in the standard form given in Equation (2). One row in such a generator matrix consists of at most $n - k + 1$ non-zero elements, since there are $n - k$ elements in the rows of matrix $\mathbf{P}$ and only one non-zero element in the rows of the identity matrix. Therefore, the maximum weight of one row in the standard form generator matrix is $n - k + 1$ and this weight is precisely the Hamming distance to the zero codeword from any codeword generated by the row in the generator matrix. The bound follows from this observation. □

Linear codes satisfying equality with the Singleton bound are called Maximum Distance Separable (MDS).

3.5 Syndrome Decoding

The property for parity check matrices: $\boldsymbol{c}\mathbf{H}^\intercal = 0$ for all $\boldsymbol{c} \in \mathcal{C}$ allows us to do *syndrome decoding*. Suppose we receive a word $\boldsymbol{r}$. Then $\boldsymbol{r}\mathbf{H}^\intercal = 0$ if and only if $\boldsymbol{r}$ is a codeword in $\mathcal{C}$.

The received word $\boldsymbol{r}$ may be expressed as a sum of some codeword $\boldsymbol{c} \in \mathcal{C}$ and error vector $\boldsymbol{e}$ of the same length, such that $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$. Multiplying this type of received word $\boldsymbol{r}$ with $\mathbf{H}^\intercal$ gives us what we call the *error syndrome*, denoted by $\mathbf{S}$.

Note that

$$\begin{aligned}\mathbf{S} &= [\boldsymbol{c} + \boldsymbol{e}]\mathbf{H}^\intercal = \boldsymbol{c}\mathbf{H}^\intercal + \boldsymbol{e}\mathbf{H}^\intercal \\ &= \boldsymbol{e}\mathbf{H}^\intercal,\end{aligned}$$

since $\boldsymbol{c}\mathbf{H}^\intercal = 0$. The idea is then that each distinct $\mathbf{S}$ corresponds to a distinct error vector, depending on the parity check matrix of the code. This correspondence may be presented as a table, and for codes with small parity check matrices the table is relatively compact.

If the number of errors in the received word is at most the error-correcting capacity of the code t, then syndrome decoding yields the correct codeword and if $w_H(\boldsymbol{e}) > t$, then this type of decoding corresponds to *maximum likelihood* decoding. This equivalence assumes the probability of an error occurring at one given coordinate as < 0.5 and that the probability is independent of other errors. Under these assumptions, syndrome decoding yields the most probable codeword with respect to the to the received word.

The general case of syndrome decoding a linear code was shown to be an NP-hard problem by Berlekamp, McEliece, and van Tilborg in 1978 [6], when decoding a linear code in the presence of t errors without having access to a parity check matrix.

Example 4. Consider the $[7, 4, 3]_2$ Hamming code from Example 3. Let us receive a word $\boldsymbol{r} = [0\ 1\ 1\ 0\ 1\ 1\ 1]$ and apply syndrome decoding. We compute the error

syndrome as:

$$\mathbf{S} = \boldsymbol{r}\mathbf{H}^{\intercal} = [0\ 1\ 1\ 0\ 1\ 1\ 1] \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0\ 0\ 1]\,.$$

Since the syndrome is not equal to zero, there received word is not a codeword and we may deduce which single error would produce the syndrome [0 0 1]. The third row of the parity check matrix asserts that the first and last bits of any codeword are the same, and hence there should be an error in either position 1 or position 7 as the bit 1 in position 3 of the syndrome indicates this type of disparity.

If the error were in position 1, then we would have to correct the received word to $\boldsymbol{c}^* = [1\ 1\ 1\ 0\ 1\ 1\ 1]$. This, however, is not a codeword in the code, so we deduce that the error occurred in position 7 and $\boldsymbol{c}^* = [0\ 1\ 1\ 0\ 1\ 1\ 0]$ is indeed a codeword. Specifically, it is a linear combination of the second and third rows of the generator matrix. Hence, this would be the output of a syndrome decoder with input $\boldsymbol{r}$.

3.6 Reed-Solomon Codes

Irving S. Reed and Gustave Solomon introduced a class of linear codes in 1960, later to be known as *Reed-Solomon codes* [49]. The code has optimal minimum distance for a linear code of length n and rank k (i.e. it meets the Sinlgeton bound), and hence, is a famous and widely utilized code structure with applications such as bar codes and data storage in media. Similar to a general linear code, the Reed-Solomon code can be parametrized as an $[n, k, d]_q$ code and it can be constructed via a generator or parity check matrix. The formal definition of a general Reed-Solomon code is as follows:

Definition 8. Let $\mathbb{F}_q$ be a finite field and let p denote a polynomial of degree less than k over $\mathbb{F}_q$. Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$ denote $n \leq q$ distinct evaluation points for the polynomial p. Then a *Reed-Solomon code* is defined as $\mathcal{C}_{\mathcal{RS}} = \{(p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_n))\}$.

The evaluation points $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ are often selected such that α is a primitive element of $\mathbb{F}$ and $n \mid q-1$, as this allows us to use Vandermonde matrices for a simple and concise construction [57]:

$$\mathbf{G}_{RS} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ \vdots & \vdots & 1 & \ddots & \vdots \\ 1 & \alpha^{(k-1)} & \alpha^{2(k-1)} & \cdots & \alpha^{(n-1)(k-1)} \end{bmatrix},$$

$$\mathbf{H}_{RS} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{(n-k)} & \alpha^{(n-k)2} & \cdots & \alpha^{(n-k)(n-1)} \end{bmatrix}.$$

Messages of the form $\boldsymbol{m} = [m_0, m_1, \ldots, m_{k-1}], m_i \in \mathbb{F}_q$ can be encoded by mapping them to a polynomial of degree $(k-1)$. Here the mapping $\mathcal{M} : \boldsymbol{m} \to p(\alpha)$ is defined by $m \mapsto \sum_{i=0}^{k-1} m_i \alpha^i$.

Note that Reed-Solomon codes comply with the definition for linear codes, since for a any $b \in \mathbb{F}_q$ and polynomials $f(\alpha), g(\alpha) \in \mathbb{F}_q[\alpha]$ of degree $\leq k-1$, the polynomials $bf(\alpha)$ and $f(\alpha) + g(\alpha)$ are also polynomials of degree $\leq k-1$ and the mapping for message encoding clearly preserves this linearity.

The Singleton bound can also be verified for Reed-Solomon codes by recalling that a non-zero polynomial of degree x over $\mathbb{F}_q$ has at most x roots in $\mathbb{F}_q$, as the Hamming weight of the difference between two encoded messages, $wt(\mathcal{C}_{RS}(\boldsymbol{m}_2) - \mathcal{C}_{RS}(\boldsymbol{m}_1))$ is equal to n minus the number of zeroes in $\mathcal{C}_{RS}(\boldsymbol{m}_2) - \mathcal{C}_{RS}(\boldsymbol{m}_1)$. This number of zeroes is precisely the number of roots the polynomial $p_{m_2}(\alpha) - p_{m_1}(\alpha)$ has among the evaluation points: $k-1$. Thus, the weight of the difference for two encoded messages is at least $n-k+1$, which results in the Singleton bound.

4 Rank Error-correcting Codes

4.1 Background and Motivation

Rank metric codes are a special type of linear error-correcting code that use a rank based metric instead of the Hamming metric. The rank metric was introduced by Loo-Keng Hua in 1951 as an "arithmetic distance" [28],[14]. In 1978, Philippe Delsarte applied the rank metric on a set of bilinear forms, or rectangular matrices [12]. He also proposed the construction of optimal rank codes in bilinear form representation.

The rank metric for vector spaces over field extensions was introduced in 1985 by Ernst M. Gabidulin, who also described optimal rank metric codes in vector form and proposed fast encoding and decoding algorithms [14]. This was the only known construction for optimal rank codes until recently, as John Sheekey presented a more general family, including the construction by Gabidulin [51].

Useful applications for rank codes have been found in areas such as distributed storage [54] and random linear network coding [55]. In cryptography, extensive study of rank metric codes is motivated by the large public key sizes inherent to code-based cryptography. Conventional schemes based on problems such as factorization of integers or the discrete logarithm problem use much smaller public keys for communication compared to code-based approaches. Yet, rank codes provide a comparative advantage with respect to the McEliece type implementations using Hamming metric codes, such as Goppa codes. This property is due a large family of rank metric codes being generated from a fixed size public key.

4.2 The Rank Metric

We are now ready to define the *rank weight* of a vector and the *rank distance* between two vectors:

Definition 9. Let $\boldsymbol{v} = \{v_i\}, \boldsymbol{w} = \{w_i\} \in \mathbb{F}_{q^m}^n$ be two vectors of length n in the vector space over the extension field $\mathbb{F}_{q^m}$. The *rank weight* $rk(\boldsymbol{v})$ is defined as the dimension of the $\mathbb{F}_q$ subspace generated by $\{v_1, v_2, \ldots, v_n\}$.

The *rank distance* $d_r(\boldsymbol{v}, \boldsymbol{w})$ for two vectors is defined as $rk(\boldsymbol{v} - \boldsymbol{w})$.

The definition of rank distance further induces a notion of "minimum rank distance" to rank metric codes, similar for the minimum Hamming distance. Formally the minimum distance for a rank metric code $\mathcal{C}$ can be expressed as $\min_{\boldsymbol{v},\boldsymbol{w}\in\mathcal{C},v\neq w} d_r(\boldsymbol{v}, \boldsymbol{w})$.

The rank weight can be equivalently defined by extending every coordinate of a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ on a basis of the quotient field $\mathbb{F}_{q^m}/\mathbb{F}_q$ to form an $m \times n$ matrix over $\mathbb{F}_q$. The rank of such a matrix is equal to $rk(\boldsymbol{v})$. This definition is useful for computing the rank weight in practice.

Yet another equivalent definition is obtained from the dimension of the *support* of a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$. The support of $\boldsymbol{v}$ is the $\mathbb{F}_q$ vector space spanned by its coordinates. The dimension of the support of $\boldsymbol{v}$ is equal to $rk(\boldsymbol{v})$.

Example 5. Let $\boldsymbol{v} \in \mathbb{F}_{2^3}^5(\alpha)$ and fix $\boldsymbol{v} = (\alpha, \alpha^5, 0, \alpha, \alpha^6) = (\alpha, \alpha^2+\alpha+1, 0, \alpha, \alpha^2+1)$. Let us compute the rank weight $rk(\boldsymbol{v})$ of the vector v.

To this end, let us form a 3×5 matrix by extending the coordinates of v onto a basis of $\mathbb{F}_{q^m}/\mathbb{F}_q$. We obtain the following matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix},$$

where for example the second column $[1\ 1\ 1]^T$ corresponds to the binary vector [1 11], which corresponds to α^5 originating from the second coordinate of v. The first and third row are clearly linearly dependent, so the rank of this matrix is 2. Therefore according to the above discussion on equivalent definitions for rank weight, we have $rk(v) = 2$.

4.3 Maximum Rank Distance

At the beginning of this section we referred to "optimal" rank metric codes vaguely without a formal definition. There is a bound for rank metric codes, which is similar to the Singleton bound for linear codes with the Hamming metric.

Definition 10. Let $\mathcal{C}$ be a linear code with the rank metric over the vector space $\mathbb{F}_{q^m}^n$. If $n \leq m$ and $k \leq n$, then $d_r \leq n - k + 1$ and rank codes satisfying equality with this bound are defined to have *Maximum Rank Distance* (MRD).

An alternative bound for maximum rank distance was given by Pierre Loidreau in [31]:

$$d_r \leq \frac{m}{n}(n-k) + 1. \tag{5}$$

This alternative bound is tighter for the case where $n > m$, whereas for $n \leq m$, the former bound remains the tighter one. This is why the former bound is defined for $n \leq m$. Each maximum rank distance code is also maximum distance separable.

4.4 Gabidulin Codes

Rank error-correcting codes are sometimes referred to as simply Gabidulin codes due to the prevalence of the famous construction by Ernst Gabidulin [14]. Gabidulin codes are designed to be maximum rank distance codes with length $n \leq m$ over an extension field $\mathbb{F}_{q^m}$. Due to highly similar structure, this family of codes is often seen as an analogue to the Reed-Solomon codes from the Hamming metric. There are two straightforward approaches for constructing Gabidulin codes, the one via parity check matrix being as follows:

Definition 11. Let $h_1, h_2, \ldots, h_n \in \mathbb{F}_{q^m}$ be chosen such that they are linearly independent over the base field $\mathbb{F}_q$ and let s be a positive integer such that $gcd(s, m) = 1$. Then a *Gabidulin code* is defined by a parity check matrix of the form

$$\mathbf{H}_{d-1} = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \\ h_1^{q^s} & h_2^{q^s} & \cdots & h_n^{q^s} \\ h_1^{q^{2s}} & h_2^{q^{2s}} & \cdots & h_n^{q^{2s}} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{q^{(d-2)s}} & h_2^{q^{(d-2)s}} & \cdots & h_n^{q^{(d-2)s}} \end{bmatrix} \tag{6}$$

where $d = n - k + 1$ denotes the minimum distance of the code.

The other option is to construct a generator matrix for the code with a similar pattern.

Definition 12. Let $g_1, g_2, \ldots, g_n \in \mathbb{F}_{q^m}$ be chosen such that they are linearly independent over the base field $\mathbb{F}_q$ and let s be a positive integer such that $gcd(s, m) = 1$. Then a *Gabidulin code* is defined by a generator matrix of the form

$$\mathbf{G}_k = \begin{bmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{q^s} & g_2^{q^s} & \cdots & g_n^{q^s} \\ g_1^{q^{2s}} & g_2^{q^{2s}} & \cdots & g_n^{q^{2s}} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{q^{(k-1)s}} & g_2^{q^{(k-1)s}} & \cdots & g_n^{q^{(k-1)s}} \end{bmatrix} \tag{7}$$

where $k = n - d + 1$ denotes the dimension of the code.

The parameter s is often fixed to be 1, which simplifies the code quite a bit. Also, the substitution in notation of $q^i \to [i]$ is often used and makes these types of definitions easier to read.

Let us construct a simple Gabidulin code in another illustrative example.

Example 6. Set $s = 1$ and select from the extension field in Example 1 elements α, α^2 and $\alpha^2 + 1$. Following the construction for a generator matrix of a Gabidulin code gives:

$$\mathbf{G}_3 = \begin{bmatrix} \alpha & \alpha^2 & \alpha^2+1 \\ \alpha^2 & \alpha^2+\alpha & \alpha^2+\alpha+1 \\ \alpha^2+\alpha & \alpha & \alpha+1 \end{bmatrix}.$$

We can verify that the maximum rank distance bound of $d_r = 1 = 3 - 3 + 1$ holds for the example, although it may not be exciting for such a small example.

Note how the columns in the generator matrix of a Gabidulin code follow the structure for linearized polynomials defined in (1). We can express codewords in terms of linearized polynomials as $(L(g_1), L(g_2), \ldots, L(g_n))$, given that the polynomial $L(x)$ has degree less than k. The set of all such linearized polynomials $L(x)$ with degree less than k forms the set of codewords $\mathcal{C}$.

The proof to the following proposition makes use of this parallel and shows that all Gabidulin codes are MRD.

Proposition 4.1. *The Gabidulin code defined by* $\mathbf{G}_k$ *in* (7) *is MRD.*

Proof. Let $x = (L(g_1), L(g_2), \ldots, L(g_n))$ denote a codeword generated by $\mathbf{G}_k$ and associated to the linearized polynomial $L(x)$, with degree less than k. The rank of x is equivalent to the dimension of the $\mathbb{F}_q$-vector space generated by $(L(g_1), L(g_2), \ldots, L(g_n))$. Also, the $\mathbb{F}_q$-vector space generated by $(g_1, g_2, \ldots, g_n)$ has dimension n, as g_i were selected to be linearly independent.

Let $ker(L) = \{a \in \mathbb{F}_{q^m} | L(a) = 0\}$. The n-dimensional $\mathbb{F}_q$-vector space generated by $(g_1, g_2, \ldots, g_n)$ contains the dimensions of $ker(L)$, that are precisely not included in the $\mathbb{F}_q$-vector space generated by $(L(g_1), L(g_2), \ldots, L(g_n))$.

$L(x)$ having degree up to $k-1$ implies that $ker(L)$ must have dimension up to $k-1$ and therefore $rk(x) \geq n - (k-1) = n - k + 1$. Hence, the rank distance d_r for the code is bounded according to the definition of MRD codes. □

4.5 Error Correction

Errors in rank codes are evaluated with respect to the rank weight. This is not quite as straightforward as looking for errors in binary codes with the Hamming metric. Nevertheless, rank codes are known to have powerful error-correcting capabilities due to their structure.

Consider again the channel model where the received word $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ is a sum of a codeword and an error vector of the same length.

Definition 13. [14] We then say that a *rank error* of rank ϵ occurs if $rk(\boldsymbol{e} \mid \boldsymbol{e} \in \mathbb{F}_{q^m}^n) = \epsilon$.

Rank codes can generally correct error patterns of the following type:

$$\boldsymbol{e}_{total} = \boldsymbol{e}_{random} + \boldsymbol{e}_{row} + e_{col}, \tag{8}$$

where the total rank error is seen as a sum of entirely random errors, as well as column and row rank erasures. These error types are described in greater detail in [14]. The random errors are characterized as vectors of the form $\boldsymbol{e}_{random} = e_1u_1+e_2u_2+\ldots+e_tu_t$, where $e_i \in \mathbb{F}_{q^m}$ and $u_j \in \mathbb{F}_q$ are assumed to be linearly independent over $\mathbb{F}_q$ and u_i are assumed to have coordinates in $\mathbb{F}_q$. Any other information is presumed unknown, including the rank t.

The row and column erasures have similar vector form, except the row and column components replacing the u_i in the random error case are presumed as known to the decoder.

The following upper bound for the error-correction capacity of Gabidulin codes was given by Gabidulin and Pilipchuk in [18]:

Lemma 4.2. *An $[n, k, d_r = n - k + 1]$ MRD code may simultaneously correct v row erasures, r column erasures and ϵ random rank errors, given that*

$$2\epsilon + v + r \leq d_r - 1. \tag{9}$$

Proof. Consider any $n \times n$ square code matrix. Erasing v rows and r columns results in a $(n-v) \times (n-r)$ matrix code with rank distance at least $d'_r = d_r - v - r$. Hence, the code allows for the correction random rank errors of rank ϵ, if $\epsilon \leq \frac{(d_r-v-r-1)}{2}$ or, equivalently,

$$2\epsilon + v + r \leq d_r - 1.$$

□

5 Code-based Cryptography

Public-key cryptography is founded on the idea of having pairs of keys, consisting of widely distributed *public keys* as well as *private keys*, the access to which is restricted to the owner. The ideal model for secure communication is such that the *sender* encrypts a message using the *receiver's* public key. This action produces a *ciphertext*, which can be easily decoded using the receiver's private key.

This core idea is illustrated in Figure 1. The difficulty of decoding the ciphertext without a private key depends on the difficulty of the mathematical problem on which the cryptographic algorithm is based. Exhaustive searches, known as *brute force attacks* are generally inefficient and successful attacks tend to exploit the subtle structures in the key design.

The focus of this section is to introduce the concept of post-quantum cryptography and then present some of the cryptographic schemes using linear codes in their key design.

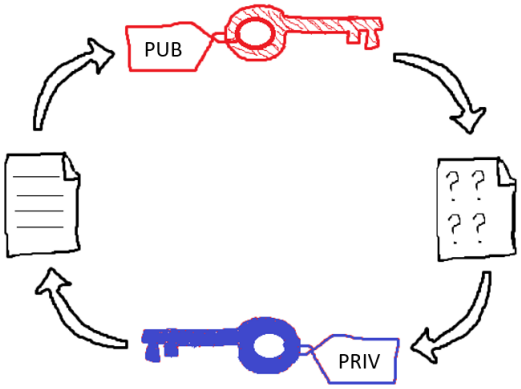

Figure 1: Public-key Cryptography

5.1 Post-Quantum Cryptography

In the present age, there is increasing demand for fast, flexible, preferably affordable and last but not least, secure communication schemes. Two of the first design criteria are met with ease via public-key cryptography, given that the implementation has a suitable design. These schemes are put to use by our phones when we send messages to our friends, by our internet browsers when we connect to (secure) websites and by our smart cards when they give our digital signature to other authentication devices.

Unfortunately, the most convenient and popular methods of public-key cryptography are directly threatened by Shor's algorithm running on a large enough quantum computer [52]. It makes use of the quantum Fourier transform to break down the periodicity of functions into approximate superpositions and then measuring them [9]. This process requires $2n + 3$ quantum bits or *qubits* and $\mathcal{O}(n^3 \log n)$ operations with them in the the variant of Shor's algorithm in [3]. These processes may also be set to run in parallel in order to deliver solutions to problems previously thought to be time consuming, and with exponential speed-up [7]. Such problems include the integer factorization problem, the discrete logarithm problem and addition of points on an elliptic curve, each of which underpins a popular conventional public-key encryption scheme.

Another well-known quantum algorithm to emerge in the 90s is Grover's algorithm [23]. It has a broader range of applications, but the potential speed-up for computation is not as striking as it is for Shor's algorithm. At the core of it, Grover's algorithm can search roots to a function by finding one root for every $\sqrt{N}$ quantum evaluations from a an input where roots occur one out of N inputs [9]. In theory, the threat of Grover's algorithm applies pressure directly to security parameters, even in cases where the cryptographic system has no explicit flaws. In practise though, it may turn out that quantum computers need to allocate resources to maintain stability in their qubits, and in that case Grover's algorithm may not be feasible as quantum computers are scaled up in size.

The appeal of code-based cryptography is a consequence of desiring to be secure against quantum computers. There are currently no known ways to use Shor's algorithm and deploy to it speed up the decoding of a general linear code and this

has stood test of time. Alternative directions to code-based cryptography include, for example lattice based approaches and multivariate polynomials [10]. With the research field being as dispersed as it is, there are also concerted efforts to standardize some form of post-quantum cryptosystem [1].

5.2 The McEliece Cryptosystem

Code-based cryptography originated from the scheme presented in the 1978 paper by Robert McEliece [33]. Based on then a recent proof that the syndrome decoding problem is NP-complete [6], the idea was to "scramble" and "permute" a generator matrix of an error-correcting linear code with matrix multiplications to make it indistinguishable from the generator matrix of a random linear code. This apparently random generator matrix could be used as a public key for encrypting a message, where t random errors would also be added to it. Since the code is chosen to be error-correcting with up to t errors, decoding can be achieved with the intial generator matrix – a private key to the cryptosystem.

The general idea of the McEliece scheme can be applied to multiple code structures. The original implementation used binary Goppa codes [22] and although uses of other codes have been proposed, most of them have had a structural attack proposed against them, making them insecure. The Goppa code implementation, however, remains not only secure but also achieves faster encryption and decryption than the popular RSA cryptosystem based on integer factorization. All the known attacks against the Goppa code implementation of the McEliece scheme have exponential complexity in the size of the code.

Unfortunately, the large public key sizes in the Goppa code implementation are what keep it from being implemented more in practice.

5.3 Binary Goppa Codes and Implementation for McEliece

Binary Goppa codes [22] — a subset of the more general class of codes described by V.D. Goppa — have interesting properties for cryptographic applications. Namely, the codes have a high error-correcting capacity with respect to code rate and the parity-check matrix is difficult to distinguish from a random, full-rank binary matrix.

As the name suggests, binary Goppa codes are defined over the binary field $\mathbb{F}_2$. Moreover, an extension field of degree m is used with an irreducible polynomial $f(\alpha)$ of degree m used as the modulus for $\mathbb{F}_2$. Note that the field $\mathbb{F}_{2^m}(\alpha)$ can also be expressed as $\mathbb{F}_2[\alpha]/f(\alpha)$ in this case, since any polynomial mod $f(\alpha)$ has at most degree m and all the combinations of m bits result in exactly 2^m coefficients to give 2^m elements. Due to $f(\alpha)$ being irreducible, each polynomial of degree less than m in $\mathbb{F}_{2^m}(\alpha)$ is guaranteed to have an inverse — a property necessary for the definition of the code later on.

Before presenting the definition a binary Goppa code, let us first briefly discuss two of its properties: the *code support* and the *error correction capacity*. Denote by

σ a sequence of n distinct elements from the binary extension field $\mathbb{F}_{2^m}(\alpha)$. We call this the *support* and it is a subset of $\mathbb{F}_{2^m}$. We can express polynomials in $\mathbb{F}_{2^m}(\alpha)$ with binary coefficients and degree at most $m-1$ as [34]:

$$p(\alpha) = \sum_{j=0}^{m-1} p_j \alpha^j. \tag{10}$$

Using similar notation, each element in the code support may be expressed as:

$$\sigma_i(\alpha) = \sum_{j=0}^{m-1} \sigma_{i,j} \alpha^j, \tag{11}$$

where $\sigma_{i,j}$ is the jth coefficient of support element $\sigma_i(\alpha)$.

For an error correction capacity of t errors, a polynomial $g(x)$ of degree t is used, such that

$$g(\sigma_i(\alpha)) \neq 0, \sigma_i(\alpha) \in \sigma. \tag{12}$$

One simple way to meet the above condition is to select an irreducible $g(x)$, since irreducible polynomials are square-free and this ensures the substitution of any $\sigma_i(\alpha)$ polynomial with each x from $g(x)$ does not cause all coefficients to cancel out.

The code itself is then defined as follows:

Definition 14. [34] Let σ denote a support with elements in $\mathbb{F}_{2^m}$ and let $f(x)$ denote an irreducible polynomial of degree t. A *binary Goppa code* is defined as:

$$\mathcal{C} = \boldsymbol{c} \in \{0,1\}^n | \sum_{i=0}^{n-1} \frac{c_i}{x - \sigma_i(\alpha)} \equiv 0 \bmod g(x). \tag{13}$$

The definition contains the inverse of an element mod $g(x)$, while the element is itself a function of both x and α. For the inverse of an element $e(x)$ mod $g(x)$ in a finite field, the extended Euclidean algorithm is generally applied and yields the following equation:

$$e(x) \cdot I_e + g(x) \cdot I_g = e(x) \cdot I_e \bmod g(x) = 1, \tag{14}$$

where I_e and I_g denote the inverse elements of $e(x)$ mod $g(x)$ and $g(x)$, respectively. Taking into account that $g(x)$ is a polynomial in both x and α yields the equation

$$(x - \sigma_i(\alpha)) \cdot I_e(x,\alpha) + g(x,\alpha) \cdot I_g(x,\alpha) = (x - \sigma_i(\alpha)) \cdot I_e(x,\alpha) \bmod g(x,\alpha) = 1. \tag{15}$$

Note that x is the main term in the polynomial $g(x)$, which gets its coefficients as a function of α. So, for example, letting $\sigma_0 = \alpha + \alpha^2$ gives $x - \sigma_0 = x + \alpha + \alpha^2 = x^0 \cdot (\alpha + \alpha^2) + x^1 \cdot 1$.

Contrary to some other code types, the generator matrix for a Goppa code is usually not constructed directly, but is instead derived through the parity check

matrix[22]. An explicit construction for the parity check matrix is as follows [11][34]:

$$\mathbf{H}_\Gamma = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ g_{t-1} & 1 & \cdots & 0 \\ g_{t-2} & g_{t-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \sigma_0(\alpha) & \sigma_1(\alpha) & \cdots & \sigma_{n-1}(\alpha) \\ \sigma_0^2(\alpha) & \sigma_1^2(\alpha) & \cdots & \sigma_{n-1}^2(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_0^{t-1}(\alpha) & \sigma_1^{t-1}(\alpha) & \cdots & \sigma_{n-1}^{t-1}(\alpha) \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{g(\sigma_0(\alpha))} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(\sigma_1(\alpha))} & \vdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & \frac{1}{g(\sigma_{n-1}(\alpha))} \end{bmatrix}. \tag{16}$$

Not all parity check matrices constructed as in (16) are convertible to generator matrices. It is estimated that around 33% of parity check matrices constructed from random code supports have linearly independent columns in the right-most $(n-k) \times (n-k)$ block, allowing for the derivation of a generator matrix as described in equations (2) and (3) [34].

Encryption and Decryption in McEliece

With a generator matrix $\mathbf{G}_\Gamma$ for a Binary Goppa code constructed, the McEliece cryptosystem instructs to first scramble and permute the generator matrix. For this two matrices are needed:

- $\mathbf{S}$, a random, $k \times k$, non-singular *scrambling* matrix
- $\mathbf{P}$, an $n \times n$, *permutation* matrix with exactly one "1" in each row and column

These matrices are used to conceal the structure of the generator matrix by multiplying them with $\mathbf{G}_\Gamma$ as:

$$\hat{\mathbf{G}}_\Gamma = \mathbf{S}\mathbf{G}_\Gamma\mathbf{P}. \tag{17}$$

To send message $\mathbf{M}$, we multiply it by $\hat{\mathbf{G}}_\Gamma$ and add some error vector $\boldsymbol{e}$, such that

$$\mathbf{T} = \mathbf{M}\hat{\mathbf{G}}_\Gamma + \boldsymbol{e} \tag{18}$$

is transmitted. This transmission is then received as $\mathbf{R}$ and the decryption process starts with

$$\mathbf{R}\mathbf{P}^{-1} = \mathbf{M}\mathbf{S}\mathbf{G}_\Gamma + \boldsymbol{e}\mathbf{P}^{-1}, \tag{19}$$

where the term $\boldsymbol{e}\mathbf{P}^{-1}$ contains the same number of errors (and has the same Hamming weight) as $\boldsymbol{e}$. These errors are removed via a decoding algorithm, such as the one presented by Patterson [44]. Effectively, the decoding algorithm extracts $\mathbf{M}$ from $\mathbf{M}\mathbf{G}_\Gamma + \boldsymbol{e}$. Finally, it remains to multiply from the right by $\mathbf{S}^{-1}$ to fully recover $\mathbf{M}$.

Public and Private Keys

While each implementation of the McEliece cryptosystem retains as private keys the matrices $\mathbf{S}$ and $\mathbf{P}$, as well as a suitable decoding algorithm $\mathcal{D}$ for the code, the Goppa code variant may add the support σ and the modulus $g(x, \alpha)$ to the list of private keys. These keys are listed in Table (2).

Table 2: The public and private keys for McEliece (Goppa)

Public	**Private**
Vector length n	Permutation matrix $\mathbf{P}$ $(n \times n)$
Field exponent m	Scrambling matrix $\mathbf{S}$ $(k \times k)$
Plain text length k	Code support σ
Error correction capacity t	mod $g(x, \alpha)$
The $(k \times (n-k))$ redundancy part of $\hat{\mathbf{G}}_\Gamma$	Efficient decoding algorithm $\mathcal{D}$

Security

The McEliece cryptosystem with a binary Goppa code implementation is projected to have a reasonable level of post-quantum security [8]. While Shor's algorithm is able to take advantage of certain periodic structures in other cryptography, Goppa codes in McEliece do not admit such structure. However, information set decoding based brute force attacks have been around since 1962 and have been improved on over the years [45]. The quantum-based Grover's algorithm could further speed up these brute force attacks [23]. The authors of the Classic McEliece submission for the National Institute of Standards and Technology (NIST) project for post-quantum cryptography also point out how careless use of the cryptosystem could open up attacks such as adaptive chosen ciphertext attacks [8]. Nevertheless, security remains the prevalent competitive advantage of this approach in post-quantum cryptography, while the most glaring disadvantage is the large public key size.

5.4 The GPT Cryptosystem

The Gabidulin-Paramonov-Tretjakov (GPT) cryptosystem was originally proposed in 1991 [13] and has since then been refined [15][40]. It is often described as an equivalent to the McEliece cryptosystem, implemented or codes over the rank metric instead of the Hamming metric. As described in the chapter on rank error-correcting codes, comparatively smaller public key sizes remain the driving motivation for pursuing a rank metric based system over McEliece.

Using Gabidulin codes, the system is parametrized by the following [42]:

- Extension field parameters q and m
- Code size parameters n and k

- Error-correction parameter $t < \frac{n-k-1}{2}$
- Additional security parameter $s \leq \min\{k, t\}$ introduced by Overbeck [42]

Key Generation, Encryption and Decryption in the GPT Cryptosystem

In addition to selecting a random generator matrix for our code, we randomly generate some matrices used to obscure the original generator matrix. The following matrices are generated randomly over $\mathbb{F}_{q^m}$:

- $\mathbf{G}_k$, a $k \times n$ generator matrix of a Gabidulin code with parameters (n, k, d)
- $\mathbf{X}$, an $n \times k$ random *distortion* matrix, having rank t over $\mathbb{F}_q$ and rank s over $\mathbb{F}_{q^m}$
- $\mathbf{S}$, a $k \times k$ non-singlar *row scrambler* matrix

An efficient decoding algorithm $\mathcal{D}$ is identified for the code and the structure of $\mathbf{G}_k$ is concealed by computing $\hat{\mathbf{G}}_k = \mathbf{S}(\mathbf{G} + \mathbf{X})$. Finally, we compute the error rank weight $\epsilon = \frac{n-k}{2} - t$.

A plaintext message vector $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$ is converted into ciphertext $\boldsymbol{c}$ by:

$$\boldsymbol{c} = \boldsymbol{m}\hat{\mathbf{G}}_k + \boldsymbol{e}, \tag{20}$$

where $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ is a random vector of rank weight ϵ.

For the decoding part, an application of the decoding algorithm yields

$$\mathcal{D}(\boldsymbol{c}) = \boldsymbol{m}\mathbf{S}\mathbf{G} \tag{21}$$

and the message can be retrieved. Since the ciphertext was only at a distance of $\frac{n-k}{2}$ from the code, and since the rank weight $rk(\boldsymbol{m}\mathbf{S}\mathbf{X}) \leq t$, the decoding should correct all of the induced errors.

Public and Private Keys

The public and private keys for the system are listed in Table (3).

Table 3: The public and private keys for GPT (Gabidulin)

Public	**Private**
Public generator matrix $\hat{\mathbf{G}}_k$ $(k \times n)$	Efficient decoding algorithm $\mathcal{D}$
Error rank parameter ϵ	Distortion matrix $\mathbf{X}$ $(n \times k)$
	Row scrambling matrix $\mathbf{S}$ $(k \times k)$

The volume of the public key is $k \cdot n \cdot m \cdot \log_2(q)$ bits and the information rate is

$\frac{k}{n}$ [40]. By comparison these public key sizes, as originally proposed would be up to 50 times smaller than respective public key sizes in the McEliece scheme.

Security

Confidence in Gabidulin code-based implementations has been shaken by effective structural attacks, first by Gibson [20] and later by Overbeck [43]. Yet, it remains an open question whether or not a suitable code structure, or some variant of the cryptoystem itself could provide a reasonable level of security.

The key generation notably lacks an equivalent to the *column permutation matrix* **P** in the McEliece scheme. There has been much optimism for such improvements or similar to counteract the threat of structural attacks [17][40]. The rank metric remarkably allows for the use of a *column scrambler* matrix in place of *column permutation* matrix, since the necessary operations do not cause unwanted rank errors as they would for the Hamming metric [40]. Unfortunately, such variants among others have recently shown to be insecure [42][26].

Fairly recent improvements to non-structural attacks also pose somewhat of a concern, as they restricts certain parameter sets as insecure without requiring any structural weaknesses [19].

It remains to be seen how long other code structures designed to resist generalizations for attacks by Gibson and Overbeck are able to do so. In the next chapter, we look to *twisted codes* for a potential solution.

6 Attacks and Countermeasures

The literature on code-based cryptography has seen a consistent cycle of generating ingenious solutions to counteract previous structural weaknesses, only to be shown vulnerable to yet another attack [20][21][40][42][46][30].

On the attacking side, new distinguishers to exploit structural properties are constantly found and existing key recovery algorithms are further combined and generalized to break down elusive variants of the cryptosystems [20][41][26][30].

Meanwhile the countermeasure side have attempted everything from new code families to modified column scramblers and twisted codes [13][40][46].

On some occasions, a vulnerability is averted by simply tweaking the parameter set slightly. At other times, a powerful new structural attack emerges, proving multiple implementations fundamentally insecure all at once.

6.1 Overbeck's Attack

In the field of rank metric based cryptography, Overbeck's attack has stirred waves of uncertainty and many dismantled rank based cryptography schemes have since been re-examined [41][42][43].

The impact of the specific attack merits a more detailed view into how the weaknesses in the GPT cryptosystem surface in the first place and how an attacker might recover the private keys.

Source of structural weakness

In the process of computing the public generator matrix with

$$\hat{\mathbf{G}}_k = \mathbf{S}(\mathbf{G} + \mathbf{X}),$$

the application of the distortion matrix $\mathbf{X}$ reduces the error-correction capability of the code. In other words, the rank weight of the artificial error vector e has to be reduced below what would otherwise still be correctable with the code [38].

Outline of the Attack

The attack applies Λ_i on the public generator matrix, an operator which applies the Frobenius operation $i \in \mathbb{N}$ times. The expected outcome of such an operation for a random linear code would be that the dimension of the code is increased by k [38]. Instead, Overbeck showed that the dimension of $\hat{\mathbf{G}}_k$ increases by only 1 for each Frobenius operation. The rank $\hat{\mathbf{G}}_k$, as well as the underlying code, is then distinguished by selecting $i = n - k - 1$, in which case the co-dimension (difference between dimensions in this case) to the public generator matrix becomes 1. In other words, the dimension of $\hat{\mathbf{G}}_k$ is n, while the dimension of $\Lambda_i(\hat{\mathbf{G}}_k)$ is $n - 1$.

The Frobenius operator, or *q-sum* Λ_i is defined in [43] as follows:

Definition 15. Let $\mathbf{M}$ be a random $l \times n$ matrix over $\mathbb{F}_q$ and let $i \in \mathbb{N}$. Then the mapping $\lambda_i : \mathbb{F}_{q^m}^{l \times n} \mapsto \mathbb{F}_{q^m}^{(i+1)l \times n}$ is defined as:

$$\Lambda_i(\mathbf{M}) := \begin{bmatrix} \mathbf{M} \\ \mathbf{M}^{[1]} \\ \mathbf{M}^{[2]} \\ \vdots \\ \mathbf{M}^{[i]} \end{bmatrix}, \tag{22}$$

where $\mathbf{M}^{[i]}$ denotes $\mathbf{M}^{q^i}$.

So, the distinguisher effectively splits the two cases as:

- $\dim \Lambda_i(\mathcal{C}) = \min\{n, ki\}$, on average for a random linear code $\mathcal{C}$
- $\dim \Lambda_i(\mathcal{C}_G) = \min\{n, k + i\}$, with high probability for a random Gabidulin code $\mathcal{C}_G$

The step-by-step key recovery process is presented in Algorithm (2) found in Appendix A.

The attack also relies on $\Lambda_{n-k-1}(\hat{\mathbf{G}}_k)$ yielding a co-dimension of 1 and on any column scrambler matrix $\mathbf{P}$ being selected from $\mathbb{F}_q$. While both of these criteria have been addressed since the publication of the attack (e.g. [48][16]), these further improvements are out of the scope for this inspection, or have since then been shown inadequate [38].

6.2 Twisted Gabidulin Codes

Gabidulin codes were considered the only construction for maximum rank distance codes for quite some time before [51] (and independently also [37]) had the idea of using a variant of linearized polynomials to define MRD codes that are not equivalent to Gabidulin codes. These codes were named *twisted Gabidulin codes*.

The linearized polynomials from [51] were later generalized to cover a broader set with all possible linear combinations also included [47]. Hence the authors of [47] refer to their own construction as twisted Gabidulin codes, implying that those presented in [51] and [37] are *special twists*, since they are contained in the more general construction.

There are two prominent open questions for twisted Gabidulin codes. The first one being whether or not there exists an efficient decoding algorithm and the second is to consider if an implementation in the GPT cryptosystem would be vulnerable to structural attacks. The former issue is outside the scope of this thesis. For the latter, on the other hand, the goal is to further continue the considerations in [46].

The authors of [46] describe twisted Gabidulin codes as:

Definition 16. ([[46], Def. 4]) Let $n, k, l \in \mathbb{N}$ with $k < n$ and $l \leq n - k$. Choose the following:

- a hook vector $\boldsymbol{h} \in \{0, \ldots, k-1\}^l$
- a twist vector $\boldsymbol{t} \in \{1, \ldots, n-k\}^l$ with distinct t_i
- $\boldsymbol{\eta} \in (\mathbb{F}_{q^m} \backslash \{0\})^l$.

The set of $[k, \boldsymbol{k}, \boldsymbol{h}, \boldsymbol{\eta}]$-*twisted linearized polynomials* over $\mathbb{F}_{q^m}$ is defined as

$$\mathcal{P}^{n,k}_{t,h,\eta} := \left\{ f = \sum_{i=0}^{k-1} f_i x^{[i]} + \sum_{j=1}^{l} \eta_j f_{h_j} x^{[k-1+t_j]} : f_i \in \mathbb{F}_{q^m} \right\}. \tag{23}$$

Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$ be linearly independent over $\mathbb{F}_q$ and write $\boldsymbol{\alpha} = [\alpha_1 \ldots, \alpha_n]$. The $[\boldsymbol{\alpha}, \boldsymbol{t}, \boldsymbol{h}, \boldsymbol{\eta}]$-twisted Gabidulin code of length n and dimension k is given by

$$\mathcal{C}_{\alpha,t,h,\eta}[n,k] = \left\{ [f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)] : f \in \mathcal{P}^{n,k}_{t,h,\eta} \right\}. \tag{24}$$

The definition results in the following generator matrix for a twisted Gabidulin code:

$$\mathbf{G}_{TG} = \begin{bmatrix} \alpha \\ \alpha^{[1]} \\ \vdots \\ \alpha^{[h_1-1]} \\ \alpha^{[h_1]} + \eta_1 \alpha^{[k-1+t_1]} \\ \alpha^{[h_1+1]} \\ \vdots \\ \alpha^{[h_l-1]} \\ \alpha^{[h_l]} + \eta_l \alpha^{[k-1+t_l]} \\ \alpha^{[h_l+1]} \\ \vdots \\ \alpha^{[k-1]} \end{bmatrix}, \tag{25}$$

where $h_1 < \ldots < h_l$.

Example 7. A twisted Gabidulin code of length 10 and dimension 5 could have hook vector $\boldsymbol{h} = [1, 3]$ and twist vector $\boldsymbol{t} = [2, 4]$. The generator matrix would have the following form:

$$\mathbf{G}_{TG} = \begin{bmatrix} \alpha \\ \alpha^{[1]} + \eta_1 \alpha^{[6]} \\ \alpha^{[2]} \\ \alpha^{[3]} + \eta_2 \alpha^{[8]} \\ \alpha^{[4]} \end{bmatrix}, \tag{26}$$

where $\eta_1, \eta_2 \in \mathbb{F}_{q^m}$ are some fixed parameters.

Intuitively, the hook vector determines which rows are twisted and the twist vector dictates the q-power from k to n of the added twist element. The parameter η is the coefficient of the twist element and it is chosen such that it either belongs to or does not belong to certain sub-fields of $\mathbb{F}_{q^m}$. The choice of η allows the dimension of the code to be k, while the elements of the linearized polynomial span a $\mathbb{F}_q$ subspace of dimension n.

This construction was shown in [47] to be MRD and in [46] it was shown to resist Overbeck's structural attack. This resistance comes from Overbeck's Λ_i operation (q-sum) not distinguishing it from a random code as easily when the parameters for the twists are chosen carefully.

The parameter set restriction proposed in [[46], Thm. 4][1] ensures that each distinct element t_j in the twist vector $t_j \in \{1, \ldots, n-k\}^l$ from the code $\mathcal{C}_{TG}$ generates a distinct linearly independent basis element in the q-sum $\Lambda_i(\mathbf{G}_{TG})$. Thus, large dimension for the q-sum can be consistently achieved with this construction, making twisted Gabidulin codes resilient to that type of structural attacks.

[1]The constraints on t_i in Thm. 4 contradict the domain for t set in Def. 4

6.3 Twisted Reed-Solomon Codes

Reed-Solomon codes being a close analogue to Gabidulin codes, it is no surprise that twisted Reed-Solomon codes have already been considered in the literature [4]. In fact, their construction was inspired by Sheekey's class of early twisted Gabidulin codes [51]. The combination of these two codes then was then used as a template for the description of the more generalized twisted Gabidulin codes in [47]. This, in turn, motivated the generalization of the single twist Reed-Solomon codes in [4] to the multi twist composition [5].

The analogy of the two code families is most evident with the definitions of their respective polynomials. Whereas a set of twisted linearized polynomials was used in the rank metric definition, the Hamming metric equivalent uses a type of *twisted polynomials*.

Definition 17. ([[5], Def. 1]) Let $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{\eta}$ be chosen as in (16), except $\boldsymbol{\eta} \in \mathbb{F}_q^n$. The set of $[k, \boldsymbol{t}, \boldsymbol{h}, \boldsymbol{\eta}]$-*twisted polynomials* is defined as

$$\mathcal{P}_{t,h,\eta}^{n,k} := \left\{ f = \sum_{i=0}^{k-1} f_i x^i + \sum_{j=1}^{l} \eta_j f_{h_j} x^{k-1+t_j} : f_i \in \mathbb{F}_q \right\}. \tag{27}$$

With the help of distinct basis elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$, we can construct the twisted Reed-Solomon code as

$$\mathcal{C}_{\alpha,t,h,\eta}[n,k] = \left\{ [f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)] : f \in \mathcal{P}_{t,h,\eta}^{n,k} \right\}, \tag{28}$$

where $k < n$ and $t \leq n - k$.

In general, twisted Reed-Solomon codes are not maximum distance separable, although certain sub-classes may be constructed as such. They were also shown to be largely distinct from generalized Reed-Solomon codes [49][4].

For cryptographers, this is yet another intriguing code family. A large subclass of MDS Reed-Solomon codes was recently theorized to be out of reach for certain structural attacks against the McEliece cryptosystem [5]. These attacks included Schur square distinguishing, the three Wieschebrink attacks and the attack by Sidelnikov and Shestakov [**?**][58][53].

Whilst the Sidelnikov-Shestakov attack breaks McEliece implementations with general Reed-Solomon codes in polynomial time, the distinguisher in that attack relies on the matrix $\mathbf{P}$ in the systematic form of the generator matrix $[\mathbf{I}_k|\mathbf{P}]$ being a Cauchy matrix specifically when the code is a generalized Reed-Solomon code [50]. This criterion is not met with twisted Reed-Solomon codes, which are rarely equivalent to the general Reed-Solomon codes.

The Schur square distinguishing process is also worth analyzing. For two codes, the Schur product is the linear span

$$\mathcal{C}_1 \star \mathcal{C}_2 := \langle c_1 \star c_2 \mid c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2 \rangle_{\mathbb{F}_q}.$$

Hence, the Schur square is the product $\mathcal{C} \star \mathcal{C}$, the dimension of which is the distinguishing feature. The two cases for the distinguisher are roughly

- $\dim \mathcal{C} \star \mathcal{C} = \min \left\{ n, \frac{1}{2}k(k+1) \right\}$ with high probability for a random linear code $\mathcal{C}$
- $\dim \mathcal{C}_{TR} \star \mathcal{C}_{TR} = \min \{n, 2k-1\}$, when $k < \frac{n}{2}$ for a Reed-Solomon code $\mathcal{C}_{TR}$, or any shortening at up to 2 positions of $\mathcal{C}_{TR}$ [**?**].

The twisted codes from [5] and their shortenings at up to 2 positions were shown to always have Schur square dimension n and dimension $k > \frac{n}{2}$, so this attack fails in its premise.

6.4 Attack on Twisted Reed-Solomon Codes

Julien Lavauzelle and Julian Renner observed in 2019 that although twisting Reed-Solomon codes protects them from previously known attacks, they still have enough structure to make other structural attacks deployed against them viable [30]. The authors presented an efficient approach, which uses the Schur square of a *subfield subcode* of the public code to enable the Sidelnikov-Shestakov attack.

They key vulnerability is that the MDS construction for twisted Reed-Solomon codes is done over a chain of subfields $\mathbb{F}_{q^{m_0}} \subsetneq \mathbb{F}_{q^{m_1}} \ldots \subsetneq \mathbb{F}_{q^{m_l}} = \mathbb{F}_{q_0}$, where m_i are non-negative integers m_0 being the smallest, the basis elements $\alpha_1, \ldots, \alpha_n$ are distinct for $i = 1, \ldots, n$, so we need $q \geq n$ and α_i are selected from the field $\mathbb{F}_{q^{m_0}}$ [5]. In this MDS construction, n and k are chosen with $2\sqrt{n} + 6 < k \leq \frac{n}{2} - 2$ [30]. The public code is intersected with the smallest subfield in the chain, $\mathbb{F}_q$, to obtain part of the structure of a Reed-Solomon code:

$$\mathcal{C}_{pub} \cap \mathbb{F}_q^n = \left\{ [f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)] : f \in \mathcal{P}_{t,h,\eta}^{n,k} \cap \mathbb{F}_q[x] \right\}. \tag{29}$$

In the attack shown in [59], Wieschebrink remarks that applying the Schur square to random subcodes of Reed-Solomon codes outputs a Reed-Solomon code with high probability. Thus, the Sidelnikov-Shestakov attack works on the Schur square code of the subfield subcode with high probability.
Letting $\hat{\mathbf{G}}_{TRS}$ denote the public generator matrix of a twisted Reed-Solomon code in the McEliece cryptosystem, the algorithm detailed in Appendix B shows the step-by-step process of retrieving an alternative valid secret key $(\mathbf{S}^*, \boldsymbol{\alpha}^*, \boldsymbol{\eta}^*)$ from $\hat{\mathbf{G}}_{TRS}$. In the algorithm pseudocode ([[30], Algorithm 1]), the function names SubfieldSubcode, Square and SidelShest refer to intersecting for the subfield subcode, taking the Schur square and applying the Sidelnikov-Shestakov attack, respectively.

The function GenSub maps

$$[\alpha_1, \ldots, \alpha_n] \in \mathbb{F}_q^n \mapsto \begin{bmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_n \\ \vdots & \ddots & \vdots \\ \alpha_1^{h_1-1} & \ldots & \alpha_n^{h_1-1} \\ \alpha_1^{h_1+1} & \ldots & \alpha_n^{h_1+1} \\ \vdots & & \vdots \\ \alpha_1^{h_l-1} & \ldots & \alpha_n^{h_l-1} \\ \alpha_1^{h_l+1} & \ldots & \alpha_n^{h_l+1} \\ \vdots & & \vdots \\ \alpha_1^{k-1} & \ldots & \alpha_n^{k-1} \end{bmatrix} \in \mathbb{F}_q^{(k-l)\times n}.$$

The interpolation in the algorithm maps $(\mathbf{a}, \mathbf{b}) \in \mathbb{F}_q^n \times \mathbb{F}_{q_0}^n$ to $\mathbf{g} \in \mathbb{F}_q^n$ with the condition

$$\sum_{j=1}^{n} g_j a_i^{j-1} = b_i$$

for $i = 1, \ldots, n$. The last function, GTRS, matches the obtained α^* and η^* with the generator matrix $\mathbf{G}^*$ of suitable Reed-Solomon code [30]. $\mathbf{S}^*$ can be computed such that $\mathbf{S}^*\mathbf{G}^* = \hat{\mathbf{G}}_{TRS}$. The algorithm runs in polynomial time with $\mathcal{O}(n^4)$ operations and asserts that the construction of [5] as insecure for at least most practical parameter sets.

6.5 Analysis of the Same Attack on Twisted Gabidulin Codes

The success of subfield subcodes with exposing the structure in twisted Reed-Solomon codes begs the question of how a similar attack would fare against a rank based cryptosystem implementing twisted Gabidulin codes. The authors of the attack in [30] suspect the approach could be used to attack the GPT system with twisted Gabidulin codes presented in [46] and also conducted some experimental simulations to test this hypothesis. Julien Lavauzelle and Julian Renner then generously shared their SageMath [56][2] implementation for an attack combining the subfield subcode part of the attack in Appendix B with Overbeck's attack, as described in e.g. Appendix A.

A variation of the GPT cryptosystem is used to implement twisted Gabidulin codes in [46]. This adaptation makes use of a column permutation matrix $\mathbf{P}$ and the public generatior matrix is computed as:

$$\hat{\mathbf{G}}_{TG} = \mathbf{S}[\mathbf{X}|\mathbf{G}_{TG}]\mathbf{P}, \tag{30}$$

where $\hat{\mathbf{G}}_{TG}$ is the $(k \times n)$ generator matrix of a twisted Gabidulin code of rank k, $\mathbf{S}$ is a random $(k \times k)$ row scrambling matrix of full rank over $\mathbb{F}_{q^m}$, $\mathbf{X}$ is a random $(k \times \lambda)$ distortion matrix with rank $1 \leq s \leq \lambda$ and $\mathbf{P}$ is a random $((n+\lambda) \times (n+\lambda))$

[2] System for Algebra and Geometry Experimentation, v.8.9

column permutation matrix of full rank over $\mathbb{F}_q$. For twisted Gabidulin codes, the restrictions of the twisted Reed-Solomon system for parameters $q \geq n$, n and k do not apply. However, recall that for twisted Gabidulin codes α_i are linearly independent in addition to being distinct.

From this system, the attack aims to recover an alternate private key $(\mathbf{S}^*, \boldsymbol{\alpha}^*, \boldsymbol{\eta}^*, \mathbf{P}^*)$ directly from the twisted public generator matrix $\hat{\mathbf{G}}_{TG}$. Overbeck's attack then gets applied to $\hat{\mathcal{C}}_{TG} \cap \mathbb{F}_q$. A successful application of Overbeck's attack yeilds $\boldsymbol{\alpha}^*$ and $\mathbf{P}^*$, with $\boldsymbol{\eta}^*$ then being obtained via interpolation as in Appendix B and $\mathbf{S}^*$ being computed such that

$$\mathbf{S}^*\mathbf{G}^* = (\hat{\mathbf{G}}_{TG}\mathbf{P}^{*-1})_{[\lambda+1:n]},$$

where $\mathbf{G}^*$ is the generator matrix of an $[\boldsymbol{\alpha}^*, \boldsymbol{t}, \boldsymbol{h}, \boldsymbol{\eta}^*]$-twisted Gabidulin code and only the final $n - \lambda$ coordinates are being considered from $\hat{\mathbf{G}}_{TG}\mathbf{P}^{*-1}$.

Figure 2 illustrates the structures of generator matrices that are relevant to the attack. It demonstrates how the generator matrix of a twisted Gabidulin code incorporates higher q-powers of $\boldsymbol{\alpha}$ without increasing the dimension of the code. It also highlights how a generator matrix for the subfield subcode is void of these higher q-powers. In the figure, $\boldsymbol{\beta}$ is some affine transformation of $\boldsymbol{\alpha}$.

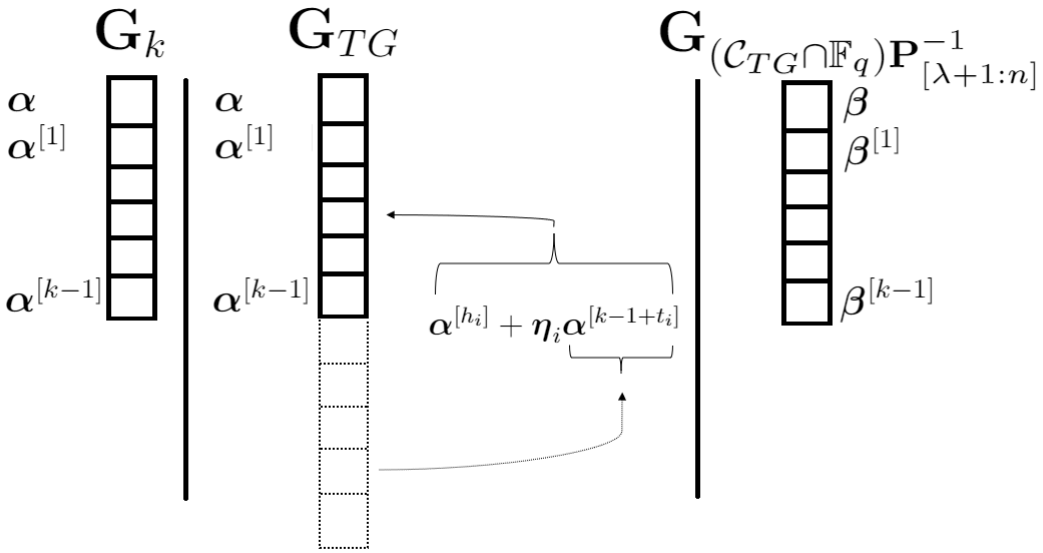

Figure 2: Structures of generator matrices

Experimental Approach

The proposed parameter sets from [[46], Table 1] were a natural starting point for experimentation. The attack failed on each of those sets of parameters in the SageMath implementation. To narrow down the problem, the next idea was to look for parameter sets for which that attack could directly resolve. Lavauzelle and Renner

had discovered that a choice of small λ and an s, such that $\mathbf{X}$ has full rank over $\mathbb{F}_q$ were promising research directions [30].

Before fully committing to the analysis of parameters λ and s, some testing was done to see if the twists, specifically the types of twists to achieve large q-sum dimension proposed in [[46], Thm. 4], would have some carry over effect into the subfield subcode. This was done under awareness of the fact that the subfield subcode structure had already been used to extract adequate structure from the code to enable Overbeck's attack for some parameters.

Various choices for the degrees of relative extension fields were also tried out. These relative extensions are constructed with q^{m_0} replacing the base field degree q in the code construction phase and q^{m_1} being inserted as the larger extension degree in place of q^m with $1 < m_0 < m_1 \in \mathbb{N}$. This is still done over the base field $\mathbb{F}_q$ and there is a chain of subfields, i.e. $\mathbb{F}_q \subsetneq \mathbb{F}_{q^{m_0}} \subsetneq \mathbb{F}_{q^{m_1}}$. The binary field with $q = 2$ was used as the base field for all experimentation and the code length was set to $n = m_0$.

In the search for patterns occurring with λ and s, different parameter choices were categorized solely on the basis of the attack succeeding. In the successful cases, the parameters were simply recorded, whereas in the unsuccessful cases the breaking point of the algorithm and variable values just prior to that point were taken note of. For parameter sets with large values for n, k, λ only one instance was simulated due to these simulations being time consuming, while parameter sets with comparatively small n, k, λ ($k \leq 34$) were simulated 10 times each.

The last step was to analyse the patterns in λ and s, especially w.r.t. n and k and compare the findings to those in the literature.

Experimental Results

As expected, the twists in the code did not prevent or slow down the attack at all. There was no noticeable change in the running times for the algorithm when number or structure of twists were varied. Not even when maximal q-sum dimension for the public code was pursued. Selecting $m_1 = 2m_0$ over $m_1 = 4m_0$ for the relative extension fields also had no observable impact.

The feasible choices for λ appeared to range from 1 to some largest value λ_L specific to the choice of n and k. Further, the choice of a valid s then depended on n, k and λ, where an s corresponding to a full $\mathbb{F}_q$ rank was always viable. The trend for suitable s then turned out in consecutive integers down from λ to some minimum value, which was some function of n, k and λ.

Table 4 shows some sets of parameters, revealed to be broken by experimental simulation.

Table 4: Experimentally broken parameter sets

k	n	λ_L	set of s for λ_L
18	26	3	$\{3\}$
21	33	5	$\{4, 5\}$
32	48	8	$\{7, 8\}$
34	52	9	$\{7, 8, 9\}$
46	80	14	$\{11, 12, 13, 14\}$
82	140	53	$\{i \in \mathbb{N} \mid 32 \leq i \leq 53\}$

The uniting feature for these parameter sets is that they cause the *right kernel* of the public generator matrix to have rank 1, where the right kernel is the set of vectors $\boldsymbol{v}$ for which $\hat{\mathbf{G}}_{TG}\boldsymbol{v} = 0$. Conversely, for almost every parameter set where the attack did not succeed, the rank of the right kernel was greater than one. The only exception to this within the tested parameter sets emerged with $\lambda = 3, s = 2$. With parameter sets having $\lambda > \max \lambda$, the subfield subcode structure does not expose the structure of the public code. This however does greatly narrow down the search space for potential attackers.

For the cases where $\lambda \leq \lambda_L$, corresponding parameter selections have been already discussed in the literature, as P. Loidreau suggested restricting the parameters of a distortion matrix $\mathbf{X}$ in Gabidulin codes for the GPT system [32]. The intent was to guarantee a sufficiently high dimension for the right kernel of the public generator matrix and it was proposed as an early countermeasure in 2010 to Overbeck's attack on Gabidulin codes.

Loidreau's design was addressed only fairly recently, as Horlemann-Trautmann et. al. showed in 2018 a way to extend Overbeck's attack further and to focus on retrieving elements of rank one directly from the code, disregarding the dual code entirely [27].

In any case, these considerations are no longer specific to twisted codes, and the simulation results support the claim that subfield subcodes may completely negate any practical cryptographic advantage from twisted codes, when $\lambda \leq \lambda_L$. In addition, the choice of secure parameters seems to be severely restricted in the $\lambda > \lambda_L$ case.

7 Conclusion

This thesis has covered a diverse array of topics in code-based cryptography, ultimately leading to the description and assessment of twisted codes. Twisted codes were the driving aspiration for much of the thesis, with the primary goal of having some meaningful discussion on them in light of the most recent publications. The topic can be quite intimidating at first, since the concepts build and expand upon themselves, as is the case in other areas of mathematics correspondingly. Hence, one of the main contributions of the thesis is the contextual narrative, which leads the reader all the way from learning the definition for a linear code, to potentially grasping the essence of structural vulnerabilities in code-based cryptography and how these challenges are approached. This write-up makes ample use of existing literature and serves also as a survey to the state-of-the-art research. The practical experimentation and simulation make for the second main contribution.

As for further research regarding twisted codes, one worthwhile prospect would be to look into the possibility of masking the subfield subcode structure. This would immediately reinstate twisted codes as a potentially viable solution for secure communications in the future. Another research direction would be to further analyse the numerous subfamilies of twisted codes, since they are a broad, recently discovered and largely unexplored family of codes overall. The fact that the specific constructions from [46] and [5] are shown to be mostly insecure does not imply the same for other subfamilies.

Rank metric codes have yet again been reconfirmed as highly susceptible to structural attacks, even though the episode with twisted codes has returned to more or less same situation as where it started — with the construction and near complete dismantling of another supplementary feature to conceal rank code structure. Nevertheless, continued interest in rank codes is bound to live on, with pressure on one side from the ever more imminent threat posed by quantum computers and on the other hand, the unfeasible public key sized of the secure post-quantum alternatives. Besides, the field of code-based cryptography is a constant back and forth exchange of ideas and in time it will be due for a novel countermeasure.

References

[1] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[2] Daniel Augot, Lejla Batina, Daniel J Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, et al. Initial recommendations of long-term secure post-quantum systems. *Available at pqcrypto. eu. org/docs/initial-recommendations. pdf*, 2015.

[3] Stephane Beauregard. Circuit for shor's algorithm using 2n+ 3 qubits. *arXiv preprint quant-ph/0205095*, 2002.

[4] P. Beelen, S. Puchinger, and J. Rosenkilde né Nielsen. Twisted reed-solomon codes. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 336–340, June 2017.

[5] Peter Beelen, Martin Bossert, Sven Puchinger, and Johan Rosenkilde. Structural properties of twisted reed-solomon codes with applications to cryptography. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 946–950. IEEE, 2018.

[6] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[7] Daniel J Bernstein. Introduction to post-quantum cryptography. In *Post-quantum cryptography*, pages 1–14. Springer, 2009.

[8] Daniel J Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, et al. Classic mceliece: conservative code-based cryptography. *NIST submissions*, 2019.

[9] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography-dealing with the fallout of physics success. *IACR Cryptology ePrint Archive*, 2017:314, 2017.

[10] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[11] R. Overbeck D. Engelbert and A. Schmidt. A summary of mceliece-type cryptosystems and their security. Cryptology ePrint Archive, Report 2006/162, 2006. https://eprint.iacr.org/2006/162.

[12] Ph. Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3):226–241, 1978.

[13] 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*, pages 482–489. Springer Berlin Heidelberg, 1991.

[14] Ernst M. Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985.

[15] Ernst M. Gabidulin. *Public-key cryptosystems based on linear codes.* Citeseer, 1995.

[16] Ernst M Gabidulin. Attacks and counter-attacks on the gpt public key cryptosystem. *Designs, Codes and Cryptography*, 48(2):171–177, 2008.

[17] Ernst M Gabidulin, Alexei V Ourivski, P Farrell, M Darnell, and B Honary. Improved gpt public key cryptosystems. *Coding, Communications, and Broadcasting*, pages 73–102, 2000.

[18] Ernst M. Gabidulin and Nina I. Pilipchuk. Error and erasure correcting algorithms for rank codes. *Designs, codes and Cryptography*, 49(1-3):105–122, 2008.

[19] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Transactions on Information Theory*, 62(2):1006–1019, 2015.

[20] J. Keith Gibson. Severely denting the gabidulin version of the mceliece public key cryptosystem. *Des. Codes Cryptography*, 6(1):37–45, July 1995.

[21] J. Keith Gibson. The security of the gabidulin public key cryptosystem. In *Eurocrypt*, 1996.

[22] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[23] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[24] Richard W. Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[25] Israel N. Herstein. *Topics in algebra.* John Wiley & Sons, 2006.

[26] Anna-Lena Horlemann-Trautmann, Kyle Marshall, and Joachim Rosenthal. Considerations for rank-based cryptosystems. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2544–2548. Ieee, 2016.

[27] Anna-Lena Horlemann-Trautmann, Kyle Marshall, and Joachim Rosenthal. Extension of overbeck's attack for gabidulin-based cryptosystems. *Designs, Codes and Cryptography*, 86(2):319–340, 2018.

[28] Loo-Keng HUA. A theorem on matrices over a sfield and its applications. In *Bulletin of the American Mathematical Society*, volume 55, pages 1046–1046. Amer Mathematical Soc 201 Charles St, Providence, RI 02940-2213, 1949.

[29] Gareth A. Jones and J. Mary Jones. *Information and coding theory*. Springer Science & Business Media, 2012.

[30] Julien Lavauzelle and Julian Renner. Cryptanalysis of a system based on twisted reed-solomon codes. *arXiv preprint arXiv:1904.11785*, 2019.

[31] Pierre Loidreau. *Etude et optimisation de cryptosystèmes à clé publique fondés sur la théorie des codes correcteurs*. PhD thesis, 2001.

[32] Pierre Loidreau. Designing a rank metric based mceliece cryptosystem. In *International Workshop on Post-Quantum Cryptography*, pages 142–152. Springer, 2010.

[33] Robert J. McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[34] Marcus Michiel. White paper on mceliece with binary goppa codes. Technical report, Department of Mathematics and Computer Science, Eindhoven University of Technology, February 2019.

[35] Stefan M. Moser and Po-Ning Chen. *A student's guide to coding and information theory*. Cambridge University Press, 2012.

[36] Oystein Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35(3):559–584, 1933.

[37] Kamil Otal and Ferruh Özbudak. Explicit constructions of some non-gabidulin linear maximum rank distance codes. *Advances in Mathematics of Communications*, 10(3), 2016.

[38] Ayoub Otmani, Hervé Talé Kalachi, and Sélestin Ndjeya. Improved cryptanalysis of rank metric schemes based on gabidulin codes. *Designs, Codes and Cryptography*, 86(9):1983–1996, 2018.

[39] Alexei V. Ourivski. Recovering a parent code for subcodes of maximal rank distance codes. In *Proc. of WCC*, volume 3, pages 357–363, 2003.

[40] Alexei V. Ourivski and Ernst M. Gabidulin. Column scrambler for the gpt cryptosystem. *Discrete Applied Mathematics*, 128(1):207–221, 2003.

[41] Raphael Overbeck. A new structural attack for gpt and variants. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology – Mycrypt 2005*, pages 50–63, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[42] Raphael Overbeck. Extending gibson's attacks on the gpt cryptosystem. In *Coding and Cryptography*, pages 178–188. Springer Berlin Heidelberg, 2006.

[43] Raphael Overbeck. Structural attacks for public key cryptosystems based on gabidulin codes. *Journal of cryptology*, 21(2):280–301, 2008.

[44] Nicholas Patterson. The algebraic decoding of goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.

[45] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[46] Sven Puchinger, Julian Renner, and Antonia Wachter-Zeh. Twisted gabidulin codes in the gpt cryptosystem. *arXiv preprint arXiv:1806.10055*, 2018.

[47] Sven Puchinger, John Sheekey, et al. Further generalisations of twisted gabidulin codes. *arXiv preprint arXiv:1703.08093*, 2017.

[48] Haitham Rashwan, Ernst M Gabidulin, and Bahram Honary. A smart approach for gpt cryptosystem based on rank codes. In *2010 IEEE International Symposium on Information Theory*, pages 2463–2467. IEEE, 2010.

[49] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[50] Ron M. Roth and Abraham Lempel. On mds codes via cauchy matrices. *IEEE transactions on information theory*, 35(6):1314–1319, 1989.

[51] John Sheekey. A new family of linear maximum rank distance codes. *arXiv preprint arXiv:1504.01581*, 2015.

[52] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[53] Vladimir M. Sidelnikov and Sergey O. Shestakov. On insecurity of cryptosystems based on generalized reed-solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[54] Natalia Silberstein, Ankit Singh Rawat, and Sriram Vishwanath. Error resilience in distributed storage via rank-metric codes. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1150–1157. IEEE, 2012.

[55] Danilo Silva, Frank R Kschischang, and Ralf Koetter. A rank-metric approach to error control in random network coding. *IEEE transactions on information theory*, 54(9):3951–3967, 2008.

[56] W. A. Stein et al. *Sage Mathematics Software (Version 8.9)*. The Sage Development Team, 2019. `http://www.sagemath.org`.

[57] Antonia Wachter-Zeh. Lecture notes in security in communications and storage, 2017/2018.

[58] Christian Wieschebrink. An attack on a modified niederreiter encryption scheme. In *International Workshop on Public Key Cryptography*, pages 14–26. Springer, 2006.

[59] Christian Wieschebrink. Cryptanalysis of the niederreiter public key scheme based on grs subcodes. In *International Workshop on Post-Quantum Cryptography*, pages 61–72. Springer, 2010.

A Appendix A

Data: Public generator matrix $\hat{\mathbf{G}}_k$
Result: Private keys $\mathbf{S}^*$ and $\mathbf{P}^*$ (row and column scramblers)
initialization;
while $\Lambda_i(\hat{\mathbf{G}}_k)$ *has full rank* **do**
  $i \leftarrow i + 1$;
  compute $\Lambda_i(\hat{\mathbf{G}}_k)$;
end
if $\Lambda_i(\hat{\mathbf{G}}_k)^\perp$ *has not full rank over* $\mathbb{F}_q$ **then**
  compute column scrambler $\mathbf{P}^*$, such that first rows of $\Lambda_i(\hat{\mathbf{G}}_k)^\perp \mathbf{P}^{*\top}$ are zero;
else
  attack fails;
end
if *The last columns of* $\hat{\mathbf{G}}_k \mathbf{P}^{*-1}$ *generate a Gabidulin code* **then**
  use some existing method (e.g. Ourivski [39]) to compute its generator vector g^* and a row scrambler $\mathbf{S}^*$;
end
Verify that $\mathbf{S}^*$ and $\mathbf{P}^*$ are part of a valid secret key

Algorithm 1: Overbeck's Attack [42]

B Appendix B

Data: Public generator matrix $\hat{\mathbf{G}}_{TRS}$
Result: Private key $(\mathbf{S}^*, \boldsymbol{\alpha}^*, \boldsymbol{\eta}^*)$
initialization;
$\mathbf{G}_{sub} \leftarrow \text{SubfieldSubcode}(\hat{\mathbf{G}}_{TRS}) \in \mathbb{F}_q^{(k-l)\times n}$;
$\mathbf{G}_{sq} \leftarrow \text{Square}(\mathbf{G}_{Sub}) \in \mathbb{F}_q^{(2k-1)\times n}$;
$\boldsymbol{\alpha'} \leftarrow \text{SidelShest}(\mathbf{G}_{sq}) \in \mathbb{F}_q^n$;
$i \leftarrow 1 \in \mathbb{N}$;
while $\mathbf{G}'(\mathbf{G}_{sub}^{\perp})^{\top} \neq 0$ **do**
  $b \leftarrow\in \mathbb{F}_q$;
  $\boldsymbol{\alpha}^* \leftarrow (\alpha'_1 - b, \ldots, \alpha'_n - b) \in \mathbb{F}_q$;
  $\mathbf{G}' \leftarrow \text{GenSub}(\boldsymbol{\alpha}^*) \in \mathbb{F}_q^{(k-l)\times n}$;
  $i \leftarrow i + 1$;
end
for $j \leftarrow 1$ *to* l **do**
  $i \leftarrow 1 \in \mathbb{N}$;
  while $g_{h_j+1} = 0$ **do**
    $\mathbf{g} \leftarrow \text{Interpolate}(\boldsymbol{\alpha'}, (\mathbf{G}_{pub_{i,1}}, \ldots, \mathbf{G}_{pub_{i,n}})) \in \mathbb{F}_{q_0}^n$;
    $i \leftarrow i + 1$;
  end
  $\eta_j^* \leftarrow \frac{g_{k-1+t_j}}{g_{h_j+1}}$
end
$\mathbf{G}_{TRS}^* \leftarrow \text{GTRS}(\boldsymbol{\alpha}^*, \boldsymbol{\eta}^*) \in \mathbb{F}_{q_0}^{k\times n}$;
$\mathbf{S}^* \leftarrow \mathbf{G}_{TRS}^* \backslash \hat{\mathbf{G}}_{TRS} \in \mathbb{F}_{q_0}^{k\times k}$;
return $(\mathbf{S}^*, \boldsymbol{\alpha}^*, \boldsymbol{\eta}^*)$

Algorithm 2: Attack by Lavauzelle and Renner [[30], Algorithm 1]