On Code-based Cryptosystems using binary codes with large minimum distance

Post-Quantum Cryptography

R. Yorgova

Delft University of Technology

TU Delft

On Code-based Cryptosystems using binary codes with large minimum distance

Post-Quantum Cryptography

by

R. Yorgova

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on August 24, 2021 at 10AM

Student number: 4952545
Project duration: May, 2020 – August, 2021
Thesis committee:

Dr. Stjepan Picek,	TU Delft, Supervisor
Dr. Zekeriya Erkin,	TU Delft
Dr.ir. Jos Weber,	TU Delft

TU Delft

Abstract

Code-Based Cryptography is a branch of the Post-Quantum Cryptography research area. As such, its focus is on developing algorithms that can be used in the current communication systems to secure them against an adversary powered in the (near) future by a quantum computer. A code-based type cryptosystem is a public key cryptosystem that is resistant or slightly reduces its security level against attacks by the known quantum algorithms. The biggest drawback of this otherwise secure cryptosystem is its large public key.

This thesis considers a specific type of linear codes, large minimum distance self-dual codes, and punctured codes derived from them that can provide the same security level as the original McEliece system with approximately a 30% smaller public key.

Estimation of the bit security level of a cryptosystem using a small example of such a code for its private key confirms that increasing the minimum weight of the code significantly reduces the public key size of the system.

Further, we determine the parameters of putative self-dual codes with a large minimum weight providing classical bit security of 80, 128 and 256 bits (quantum 67, 101, and 183 bits), respectively. For the parameters corresponding to the classical 80 bits security of the McEliece Cryptosystem, a particular example of a binary high minimum distance self-dual code is constructed. It is the first code of its type and length. A punctured code of this example is used for the private key of the McEliece cryptosystem. A new decoding algorithm is introduced, which is suitable for the specific construction of the new self-dual code. Moreover, we present a decryption strategy that decodes the complete code instead of the punctured private key. All this results in a McEliece type cryptosystem with 80 bits security classical (68 bits quantum) and reduced public key size of around 38.5% compared to the original system. Reducing the key size makes the quantum safe McEliece Cryptosystem more attractive for practical use.

Preface

1. On the content

This report describes my study and research on Code-based cryptosystems. It includes the definition and analysis of a McEliece type encryption scheme using a family of large minimum distance codes. This family of codes has not been previously considered for use in such a cryptosystem. To define the Code-based scheme, we deliver the following results:

1. An implementation and cryptanalysis of a McEliece type cryptosystem using a small example of a binary code with a large minimum distance.
2. Estimated parameters of a large minimum weight code which would provide the currently required security levels of 80, 128 and 256 bits for the cryptosystem.
3. An example of a code suitable for the security level of 80 bits.
4. A decoding algorithm developed for the specific structure of the code from 2.
5. A proof that the new decoding is a valid scheme for a large group of codes.
6. An encryption scheme that integrates the new decoding algorithm.

The results are divided into parts 1, 2, 3 and 6 in one group, and 4 and 5 in another. The first group of results are included in a conference submission, written together with Luca Mariot and Stjepan Picek. The second group results are submitted to a journal. Both submissions are still waiting to be reviewed. They can be found on:

- https://eprint.iacr.org/2021/837, L. Mariot, S. Picek and R. Yorgova, 'On McEliece type cryptosystems using self-dual codes with large minimum weight', submitted 19 Jun 2021.
- https://arxiv.org/abs/2106.11146, R. Yorgova, On decoding a specific type of self-dual codes, submitted 16 Jun 2021.

2. On the implementation

We have implemented proof of concepts for most of the algorithms developed in this work. It consists of C^{++} and $SageMath$ code. The code is not provided.

The data generated and used for the experiments can be found in a git repository with a link https://github.com/yorgova/MsThesisData.

3. Acknowledgements

During my thesis project, I was supervised by Stjepan Picek and Luca Mariot. I would like first to thank Stjepan for accepting me as a 13^{th} graduate student and for accepting my project idea that was quite away from the research of the group. Next, I would like to thank him for the

support, useful discussions and guidance this year. I like to express my gratitude to Luca for the discussions, for questioning some of my ideas, and for the thorough check of our submission and parts of this work. I also want to thank the students and researchers of AISyLAB for the technical support regarding the HPC/cluster and the cheering up online group meetings.

Further, I like to thank Zekeriya Erkin and Jos Weber for being members of my thesis committee. I thank Jos Weber also for introducing me the code-based cryptosystems, which provoked my interest.

Finally, I would like to say that I am lucky to have my partner, my son and all my family behind my back all the time. I want to thank all of them for their patience and for accepting my absence due to my busy time during the project. Special thanks to my partner, who supported me constantly in this studying adventure.

R. Yorgova
Delft, August 2021

Contents

1

Introduction

The development of quantum computers has a very high priority within the scope of international research and development. While the high speed of quantum computations brings great opportunities, for example, for medical research, it increases the risk for the security of the cryptography used in the current communication systems and stored data.

Interception of a communication channel and getting a copy of the information traffic is a feasible and common practice used, e.g., by Internet Service Providers (ISP) and State Agencies. Therefore, in order to protect the data, encryption is often applied in different layers of the communication network. Since a large amount of the intercepted/stolen data is stored for the time when a powerful quantum computer will be available, it is of high importance that the currently implemented cryptographic algorithms are safe against both classical and quantum computing.

In the current cryptosystems, communication protocols (e.g., TLS), and digital signatures, there are two public key algorithms[1] which are widely implemented. These crypto algorithms are based on mathematical problems that are hard towards classical computing, but they are vulnerable towards the quantum Shor's algorithm. This quantum computing algorithm can solve both problems in polynomial time, which implies that all the currently intercepted/stolen encrypted[2] data are at risk to be revealed in the coming years.

There is still time to update digital signatures until a powerful quantum computer is developed and available. It is because digital signatures are valid only for the time of use. In the future, when the signature algorithm/public key is changed, then the old digital signatures would not be accepted. On the contrary, for the other applications using encryption, the switch to quantum safe algorithms is urgent. Post-Quantum Cryptography responds to this need.

1. Post-Quantum Cryptography and Code-Based Cryptosystems

Post-Quantum Cryptography (also called quantum-resistant cryptography) is the area of research focused on the development of algorithms that can be used in the current communication systems and which can secure them against an adversary using both classical and quantum computers.

Post-Quantum Cryptography has five branches named after the type of mathematical problems that are resistant to the existing quantum algorithms and are believed to be hard to solve

[1] Rivest-Shamir-Adleman (RSA) and the Elliptic Curve Diffie-Hellman

[2] Encrypted by one of the non quantum safe public key algorithms

with a large-scale quantum computer. These branches are Hash-based, Code-based, Lattice-based, Multivariate, and Supersingular isogeny-based.

Our focus is on Code-Based Cryptography. As the name suggests, it studies cryptosystems based on error-correcting codes. The first Code-Based cryptosystem is the public key cryptosystem, proposed by Robert McEliece back in 1978 [40], which after longer than 40 years of study remains with almost no change in its security level.

The code-based cryptosystems are resistant to the quantum Shor's algorithm. The other well known quantum algorithm, Grover's search algorithm, has an impact on them by reducing the bit security of the system by at most half. This reduction can be suppressed by doubling the target security level.

Still, there is a major drawback, namely the large size of their public keys. This is a practical limitation for broad use in the current communication systems. For comparison, for the 128 bits security level of the McEliece cryptosystem, the size of its public key is around 187.69 Kb [8], whereas the public key of RSA for the same bit security is 3 Kb (or equivalently, 3 072 bits), [45, Table 2].

2. Motivation

The necessity of developing quantum-resistant cryptographic algorithms has also been recognized by The National Institute of Standards and Technology (NIST), which in Dec 2016 has announced a call for Public-key Post-Quantum Cryptography. There were 69 initial submissions of Public-Key Encryption algorithms. Some of them have been broken within a month after publishing, others within a year, whereas the remaining ones have been merged and improved. In Jan 2019, seventeen (17) of the encryption algorithms have been admitted to the second round. A year and a half later (Jul 2020), four finalists selected for the third round have been announced. One of them is a code-based, namely The Classic McEliece by D. Bernstein et al. This fact indicates that after a long time of research on the original encryption scheme [40], it remains one of the most proven secure public-key cryptosystems.

At the same time, this system has received only a slight improvement over all these years. Even the finalist with the proposed parameters does not lead to a solution to the main drawback of the system - the large key size.

The remaining disadvantage inspires the research focus in this thesis: to reduce the key size while preserving the security of the system.

A significant number of studies aim to minimize the key size of the McEliece type cryptosystem by using different families of error-correcting codes. Most of the proposed cryptosystems in the short term have been proven not secure. One common characteristic of these systems, besides the original one, is that they use codes with a low error-correction capability[3] [26, 2, 43].

In this work, we propose a McEliece type cryptosystem using codes with error-correction capability higher than the capability of the codes adopted until now. It can be seen as a study on the trade-off between the error-correction capability and the size of the public key. The codes with this property that we use are large minimum distance self-dual codes and punctured codes derived from them.

The codes with this property that we use are optimal self-dual codes. To the best of our knowledge, such codes have not been implemented in a code-based cryptosystem until now.

The reason is most likely twofold: first, because the optimal self-dual codes are known for the lengths up to 130, which are too small to be used for the current security requirements and second, because there is no fast decoding algorithm for these codes, an exception being the extended Golay code [49].

[3] The number of errors which the code is capable to correct.

3. Research Question

Here we define the main research question in a broad aspect. Later, when we have introduced the terminology and the existing results for McEliece type cryptosystems, we will formulate it more specifically together with the sub-questions, the solutions of which justify the main result.

RQ *To what extent can a McEliece type cryptosystem using codes with a high error-correcting capability– derived from self-dual codes– be a secure and practically applicable post-quantum cryptosystem?*

The property *secure* is considered as secure against known attacks and *practically applicable* when the key size is reduced.

Note that the research question we stated is rather involved and long. Still, this is a conscious decision we follow since we deal with a topic that requires mathematical precision.

4. Contributions

In this work, we research for the first time the relation between the security level and the key size of a McEliece type cryptosystem using a code obtained from a specific type of codes with a high error-correcting capacity. The considered codes are large minimum distance binary self-dual codes. We study and implement in SageMath a small example of the cryptosystem using a code derived from a self-dual code of length of 104 and a minimum distance of 18. A decoding algorithm, as part of the decryption process, is selected and applied. We prove that this cryptosystem has at least 22 bits security level using a key of size 0.3251 Kb, whereas the key of the original type McEliece cryptosystem with the same bit security level is at least 0.4515 Kb. Thus, our example achieves a reduction of the key of around 28%.

We determine the parameters of a putative optimal self-dual code and a punctured code derived from it, which, if implemented into a McEliece cryptosystem, would provide a bit security level of 80, 128, and 256 (quantum 67, 101, and 183 bits), respectively.

Further, for the 80 bits security level, an optimal binary self-dual code with length 1 064 has been constructed. A code with these parameters is presented here for the first time. A punctured code of it is used for the private key of the McEliece cryptosystem.

To define the McEliece type cryptosystem, we specify the algorithms in each step of the scheme. The decryption step requires a different decoding algorithm than the one used for the small example. Therefore, a new decoding algorithm suitable for the generated large optimal self-dual code is developed. It is integrated into a decryption scheme with a new strategy, namely decrypting a padded ciphertext via the self-dual code. Thus, we introduce a McEliece type cryptosystem with two private keys- an optimal self-dual code and a punctured code of it. The punctured code is used for creating the public key, while the self-dual code is for decoding in the decryption step. Thus, we show that codes with large error-correcting capacity are applicable for McEliece type cryptosystems.

By cryptanalysis, we confirm that the newly generated code provides 80 bits classical (67 bits quantum) security of the system with a key 38% smaller than the keys of the original McEliece cryptosystem. The comparison is with the family of codes of the original McEliece system, as the system with these codes has been a long time thoroughly researched, and its level of security has been proven. Moreover, one of the four finalists of the NIST competition, The Classic McEliece, uses this family of codes.

5. Thesis Outline

The rest of this work is structured as follows: Chapter 2 contains the preliminaries on error-correcting codes, the two main code-based cryptosystems, and their security, and at last, a short description of the Key Encapsulation Mechanism (KEM). In Chapter 3 we present an overview of the known McEliece type cryptosystems using different families of codes and their security status at the time of writing[4]. The next chapter presents a small example of the McEliece type cryptosystem using a binary self-dual code of length 104 with a minimum distance of 18 and a security analysis of the system. Next, we compute possible parameters of large minimum distance self-dual codes, which could provide 80, 128, and 256 classical bit security when implemented in McEliece type Cryptosystem. For the case of 80 bits security in Chapter 6, we construct an example of a self-dual code with the required parameters. Then, a new decoding algorithm suitable for the specific structure of the example code is presented together with a new decryption algorithm applying this decoding scheme. At last, we discuss limitations in the results regarding the proposed McEliece type scheme and open problems for future research.

[4] July 2021

2

Preliminaries

This chapter presents the main objects and building blocks of the McEliece cryptosystem, which are required later when constructing the scheme using codes with a high error-correcting capacity. We start by introducing the necessary terminology on error-correcting codes, followed by the formal definition of the two main code-based cryptosystems, McEliece and Niederreiter encryption schemes. Then the notion of security of the system and several common attacks on this type of systems are presented. For completeness, an informative description of the key encapsulation mechanism (KEM) is included at the end of this chapter.

The theory of error-correcting codes is initially developed to detect and correct (with certain limitations) errors in information transferred over a noisy communication channel. The introduced errors can be detected and corrected based on the redundant information sent together with the message (Figure 2.1). In Code-based cryptography, this error-correcting property is used in the decryption process.

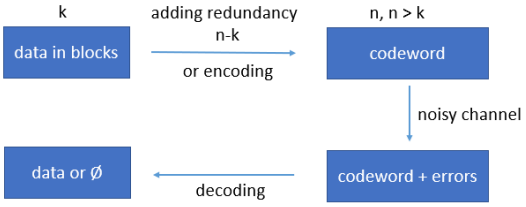

Figure 2.1: Codes for Error Correction

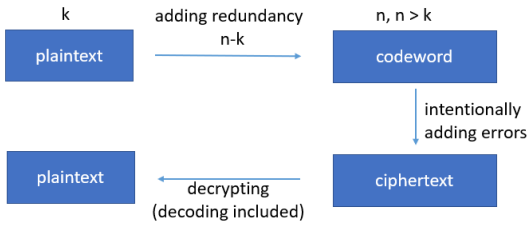

Figure 2.2: Codes for Cryptography

The first cryptosystem based on coding theory was proposed by Robert McEliece back in 1978 [40]. It is a public key cryptosystem (PKC) that uses a specific type of error-correcting codes for creating the public key. The encryption of a message block involves intentionally adding the maximum number of errors which the code can correct (Figure 2.2). To be able to define the system formally, we need a brief introduction to error-correcting codes.

Note that we will use algebraic structures such as field, polynomial ring, principle ideal, vector space without introducing them. For completeness, we include their definitions in Appendix 1.

1. Error-correcting codes

1.1. Linear block codes

Here we are interested in a particular type of error-correcting codes, namely linear block codes over a finite field, because of their nice algebraic properties that we will discuss in the places where we use them.

Let $\mathbb{F}_q$ be the finite field with q elements, where q is a prime or a power of a prime number, and $\mathbb{F}_q^n$ is the standard vector space over $\mathbb{F}_q$. Let $u = (u_1, \ldots, u_n)$ and $v = (v_1, \ldots, v_n)$ be vectors in $\mathbb{F}_q^n$. The *inner product* $u \cdot v$ in $\mathbb{F}_q^n$ is defined as $u \cdot v = u_1v_1 + u_2v_2 + \cdots + u_nv_n$. Two vectors u and v are called *orthogonal* if $u \cdot v = 0$.

The (*Hamming*) *distance* $d(x, y)$ between two vectors $x, y \in \mathbb{F}_q^2$ is the number of coordinates in which x and y differ, i.e. $d(x, y) = \#\{i|\ 1 \leq i \leq n, x_i \neq y_i\}$. The Hamming distance is a metric[1] on the linear space $\mathbb{F}_q^n$.

The (*Hamming*) *weight* $wt(x)$ of a vector $x \in \mathbb{F}_q^n$ is the number of its nonzero coordinates, i.e. $wt(x) = \#\{i|\ 1 \leq i \leq n, x_i \neq 0\}$.

Clearly, the distance between two vectors x and y in $\mathbb{F}_q^n$ is the weight of the vector $x - y$.

Definition 1.1. If C is a k-dimensional subspace of $\mathbb{F}_q^n$, C is called *linear code* over $\mathbb{F}_q$ of length n and dimension k, or shortly an $[n, k]_q$-code.

Usually, the elements of a code are called *codewords*.

Definition 1.2. The *minimum distance* of a code C is $d = min\{d(x, y)|\ x, y \in C,\ x \neq y\}$.

An $[n, k, d]_q$-code is an $[n, k]_q$-code with a minimum distance d.

Definition 1.3. The *minimum weight* of a code C is the smallest weight of any nonzero codeword, i.e. $wt(C) = min\{wt(x)|\ x \in C,\ x \neq 0\}$

The minimum distance of a code determines the error-detecting and error-correcting capability of the code, more specifically:

Theorem 1.1. [50, p.12] An $[n, k, d]_q$-code C can detect $s < d$ errors and correct $t \leq \lfloor \frac{d-1}{2} \rfloor$ errors.

Theorem 1.2. [39, p.10] For a linear code, the minimum distance is equal to the minimum weight.

Since a linear code is a subspace, it can be determined by a set of basis vectors. A *generator matrix* G of a $[n, k]_q$-code C is a $k \times n$ matrix where the rows of G form a basis in C. A generator matrix G is said to be in *systematic*, or *standard form* if $G = [I_k \mid G_1]$, where I_k is the identity $k \times k$ matrix.

A *parity check* matrix H of an $[n, k]_q$-code C is an $(n-k) \times n$ matrix, such that H is orthogonal to every codeword of C:

$$c \in C \Leftrightarrow H \cdot c^\top = 0^\top. \tag{2.1}$$

The parity check matrix generates also a linear code, which is *orthogonal* or *dual* to the code C. The orthogonal code is denoted by $C^\perp$, $C^\perp = \{v \in \mathbb{F}_q^n :\ u \cdot v = 0,\ \forall u \in C\}$.

The code C is called *self-orthogonal* if $C \subset C^\perp$, and *self-dual* if $C = C^\perp$.

If G is in a standard form $[I_k \mid G_1]$ one can show that for the matrix $[-G_1^\top \mid I_{n-k}]$ the following holds:

$$[I_k \mid G_1] \cdot [-G_1^\top \mid I_{n-k}]^\top = -G_1 + G_1 = 0,$$

[1] Holds the properties: $d(x, y) = 0 \Leftrightarrow x = y$; $d(x, y) = d(y, x)$; $d(x, y) \leq d(x, z) + d(z, y)$.

which implies that $[-G_1^\top \mid I_{n-k}]$ is a parity-check matrix for the code C.

A linear code is called *binary* if it is a code over $\mathbb{F}_2$. To illustrate the terminology, we give an example of a binary code.

Example 1.1. A binary $[8,3]$-code C has a generator matrix

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

Since the rows form a basis, each vector in C is a linear combination of them. Let v_1, v_2, v_3 be the rows of G, i.e. $v_1 = (1,0,0,0,1,1,1,1), v_2 = (0,1,0,1,0,1,1,1), v_3 = (0,0,1,1,1,0,1,1)$. Then each codeword $c = a_1v_1 + a_2v_2 + a_3v_3$, where $a_1, a_2, a_3 \in \mathbb{F}_2$. Then:

$$C = \{0,\ v_1,\ v_2,\ v_3,\ v_1+v_2,\ v_1+v_3,\ v_2+v_3,\ v_1+v_2+v_3\}.$$

For example:

$$v_1 + v_2 = (1,0,0,0,1,1,1,1) + (0,1,0,1,0,1,1,1) = (1,1,0,1,1,0,0,0).$$

The weight of this sum is 4. To define the minimum weight of the given code we need the weights of all the nonzero codewords. They are $wt(v_1) = 5$, $wt(v_2) = 5$, $wt(v_3) = 5$, $wt(v_1+v_2) = 4$, $wt(v_1+v_3) = 4$, $wt(v_2+v_3) = 4$, $wt(v_1+v_2+v_3) = 5$. Then the minimum weight is $d = 4$, which implies that the code can detect up to 3 errors and correct up to $t \leq \lfloor \frac{d-1}{2} \rfloor = 1$ error.

Since the generator matrix is in a systematic form, we can directly write the parity-check matrix as $[-G_1^\top \mid I_{n-k}]$, i.e.

$$H = [-G_1^\top \mid I_5] = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The inner product of v_1 and v_2 is $v_1 \cdot v_2 = 0+0+0+0+0+1+1+1 = 1$ in $\mathbb{F}_2$, which implies that the vectors v_1 and v_2 are not orthogonal. Then the given binary $[8,3]$-code C is not self-orthogonal and therefore it cannot be self-dual.

Two linear codes C_1 and C_2 of length n are called *permutation equivalent* if one can be obtained from the other by a permutation of coordinates. That is, if there exist a permutation $\sigma \in S_n$, S_n- the symmetric group of degree n, such that $\sigma(C_1) = C_2$. In particular, if a permutation σ maps a code C to itself, then σ is called *a permutation authomorphism* of the code.

1.2. Basic decoding schemes

A message m is encoded in a codeword w and transmitted through a noisy channel. The received vector r can be considered as $r = w + e$, where e is the error vector. The decoder must decide from the received r which codeword was sent.

Note. *Here we consider only symmetric channels, which means that the probability of flipping 0 into 1 during transmission is equal to the probability of flipping 1 into 0 during transmission, and this probability is less than 0.5. If it is larger than 0.5, then the measurement result at the receiver must be reversed when read. Therefore, the probability of receiving a vector of length n with one error is $p < 1/2$, with two errors, is p^2, as $p^2 < p < 1/2$, and so on. Thus, we always assume that the errors during transmission are as few as possible, and the decoder is always searching for the closest codeword to the received vector.*

If e is the zero vector, then there is no error and $r = w$, i.e. w is a codeword. Otherwise, depending on the error vector e, r can be a codeword or not.

Finding the codeword whose distance to r is as small as possible can be done by exhaustive search comparing r to all the codewords and pick the closest one. This approach is called *nearest neighbour decoding*. For small codes, this approach is feasible, but for larger codes, it would be computationally very expensive. The algebraic properties of linear codes are useful to define more efficient decoding strategies using the concept of a syndrome.

Definition 1.4. Let C be an $[n, k, d]_q$ code with parity-check matrix H. The *syndrome* of a vector $v \in \mathbb{F}_q^n$ is the vector $S(v) = H \cdot v^\top$ of length $n - k$ and coordinates of $\mathbb{F}_q$.

By Equation (2.1), $w \in C$ if and only if $H \cdot w^\top = 0^\top$, which is equivalent to saying that, a vector of $\mathbb{F}_q^n$ is a codeword if and only if its syndrome is 0.

Since $r = w + e$, then $H \cdot r^\top = H \cdot (w + e)^\top = H \cdot w^\top + H \cdot e^\top = 0^\top + H \cdot e^\top = H \cdot e^\top$, that is $S(r) = S(e)$. Thus, the syndrome of the error vector is known to the decoder and the question of finding the error e is shifted to how to find a vector whose syndrome is equal to $S(e)$.

Next we introduce the concept of coset and its relation to syndromes. Let $u \in \mathbb{F}_q^n$. Then the set $u + C = \{u + c |\ c \in C\}$ is called a *coset* of C. If we have two elements of the same coset, $u_1, u_2 \in u + C$, then $u_1 = u + c_1$, $u_2 = u + c_2$ for some $c_1, c_2 \in C$. Further we have $S(u_1) = S(u + c_1) = S(u)$, $S(u_2) = S(u + c_2) = S(u)$, which means that all the elements of a coset have the same syndrome.

Let we assume that these two elements u_1, u_2 have weight t_u, where $t_u \leq \lfloor \frac{d-1}{2} \rfloor$. Then the vector $u_1 - u_2$ has weight at most $2t_u < d$. Substitution of u_1 and u_2 in $u_1 - u_2$ leads to $u_1 - u_2 = c_1 - c_2 \in C$. Thus, $u_1 - u_2$ must be a codeword and therefore the weight of $u_1 - u_2$ is greater or equal to d which is a contradiction to the wight of $u_1 - u_2$ to be at most $2t_u < d$. Thus, the assumption for existence of two elements of weight $t_u \leq \lfloor \frac{d-1}{2} \rfloor$ in the same coset is not true. Hence, every coset with a minimum weight of up to t has a unique minimum weight vector. This element is called *coset leader*.

We can now define the following maximum likelihood decoding algorithm.

Algorithm 1: Syndrome Decoding

1 For a received vector r calculate $S(r) = r \cdot H^\top$
2 In the coset with the syndrome equal to $S(r)$ find the coset leader e'
3 Decode r into $w' = r - e'$

This algorithm is still computationally expensive. It needs the coset leaders of q^n/q^k number of cosets and to store their syndromes, which is practically infeasible for a large code. In the next sections, we will discuss other decoding algorithms appropriate for larger codes.

1.3. Basic Information Set Decoding

The Information Set Decoding (ISD) technique was introduced by Prange in 1962 [54]. An *information set* for a $[n, k, 2t+1]_2$ code $\mathcal{C}$ is any subset $A = \{i_1, \cdots, i_k\}$ of k coordinates such that, for any given set of values $b_i \in \mathbb{F}_2$, with $i \in A$, there is a unique codeword $c \in C$. Thus, the information set consists of any k indices such that the corresponding k columns of a generator matrix of $\mathcal{C}$ have rank k.

Let $r = mG + e$ be the received vector after transmission of the encoded message m by a generator matrix G of the code $\mathcal{C}$, where e is an error vector of weight t. Further, let S be a set of k random coordinates defined as $S = \{i_1, i_2, \ldots, i_k \mid 1 \leq i_j \leq n, i_j \neq i_s\}$.

Following [7], we describe the *Basic Information Set Decoding* as follows:

- Choosing S randomly we make the following two assumptions: first, all entries of the error vector e indexed by the set S are 0, i.e. $e_{i_1} = e_{i_2} = ... = e_{i_k} = 0$ and second, S is an information set, i.e. if we denote the columns of G with $g_1, g_2, ..., g_n$, then the matrix $[g_{i_1}, g_{i_2}, ..., g_{i_k}]$ is nonsingular.
- Consider the projection $\phi : G \longmapsto G_S$ which maps the generator matrix G into the k randomly chosen columns of it.

  Then, $\phi(r) = \phi(mG + e) = \phi(mG) + \phi(e) = mG_S$. Since G_S is nonsingular by the second assumption, we can find the message m as $m = \phi(r){G_S}^{-1}$.
- Obtain $e = r - mG$. If $wt(e) = t$ then the message is m. Otherwise, go to the beginning and try another set S.

Algorithm 2 presents the Basic Information Set Decoding in a more structured way as pseudocode.

Algorithm 2: Basic Information Set Decoding

1 Choose k coordinates $S = \{i_1, i_2, ..., i_k\}$ and form the matrix G_S. Repeat this step until $det(G_S) \neq 0$
2 Calculate ${G_S}^{-1}$
3 Compute $\phi(r){G_S}^{-1} = m$, where $\phi(r)$ is the vector of the k chosen coordinates of r
4 Compute $e = r - mG$. If $wt(e) = t$ then the message is m, else repeat from 1.

1.4. Quantum Information Set Decoding

Let $r = mG + e$, G and e be defined as in Subsection 1.3. The structure of a Basic Quantum Information Set Decoding is given as pseudocode in Algorithm 1.4.

Algorithm 3: Basis Quantum Information Set Decoding

1 Choose k coordinates $S = \{i_1, i_2, ..., i_k\}$ and form the matrix G_S. If $det(G_S) \neq 0$, find ${G_S}^{-1}$ else, giving up
2 Compute $(r_{i_1}, r_{i_2}, ..., r_{i_k}) \cdot {G_S}^{-1} = m$, $m \in \mathbb{F}_2^k$
3 Compute $mG \in \mathbb{F}_2^n$
4 Compute $e = r - mG$. If $wt(e) \neq t$ then giving up
5 Returns 0.

According to [7], randomly searching for a root of the function in Algorithm 3 can succeed in approximately $\binom{n}{k}/0.29\binom{n-t}{k}$ iterations, where one iteration of this function has around $O(n^3)$ bit operations. Grover's algorithm uses about square root of the number of iterations, i.e. $\sqrt{\binom{n}{k}/0.29\binom{n-t}{k}}$. Then, the complete number of qubit operations for finding a solution is

$O(n^3)\sqrt{\binom{n}{k}/0.29\binom{n-t}{k}}$. Note that the meaning of 0.29 is that, on average, 29% of the selected matrices G_S are nonsingular when G is a generator matrix of a Goppa code (introduced at the end of the section).

1.5. Decoding problem

Based on the previous two subsections, we can define the decoding problem. Let an $[n, k, 2t+1]$ code C have a generator matrix G and a parity-check matrix H. Let a message m be encoded as mG and transmitted, and the received vector be denoted by $r = mG + e$. Then the decoding problem is one of the following equivalent problems [47]:

1. Find a codeword x of C that is the closest one to the received vector r, i.e. find x such that $d(x, r) = min\{d(a, r) \mid a \in C\}$;
2. Find an error vector e of the coset $r + C = \{r + a \mid a \in C\}$ of minimum weight;
3. Find an error vector e with minimum weight in this coset of C that has a syndrome equal to the syndrome of the received vector r.

When in the above problem we fix an upper bound for the weight of the error vector the problem is called *Computational Syndrome Decoding* (CSD) problem. It is defied as:

Definition 1.5. [47] ***(CSD)*** For a given an $[n, k, 2t+1]$ code C with a parity-check matrix H, $(n-k) \times n$, a vector s of length $n-k$ and a positive integer t', $t' \leq t$, find a vector e of weight t' in the coset of C with syndrome s.

In [5], the following main result regarding the CSD problem for an arbitrary linear code is proven:

Theorem 1.3. [5] The general decoding problem for linear codes and the general problem of finding the weights of a linear code are both $\mathcal{NP}$-complete.

1.6. Cyclic codes

Consider the factor ring $\mathcal{R} = \mathbb{F}_q[x]/(x^n - 1)$ which denotes the polynomials of $\mathbb{F}_q[x]$ mod $(x^n - 1)$. Then $\mathcal{R}$ consists of all polynomials with coefficients of $\mathbb{F}_q$ and power at most $n-1$. Moreover, we assume that n and q are co-prime.

Any polynomial $c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ of $\mathcal{R}$ can be associated with the vector $(c_0, c_1, c_2, \ldots, c_{n-1})$ of $\mathbb{F}_q^n$.

Definition 1.6. [39] An $[n, k]_q$ code C is called *cyclic* if any cyclic shift of a codeword is also a codeword.

That is, whenever $(c_0, c_1, c_2, \ldots, c_{n-1})$ is in C, then so is $(c_{n-1}, c_0, c_1, \ldots, c_{n-2})$. Using the corresponding polynomials in $\mathcal{R}$, it is equivalent to: if $c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ is in C then $xc(x)$ is also in C since

$$xc(x) = x(c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}) = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-2} \mod (x^n - 1).$$

Thus, a right cyclic shift with one position is identical with a multiplication with x. Therefore, $c(x)g(x) \in C$ for any $g(x) \in \mathcal{R}$, which means that the cyclic code C is an ideal in $\mathcal{R}$.

From the definition for cyclic code C it follows that the permutation $(1, 2, 3, \ldots, n)$ is an automorphism of C.

Theorem 1.4. [39, p.190] Let C be a nonzero cyclic code in $\mathcal{R}$. Then:

(a) There is a unique monic polynomial $g(x)$ of minimum degree in C.

(b) $C = \langle g(x) \rangle$, i.e. C is a principle ideal in $\mathcal{R}$ generated by $g(x)$.

(c) $g(x)$ is a factor of $x^n - 1$.

(d) Any $c(x) \in C$ can be written uniquely as $c(x) = f(x)g(x)$ in $\mathbb{F}_q[x]$, where $deg(f(x)) < n - deg(g(x))$. The dimension of C is $n - deg(g(x))$. Thus, the message $f(x)$ becomes the codeword $f(x)g(x)$.

(e) If $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_r x^r$, then C is generated, as a subspace of $\mathbb{F}_q^n$, by the rows of the generator matrix

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_r & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{r-1} & g_r & \cdots & 0 \\ & & & \vdots & & & & \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & & g_r \end{pmatrix}$$

$$= \begin{pmatrix} g(x) & & & & \\ & xg(x) & & & \\ & & \cdots & & \\ & & & x^{n-r-1}g(x) & \end{pmatrix}.$$

The polynomial $g(x)$ in Theorem 1.4 is called the *generator polynomial* for the cyclic code C.

If $C = \langle g(x) \rangle$, then $g(x)|(x^n - 1)$. Denote $h(x) = \frac{x^n-1}{g(x)}$. Consider an element $c(x) \in C$. Then $c(x) = f(x)g(x)$ for some element $f(x) \in \mathbb{F}_q[x]$ and, $h(x)c(x) = h(x)f(x)g(x) = 0$ in R. The opposite is also true ([39, p.195]), i.e. if $c(x)h(x) = 0$ for $c(x) \in \mathcal{R}$ then $c(x) \in C$.

The polynomial $h(x) = \frac{x^n-1}{g(x)}$ is called *check polynomial* of C.

Theorem 1.5. [39, p.195-196] Let $\mathcal{C}$ be a nonzero cyclic code in $\mathcal{R}$ with generator polynomial $g(x)$ and check polynomial $h(x) = (x^n - 1)/g(x)$. Then:

(i) the dual code $C^\perp$ is cyclic and has generator polynomial $g^\perp(x) = x^{deg(h(x)}h(x^{-1})$ (the reciprocal polynomial of $h(x)$);

(ii) if $h(x) = h_0 + h_1 x + \cdots + h_k x^k$, then the parity check matrix of C is

$$H = \begin{pmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ & & \vdots & & & & & \\ 0 & \cdots & 0 & & h_k & \cdots & h_1 & h_0 \end{pmatrix}.$$

A polynomial $e(x)$ of $\mathcal{R}$ is called an *idempotent* if $e(x) = e(x)^2$.

Example 1.2. Let $\mathcal{R}_1 = \mathbb{F}_2[x]/(x^7 - 1)$, $f(x) = x + x^2 + x^4$. Then,

$f(x)^2 = (x + x^2 + x^4)^2 = x^2 + x^4 + x = f(x)$ in $\mathcal{R}_1$. Therefore $f(x)$ is an idempotent in $\mathcal{R}_1$.

The next theorems in this subsection are restricted within the factor ring $\mathbb{F}_2[x]/(x^n - 1)$ where n is odd.

Theorem 1.6. [51, p.54] Let C be a cyclic code in $\mathcal{R}$. Then there exists a unique idempotent $e(x) \in C$ such that $C = \langle e(x) \rangle$. If $e(x)$ is a nonzero idempotent in C, then $C = \langle e(x) \rangle$ if and only if $e(x)$ is a unit element of C, where a unit element means a neutral element in C regarding multiplication.

The unique idempotent of Theorem 1.6 is called the *generating idempotent* of C.

The next theorem presents results for calculating an idempotent of any ideal in $\mathcal{R}$. For that we introduce the *formal derivative* of a polynomial in $F_q[x]$.

For $f(x) = a_0 + a_1 x + \cdots + a_n x^n \in F_q[x]$ the *formal derivative* of $f(x)$ is defined as the polynomial $f'(x) = a_1 + 2a_2 x + 3a_3 x^2 + \cdots + n a_n x^{n-1} \in F_q[x]$.

Theorem 1.7. [39, p.223] Let $h(x)$ be a polynomial which divides $x^n - 1$. Let $e(x)$ be the idempotent of the ideal with generator polynomial $g(x) = \frac{x^n-1}{h(x)}$. Then:

$$e(x) = (x^n - 1)\left(\frac{x h'(x)}{h(x)} + \delta\right),$$

where $h'(x)$ is the derivative of $h(x)$, and $\delta = 0$ if the degree of $h(x)$ is even, and 1 if the degree of $h(x)$ is odd.

Theorem 1.8. [51, p.55] Let C be a cyclic $[n, k]$-code with generator idempotent $e(x) = \sum_{i=0}^{n-1} e_i x^i$. Then the $k \times n$ matrix

$$\begin{pmatrix} e_0 & e_1 & e_2 & \cdots & e_{n-2} & e_{n-1} \\ e_{n-1} & e_0 & e_1 & \cdots & e_{n-3} & e_{n-2} \\ & & & \vdots & & \\ e_{n-k+1} & e_{n-k+2} & e_{n-k+3} & \cdots & e_{n-k-1} & e_{n-k} \end{pmatrix}$$

is a generator matrix for $\mathcal{C}$.

The next result is valid in $\mathbb{F}_q$. We consider the factorization of the polynomial $x^n - 1$ over $\mathbb{F}_q$ $x^n - 1 = h_0(x)h_1(x)\ldots h_s(x)$, where $h_0 = x - 1$ and denote $g_j(x) = \frac{x^n-1}{h_j(x)}$.

Let $I_j = \langle g_j(x) \rangle$ be the ideal of $\mathcal{R}$ generated by $g_j(x)$ for $j = 0, 1, \ldots, s$ and, $e_j(x)$ be the generator idempotent of I_j for $j = 0, 1, \ldots, s$.

Theorem 1.9. [51, p.56]

(i) the ideal I_j, $j = 0, 1, \ldots, s$, is a minimal ideal of $\mathcal{R}$;

(ii) $\mathcal{R} = I_0 \oplus I_1 \oplus \cdots \oplus I_s$;

(iii) I_j is a field which is isomorphic to the field $\mathbb{F}_q^{deg(h_j(x))}$, $j = 0, 1, \ldots, s$;

(iv) $e_i(x)e_j(x) = 0$, $i \neq j$;

(v) $\sum_{j=0}^{s} e_j(x) = 1$.

The idempotents $e_j(x)$, for $j = 0, 1, \ldots, s$, are called the *primitive idempotents* of $\mathcal{R}$.

1.7. Self-dual codes

An $[n,k]_q$ code $\mathcal{C}$ is called *self-dual* if $\mathcal{C} = \mathcal{C}^{\perp}$. It is known that the weight of any codeword of a self-dual code is even. A binary self-dual code $\mathcal{C}$ is called *doubly-even* if the weight of every codeword is divisible by four, and *singly-even* if there is at least one codeword of even weight not divisible by 4, i.e., weight $\equiv 2 \pmod 4$ [52, p.11].

Upper bounds for the minimum weight of a binary self-dual $[n, n/2, d]$ code are given in [55]:

$$\begin{aligned} d &\leq 4\lfloor \tfrac{n}{24} \rfloor + 4, \quad if \quad n \not\equiv 22 \pmod{24}, \\ d &\leq 4\lfloor \tfrac{n}{24} \rfloor + 6, \quad if \quad n \equiv 22 \pmod{24} \end{aligned} \tag{2.2}$$

Stricter bounds are known for some specific lengths, like for $n = 78$ the maximum d is 14 instead of 16 [20]. Self-dual codes which reach the minimum weight bounds are called *extremal*, whereas self-dual codes with the largest minimum weight for a given length among the known ones are called *optimal*. In [68], it is proven that binary extremal double-even self-dual codes do not exist for lengths $n > 3\,928$.

It is known that with an increasing length, the number of self-dual codes grows fast. For example, where are 85 inequivalent double-even self-dual codes of length 32 and at least 17 000 of length 40.

For further reading about families of self-dual codes and their properties, we recommend [56]. Some results on binary self-dual codes with a particular structure are presented in Section 1.1.

1.8. Goppa codes

Here we present only the parameters of the codes. These parameters will be used later in Chapter 4 and Chapter 5. A detailed description of Goppa codes and their properties the reader can find in [39, p.338-368].

The Goppa codes used in the McEliece type cryptosystem have the parameters length n, dimension k and minimum weight d where $n \leq 2^m$, $k = n - mt$, $d \geq 2t+1$ for some positive integer m and error-correcting capability t. For example, the Goppa code used in [40] is constructed for $m = 10$ and $t = 50$, i.e. a binary $[1\,024, 524, d \geq 101]$ code.

It is also known that for particular m and t, there exist many inequivalent Goppa codes.

2. Code-Based Cryptosystems

2.1. McEliece Cryptosystem

The McEliece Cryptosystem is the first code-based cryptosystem proposed by Robert McEliece in 1968 [40]. The security of the McEliece cryptosystem relies on the difficulty of the general decoding problem Section 1.5. The original cryptosystem uses a binary $[1024, 524]$ code with an error-correcting capability of 50 errors. The steps of the encryption scheme are as follows:

1. *Define the system parameters:* k - the length of the message block, n - the length of the ciphertext, t - the number of the intentionally added errors (equal to the error-correcting capability of the implemented linear code).

2. *Key generation:* define: G - a generator matrix of an $[n, k, 2t+1]$ code for which there is a fast decoding algorithm; P - a random $n \times n$ permutation matrix and S - a random dense $k \times k$ non-singular matrix and, compute $G' = SGP$, S^{-1} and P^{-1} - the inverse of P and S. Note that G' generates a linear code with the same n, k and t. Then, (G', t) - *Public key*, (G, P, S) or (Dec_G, P, S) - *Private key*, where Dec_G is the fast decoding algorithm.

3. *Encryption:* split the data for encryption into k-bit blocks. Then each block m is encrypted as $r = G'm + e$, where e is a random vector of length n and weight t.
4. *Decryption:* The received vector r is decrypted as follows:
   (a) Compute $r' = rP^{-1}$, which is $mSG + eP^{-1}$.
   (b) Decode r' into a codeword c' using the efficient decoding algorithm for the code with generator matrix G, $c' = mSG$.
   (c) Compute c such that $cG = c'$ (If G is in a systematic form, then c is the first k bits of c').
   (d) Compute $m = cS^{-1}$.

The presented scheme can be applied with any linear code for which a fast decoding algorithm is known. In particular, the original system in [40] employs a binary $[1\,024, 524, 101]$ Goppa code. We call this encryption scheme a *McEliece type cryptosystem* if it uses a family of codes other than the binary Goppa codes.

The existence of many inequivalent codes of the family of codes implemented in a McEliece type cryptosystem is an initial requirement for the security of the system.

2.2. Niederreiter Cryptosystem

We describe the original Niederreiter Cryptosystem as presented in [44].

Let C be a $[n, k, 2t+1]_q$-code and let H be a perity-check matrix of C. Then H is $(n-k) \times n$ matrix and $c \in C \iff H \cdot c^\top = 0^\top$. Thus, H inducts bijection mapping between $\mathbb{F}_q^n$ and $\mathbb{F}_q^{n-k}$ for the vectors of weight $\leq t$ in both. Remind that for every $v \in \mathbb{F}_q^n$, $S(v) = H \cdot c^\top$ is the syndrome of the vector(Def 1.4).

Why is that bijection? From Section 1.2, if $u, v \in \mathbb{F}_q^n$, where $wt(u) \leq t$, $wt(v) \leq t$, then u, v are coset leadres of different cosets, i.e. $S(u) \neq S(v)$. From $H \cdot u^\top \neq H \cdot v^\top \iff H(u^\top - v^\top) \neq 0$. If we assume that $u = v$ then $H(u^\top - v^\top) = H \cdot 0^\top = 0$. Therefore, H indeed induces a bijection.

1. *Define the system parameters:* n - the length of the message block, $n-k$ - the length of the ciphertext, t - the weight of the message block (equal to the error-correcting capability of the implemented linear code).
2. *Key generation:* define: H - a parity-check matrix of an $[n, k, 2t+1]_q$ code C for which there is a fast decoding algorithm Dec_C; P - a random $n \times n$ permutation matrix with entries of $\mathbb{F}_q$; S - a random dense $(n-k) \times (n-k)$ non-singular matrix with entries of $\mathbb{F}_q$. Compute $K = SHP$, S^{-1} and P^{-1} - the inverse of P and S.

   Then, (K, t) - *Public key*, (H, P, S, Dec_C)- *Private key.*
3. *Encryption:* map the data for encryption into n-bit blocks, each with weight at most t. Then each block m is encrypted as $r = Enc(m) = Km^\top$. The ciphertext r is the syndrome of the message.
4. *Decryption:* The received vector r is decrypted as follows:
   (a) Compute $r' = S^{-1}r$, which is $r_1 = S^{-1}r = S^{-1}Km^\top = S^{-1}SHPm^\top = HPm^\top = H(mP^\top)^\top$. Note that $mP^\top$ is also a vector of weight $\leq t$ since $wt(m) \leq t$ and the matrix P is a permutation matrix with entries of $\mathbb{F}_q$. The vector $r_1 = H(mP^\top)^\top$, i.e. r_1 is the syndrome of the vector $mP^\top$ of weight at most t.
   (b) Decode, by syndrome decoding, r_1 into the leader of the coset whose syndrome is equal to r_1, i.e. into coset leader r_2 of $u + C$, such that $S(u + C) = r_1$. Thus, r_2 must be equal to $mP^\top$.

(c) Compute m as $P^{-1}{r_2}^\top = P^{-1}Pm^\top = m$.

(d) Map back the decrypted n-bit block m with weight at most t into the initial data.

2.3. Comparison

An advantage of the Niederreiter cryptosystem is that the plaintext is of length n while the ciphertext to be transferred, is of length $n-k$, which is shorter than n. In McEliece system it is the opposite, the plaintext length is $k < n$, the ciphertext length is $n > n-k$. The Niederreiter Cryptosystem is often referred as the *dual* McEleiece cryptosystem.

On the other hand, a shortcoming of the Niederreiter cryptosystem is that the message weight must be at most t. Therefore, before encoding, each message must be mapped to such an n bit sequence with maximum weight t, and when decrypted, the message must be mapped back.

In [36] it has been proven that both systems are equivalent, then their security levels are equal, i.e., if an attack is successful in breaking one of the systems, it will be successful in breaking the other system as well.

In order to improve the size of the public key of both systems, the public key is transformed into a systematic form $[I_k|G'_{k,n-k}]$ or $[H'_{n-k,k}|I_{n-k}]$ and instead of the complete matrix to be uploaded, only the part of the corresponding matrix without the identity matrix is published, i.e. $G'_{k,n-k}$ or $H'_{n-k,k}$.

3. Security

3.1. Public Key Encryption Schemes

Similar to the McEliece and Niederreiter encryption schemes described in the last subsections, any public key encryption scheme contains the three building blocks: key generation, encryption, and decryption. Here we define it formally.

Definition 3.1. [32, p.378] A public-key encryption scheme is a triple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a pair of keys (pk, sk), where pk is the *public key*, sk is the *private key*, and both have a length of at least n.

2. The probabilistic encryption algorithm Enc takes as input the pk and a message m from some message space, and it outputs a ciphertext c, $c = \mathsf{Enc}_{pk}(m)$.

3. The deterministic decryption algorithm Dec takes as an input (sk, c) and outputs a message m or a failure denoted by $\perp$. We write this as $m = \mathsf{Dec}_{sk}(c)$.

It is required that, $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$ for any (legal) message m excepting a negligible probability of decryption error when $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) \neq m$.

The encryption and decryption algorithms are polynomial-time, i.e., efficient for the sender and the receiver. Besides that, for an adversary who can intercept the communication and obtain a batch or batches of ciphertexts, it must be extremely difficult to recover the messages from the ciphertexts without knowing the secret key and the decryption algorithm (or one wayness of the encryption function).

We refer [32, p.43-52] and [59] for detailed definitions of probabilistic polynomial-time algorithms, a negligible probability, security games, etc., which we use here without introducing.

Defining the security of an encryption system includes defining the following three components:

1. *Adversarial goals* - define the attacker's objectives and define when an attack is considered successful (or the winning condition of a security game).
2. *Attack models* - the amount of information that the attacker has about the encryption system, about the plaintext and the ciphertext (or which oracles in a security game are allowed to be used by the attacker).
3. *Computational model* - the amount of computing resources available to the attacker in order to succeed.

The *adversarial goals* can be:

- *a total break* of the system, which is obtaining the secret key and discovering a functionally equivalent algorithm to the decryption algorithm;
- *a partial break* - the attacker is able to decrypt some ciphertexts with a certain probability and not able to discover the secret key;
- *a distinguishing break* - when the attacker is able to distinguish the ciphertexts of two messages with a probability higher than 1/2 (as in a security game with two oracles).

The common *attack models* are:

- The Ciphertext-Only Attack (COA) model assumes that the adversary only observes a ciphertext (or multiple ciphertexts) without the secret key and without the decryption algorithm.
- The Known- Plaintext Attack (KPA) model assumes access to one or more plaintext/ciphertext pairs generated using some encryption key and algorithm.
- The Chosen-Plaintext Attack (CPA) model assumes the adversary can request and receive plaintext/ciphertext pairs for self-chosen plaintexts, i.e., has access to an encryption oracle. This can be done only once and together for all plaintexts in one batch.
- The adaptive Chosen-Plaintext Attack (CPA2) model is the CPA model with the modification that the attacker can obtain in multiple batches plaintext/ciphertext pairs for self-chosen plaintexts. Then the attacker can adapt each of the next plaintexts by observing the results of previous pairs.
- The Chosen-Ciphertext Attack (CCA) model assumes all from the CPA2 model with the added advantage for the attacker: he/she can request and receive the decrypted messages of a self-chosen single batch of ciphertexts, i.e., has access to both: an encryption oracle and decryption oracle. The ciphertexts in the request must be different from the ciphertext in question.
- The Adaptive Chosen-Ciphertext Attack (CCA2) model is the CCA model with one more advantage: the attacker is allowed to submit multiple batches of ciphertexts of his/her own choice for decryption. Thus, the attacker is allowed to adapt the requests in response to the previous plaintext/ciphertext pairs.

The attack models above are ordered in the increasing power of the attacker. Thus, an encryption scheme has the highest security if it is secure against the last, CCA2, attack model.

There is at least one more common attack model, the side-channel attack model. This model attacks the difficulty of the mathematical problem on which the security of the encryption scheme

is based by exploring possible vulnerabilities in the software/hardware implementation of the encryption scheme.

- A Side-Channel Attack (SCA) assumes that the attacker has access to the system (decryption device) or is very close to it so that he/she can measure the power consumption, electromagnetic leaks, or other signals. Analysing these measurements, the attacker can gain information about the operations of the decryption device and, therefore, about the decoding algorithm.

Notice that in a public key encryption system, the adversary always has access to the following information: the encryption parameters, the encryption algorithm and the public key. Therefore the attack model can be at least CPA2.

There are two categories of adversaries in terms of the *computational resources* at their disposal to succeed:

- an attacker with available unbounded computing power;
- an attacker with available only a polynomial (in some security parameter) amount of computing power - the best attack against a cryptosystem is by providing at most N operations, with N being a very large number.

We note that CCA to the end of this section corresponds to the notation of CCA2 above. Following [59], we define when a public key encryption scheme is called IND-CCA secure. It will be used further in Section 4.

An encryption scheme is said to have *perfect security* (or information-theoretic security) if an adversary with *infinite computing power* can learn nothing about the plaintext from a given ciphertext. In probabilities, it is: the a posteriori probability that a message m was sent, conditioned on the given ciphertext c, should be equal to the a priori probability that m was sent:

$$Pr[M = m \mid C = c] = Pr[M = m]$$

Semantic security is like perfect security, but the adversary is with *bounded computing power* and, in polynomial time, the adversary can learn nothing about the message m having the ciphertext c. In this case, the run time can be considered bounded by a polynomial function of the key size.

A public key encryption algorithm is said to have *indistinguishable encryptions against a chosen ciphertext attack* (IND-CCA secure) if for all probabilistic polynomial time adversaries A, the advantage of the adversary in the IND-CCA security game, Alg. 4, is negligible [32]. That is:

$$Pr(Game_{output} = 1) = Pr(A_{win}) \leq \frac{1}{2} + negl(n),$$

where $negl$ is a negligible function and n is the size of the public key.

Algorithm 4: [32, p.414] IND-CCA security game for public key encryption scheme.

1 Run $\mathsf{Gen}(1^n)$ to obtain a pair of keys pk, sk (public, private).

2 The adversary A is given pk and access to a decryption oracle $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$. It outputs a pair of messages $m_0,\ m_1$ of the message space of the same length.

3 A uniform bit $b \in \{0, 1\}$ is chosen and then a ciphertext $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m_b})$ is computed and given to A.

4 A continues to interact with the decryption oracle, but may not request a decryption of c itself. Finally, A outputs a bit b_0.

5 The output of the experiment is defined to be 1 if $b_0 = b$, and 0 otherwise.

3.2. Cryptanalysis of the McEliece Cryptosystems

Like any other public encryption scheme, the McEliece cryptosystem provides the following advantages to the attacker: the encryption parameters, the encryption and decryption algorithms, and the public key. Thus, the adversary can also select any plaintext and compute the corresponding ciphertext. Therefore, as was mentioned in the previous section, we assume that the attack model always is at least CPA2.

Concerning the adversary goals (total break, partial break or distinguishing break), there are three main categories of attacks:

- *Key-recovery attack*: the attacker deduces the private key. In case the structure of the code for the private key is revealed, the attack is known as a *structural attack*;
- *Message-recovery attack*: the attacker obtains a part or complete plaintext corresponding to a ciphertext without knowing the private key. Such an attack is also known as a *generic attack*;
- *Distinguishing attack*: the attacker can distinguish the cipher from a random message without knowledge about the private key.

We consider several of the known attacks towards the McEliece encryption scheme. For each attack, we will evaluate *the probability of success* or as an inverse problem - we will evaluate the average number of attempts of the attack until the adversary achieves its target.

For algorithmic attacks, *the security level* of a system is defined as a minimum work factor. A *work factor* is the average number of elementary (binary) operations needed to perform a successful attack [3, p.72].

Next, we describe the main attacks published in the relevant literature, assuming that a McEliece cryptosystem is defined by a private key (G, P, S), where G is a generator $k \times n$ matrix of a binary $[n, k, 2t + 1]$ code, P is a random $n \times n$ permutation matrix, S is a random dense $k \times k$ non-singular matrix, and a public key (G', t) where $G' = SGP$. Further, we assume that the attacker has access to a ciphertext c produced by the encryption scheme. Thus, we start by first recalling the components over which brute-force attacks can be mounted. Then, we describe the basic ISD attack and its work factor, along with some of its improved versions, particularly Stern's ISD attack. The last one we use for software implementation and experiments.

Brute-Force Attacks. A brute-force attack is achievable towards different components of the encryption system:

- *Towards the message*: the attacker takes a random message m_1 of length k, encrypts it to

$c_1 = m_1 \cdot G'$ and compute the difference $e_1 = c - c_1$. If the difference e_1 has weight $\leq t$, then the plaintext corresponding to the ciphertext c is exactly m_1, and the attack succeeds. Then the probability of success is $1/2^k$ since the number of all possible messages of length k is 2^k. Note that this is the decoding problem 2 defined in Section 1.5.

– *Towards the coset leaders of the code generated by* G': the attacker computes the syndrome of all coset leaders. The coset leader with syndrome equal to the syndrome of the ciphertext c is the error vector. Knowing the error vector, one can compute the codeword and then the message. The number of the coset leaders is $|\mathbb{F}_2^n|/|C'| = 2^{n-k}$. Therefore the work factor of this attack is at least 2^{n-k}.

– *Towards the error-vector*: the attacker searches among the vectors e of length n and weight t such that the syndrome of e is equal to the syndrome of the received vector c (the ciphertext). Thus, a search on e such that $S(e) = e \cdot H^\top$ equals $S(c)$, where H represents the parity-check matrix corresponding to G'. This problem is equivalent to the problem of finding a linear combination of t columns of H, which results in a column vector with weight $S(c)$. Since there are $\binom{n}{t}$ possible choices for the vector e, then the work factor of the brute force attack towards the error vector is $\binom{n}{t}$.

Basis Information Set Decoding Attack (ISD). As it was mentioned in Section 1.3, the ISD technique was proposed by Prange back in 1962 [54] as an efficient decoding method for cyclic codes. Several works (e.g., [35, 48, 29]) considered increasingly improved versions of the ISD decoding algorithm to attack the McEliece cryptosystem. (See Section 1.3 for the decoding algorithm.)

Let $r = mG' + e$, where G' is a generator matrix of an $[n, k, 2t + 1]$ code $\mathcal{C}$ and e is an error vector of weight t. Let A be an information set of k coordinates such that all entries of the error vector indexed by A are 0. Recall that an information set for a code $\mathcal{C}$ is any set of k coordinates such that the corresponding k columns of a generator matrix of $\mathcal{C}$ form a nonsingular matrix.

In summary, the algorithm for the ISD attack works as follows:

1. Choose k out of n indices for the information set. There are $\binom{n}{k}$ possibilities. These k columns of G' are permuted to the first k positions which is $G'P = [A_k \mid A_{n-k}]$, where A_k are the chosen k columns and A_{n-k} the rest of G';
2. Transform the matrix $[A_k \mid A_{n-k}]$ into a systematic form, which takes $\mathcal{O}(k^3)$ operations [40], since it entails solving k linear equations in k unknowns. This is equivalent to transforming $G'P$ into $[I_k | A'_{n-k}] = UG'P$, where U is the transformation matrix;
3. Compute m as $m = r_A U$, where r_A are the k coordinates of r in the positions of the information set A. Then $e = r - mG'$. If $wt(e) = t$, then m is the encrypted message. The possibilities for the error vector e to have 0 coordinates in the information set are k out of $n - t$ coordinates, i.e. $\binom{n-t}{k}$;
4. Estimate how many choices for k out of n columns have rank k of the generator matrices of the family of $[n, k, 2t + 1]_2$ codes. In the original code-based cryptosystem, Goppa codes were used, and for these codes, around 29% of the choices of k columns are invertible.

Therefore, the work factor for the ISD attack is $\frac{k^3\binom{n}{k}}{\beta\binom{n-t}{k}}$, where β is the proportion of the invertible k columns out of n for the generator matrices of the family of $[n, k, 2t + 1]$ codes. Note that β is different for each family of codes.

Stern's ISD Attack. Stern [60] proposed an improvement of the ISD attack, which is based on the following result:

Lemma 3.1. [3, p.76] The $(n, k+1)$ linear code generated by

$$G'' = \begin{pmatrix} G' \\ x \end{pmatrix} = \begin{pmatrix} G' \\ u \cdot G' + \mathrm{e} \end{pmatrix}$$

has only one minimum weight codeword, which coincides with e.

The idea behind the attack is to use the extended code generated by G'' and find the corresponding unique codeword e of weight t. Stern's algorithm is probabilistic, using two input parameters, p and l, together with the parity check matrix of the extended code [60].

Next, we describe the steps of the attack. Later we implement them in a SageMath code that is used for experiments in Section 4.

Let the extended code be an $[n, k]$ code with a parity check $(n-k) \times n$ matrix H, e be the unique codeword of weight t and p and l - small parameters. Stern's algorithm for finding the codeword e is as follows:

1. Select randomly $n-k$ linear independent columns from H. Permute H so that the selected $n-k$ columns are in the first positions and transform the matrix HP in a systematic form;
2. Partition the remaining k columns into two subsets of X and Y where for each column the decision is independent and uniformly to join X or to join Y;
3. Select l rows randomly from the $n-k$ rows of H and denote the set of these l rows by Z;
4. Compute: for every subset A of X, A of p columns, and for every subset B of p columns of Y, compute the sum of the columns in A and the sum of the columns in B only for the rows in Z. The obtained two columns of length l denote by $\pi(A)$ and $\pi(B)$;
5. Compute: for every A and B such that $\pi(A) = \pi(B)$ compute q as the sum of all columns in $A \cup B$. This sum in the row positions of Z has only 0 since $\pi(A) = \pi(B)$. If the weight of q is $t-2p$, then proceed to 6), otherwise if no vector q of weight $t-2p$ can be found, then the attack has failed;
6. Construct two vectors y and z such that:
   - $y_i = 1$ iff $i \in A \cup B$, $wt(y) = 2p$ since $|A| = |B| = p$,
   - $z_j = 1$ iff j is a column number in I_{n-k} such that q has entry 1, and on the same row, the column j in I_{n-k} also has 1.

   The weight of z is exactly the weight of q, i.e. $t-2p$, the weight of y is $2p$. Moreover, the entries of 1 in y and in z are respectively in the last k and in the first $n-k$ positions. Then $x = y + z$ has exactly weight $2p + t - 2p = t$. Is the vector x in the code with a parity check matrix H? The product $x \cdot H$ is summing up the columns on positions z_j and, the p positions A of X with the p positions B of Y, which result in 0 in the rows of Z and in $t-2p$ weight vector in the last $n-k-l$ rows. These $t-2p$ entries with 1 cancel out with the entries of 1 in the z_j columns. Therefore $x \cdot H = 0$, which implies that the vector x is a codeword.

The work factor of one iteration of the attack has the following three parts [60]:

- work factor of the Gaussian elimination: $f_1 = \frac{1}{2}(n-k)^3 + k(n-k)^2$
- work factor of step 4: $f_2 = 2pl\binom{k/2}{p}$
- work factor of step 5: $f_3 = 2p(n-k)\frac{\binom{k/2}{p}^2}{2^l}$

Then the work factor of one iteration is $B = f_1 + f_2 + f_3$.

The total work factor of the attack is $\frac{B}{P_t}$, where P_t is the probability of finding a codeword with weight t in one iteration. In particular, P_t is estimated in [60] as:

$$P_t = \frac{\binom{t}{2p}\binom{n-t}{k-2p}}{\binom{n}{k}} \cdot \frac{\binom{2p}{p}}{4^p} \cdot \frac{\binom{n-k-t+2p}{l}}{\binom{n-k}{l}}. \tag{2.3}$$

Quantum Basic Information Set Decoding Attack. The Quantum Basic Information Set Decoding function (Algorithm 3) is presented in Section 1.4. The number of qubit operations per iteration and the number of iterations required to find a root of the function are given, respectively $O(n^3)$ bit operations and $\binom{n}{k}/0.29\binom{n-t}{k}$ iterations.

Then the work factor for the Basis Quantum Information Set Decoding attack, which is the complete number of qubit operations for finding a solution, is $O(n^3)\sqrt{\binom{n}{k}/0.29\binom{n-t}{k}}$.

Note that the meaning of 0.29 is that, on overage, 29% of the selected matrices G_S are nonsingular when G is a generator matrix of a Goppa code.

We stress the fact that we do not discuss the following attacks:

1. Taking advantage of partially known plaintexts;
2. Taking advantage of known relations between messages;
3. Message resent;
4. Reaction attack;
5. Malleability.

A detailed description of each of them, applied on the McEliece cryptosystem, can be found in, e.g.[14] and [21]. The reason these attacks are excluded is that they all can be avoided by suitable conversions for the original McEliece cryptosystem, as presented in [33]. These conversions are adapted Pointcheval's Generic Conversion [53] and Fujisaki-Okamoto's Generic Conversion [23]. These general transformations can convert any IND-CPA secure encryption scheme into an IND-CCA secure encryption scheme.

4. Key Encapsulation Mechanism

The public key encryption schemes do not have a difficult key distribution, but they are very inefficient when huge amounts of data must be encrypted fast. For fast encryption, private key encryption algorithms are used. In practice, so called *Hybrid Encryption* is implemented, where the public key encryption is applied for establishing the session key, and then, the actual data are encrypted by the private key encryption algorithm. This improves both the efficiency and the bandwidth because the private key schemes have lower ciphertext expansion.

We define the hybrid encryption via the *KEM/DEM approach* where the *key encapsulation mechanism* (KEM) is the public key component of the encryption and the *data encapsulation mechanism* (DEM) is the private key component.

Following [32], the KEM includes three algorithms: Gen, Encaps and Decaps: Gen denotes the key-generation algorithm, and it creates a pair of public and private keys; Encaps is an *encapsulation* algorithm with the only input, a public key, and it outputs a ciphertext along with a session key; Decaps is a decapsulation algorithm that obtains the same session key by using the ciphertext and the private key. Here it is formally defined:

Definition 4.1. [32, p.390] A key encapsulation mechanism is a tuple of probabilistic polynomial-time algorithms (Gen, Encaps, Decaps) such that:

1. The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a pair of keys (pk, sk), where pk is the *public key*, sk is the *private key*, and both have a length of at least n.
2. The encapsulation algorithm Encaps takes as input a public key pk and the security parameter 1^n, and outputs a ciphertext c and a key $k \in \{0,1\}^{l(n)}$ where l is the key length. It is denoted by $(c,\ k) \leftarrow \mathsf{Encaps}_{pk}(1^n)$.
3. The deterministic decapsulation algorithm Decaps takes as input a private key sk and a ciphertext c, and outputs a key k or a failure denoted by $\perp$. Its notation is $k := \mathsf{Decaps}_s\, k(c)$.

It is required that with all but negligible probability over $(sk,\ pk)$ output by $\mathsf{Encaps}_{pk}(1^n)$, if $\mathsf{Encaps}_{pk}(1^n)$ outputs (c, k) then $\mathsf{Decaps}_{sk}(c)$ outputs k.

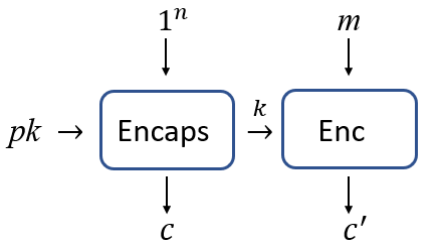

Figure 2.3: Hybrid encryption.

Using a KEM, the hybrid encryption can be expressed as the scheme in Fig 2.3. In it, the sender and the receiver perform the following steps:

- To encrypt a message m to a user with a pair of keys (pk, sk), the sender fulfils the following:
  1. Run the encapsulation algorithm and compute $(c,\ k) \leftarrow \mathsf{Encaps}_{pk}(1^n)$.
  2. Run the encryption algorithm and compute $c' \leftarrow \mathsf{Enc}_k(m)$.
  3. Output the ciphertext (c, c').

- To decrypt the received ciphertext (c, c'), the recipient carries out the steps:
  1. Run the decapsulation algorithm and compute $k \leftarrow \mathsf{Decaps}_{sk}(c)$
  2. If $k = \perp$ output $\perp$.
  3. Run the decryption algorithm and compute $m \leftarrow \mathsf{Dec}_k(c')$.
  3. Output the message m.

In the description above, the sender implements a private-key encryption scheme to encrypt its message m by using the key k. This private key encryption scheme is called a *data-encapsulation mechanism* (DEM).

In [59, Theorem 16.7], it is shown that if KEM is IND-CCA secure and DEM is IND-CCA secure, then the hybrid encryption scheme using these KEM and DEM is also IND-CCA. The definitions for IND-CCA secure encryption scheme is given in Section 3.1.

3

Related work and research questions

1. McEliece type Cryptosystems - existing solutions and security

After the original McEliece encryption scheme [40] was proposed in 1978, there are numerous publications (more than 1890[1]) on modifications of this cryptosystem by implementing different families of codes, and on attacks and security analysis of these systems.

Ten families of codes used in McEliece cryptosystems are listed in [61]. In this section, we extend the list of studied codes with more recent implementations, and we add the current status of the systems and the parameters of the unbroken systems at the time of writing.

Table 3.1: Codes used in McEliece type cryptosystems.
Symbols used for current status *: only specific instances are broken;
†: NIST submission; ‡: NIST finalist.

N	Code	Proposed by	Current status
1	Binary Goppa codes	McEliece, 1978 [40]	Unbroken as of 2021
		Bernstein et al., 2019 [6]	Classic McEliece‡
2	GRS codes	Niederreiter, 1986 [44]	Broken in 1992 [58]
3	MRD codes	Gabidulin, 1991 [25]	Broken in 1995 [27]
		Gabidulin et al., 1995 [24]	Broken in 1996 [28]
4	Reed-Muller codes	Sidelnikov, 1994 [57]	Broken in 2007 [42]
5	QC-BCH subcodes	Gaborit, 2005 [26]	Broken in 2010 [46]
6	QC-LDPC codes	Baldi et al., 2007 [2]	Broken in 2008 [16]
7	Wild McEliece	Bernstein et al., 2010 [9]	Broken* in 2014 [17]

[1] On https://www.semanticscholar.org search on 'McEliece cryptosystem' in ('journal article' or 'conference').

8	Wild McEliece Incognito	Bernstein et al., 2011 [10]	Broken* in 2014 [22]
9	Convolutional codes	Löndahl et al., 2012 [38]	Broken in 2013 [34]
10	QC-MDPC codes	Misoczki et al., 2013 [43] Aragon et al., 2019 [1]	Unbroken as of 2021 BIKE†
11	Random linear codes	Wang, 2016 [63]	Broken* in 2019 [15] RLCE† [62]
12	Rank-Metric codes	Aguilar Melchor et al., 2019 [41]	Reduced security 2020 [4] ROLLO†
13	Specific self-dual codes	Domosi et al., 2019 [18]	Not studied

Table 3.2: Codes used in McEliece type cryptosystem - attacks & system parameters

Code	Type of attack	Code & System parameters
1 Binary Goppa codes	Proven security level against all known attacks	length / public key size / security in bits [6] 3488 / 255KB / 128 4608 / 511.88KB / 196 6688 / 1020.5KB / 256 6960 / 1022.21KB / 256 8192 / 1326KB / 256
2 GRS codes	Structural attack	**Broken**
3 MRD codes	Structural attack	**Broken**
4 Reed-Muller codes	Structural attack	**Broken**
5 QC-BCH subcodes	Structural attack	**Broken**
6 QC-LDPC codes	Structural attack	**Broken**
7 Wild McEliece	Structural attack	**Broken**
8 Wild McEliece Incog.	Structural attack	**Broken**
9 Convolutional codes	Structural attack	**Broken**
10 QC-MDPC codes	Proven security level against all known attacks	length / public key size / security in bits [1] 24646 / 36.21MB / 128 49318 / 144.97MB / 196 81946 / 400.25MB / 256
11 Random linear codes	Structural attack	**Broken** for length 532, 846, 1160 **Unbroken** for length / public key size / security in bits 630 / 188kB / 128 1000 / 450kB / 196 1360 / 1232kB / 256
12 Rank-Metric codes	Generic attack	Unbroken as of 2020 Reduced security from 256 to 200 bits No new parameters after the attack
13 Spec. self-dual codes	Not studied	length / public key size / security in bits 4096 / 512kB / unknown min weight 64

The summary in Table 3.1 shows that most of the implementations are broken. The attacks used in the security analysis are mainly *structural attacks* (see Table 3.2), which succeed in revealing the private key. The common problems in the broken systems are:

1. The McEliece type cryptosystems use codes with too much structure.
2. The public key does not hide well the structure of the private key.

The security of the code-base cryptosystem relies on Theorem 1.3 [5] that the problem of Decoding Random Linear Codes is NP-complete. When the public key is distinguishable from a random code, this is no longer true.

From the summarized results, besides the original cryptosystem based on Goppa codes, there is one more unbroken system, BIKE, based on Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) codes. The other two implementations (11 and 12 in Table 3.1) have some problems. In 11, there are six proposed codes for private keys claimed as random codes, but they have a special structure. In three of these cases, the private key has been retrieved from the public key in polynomial time. Thus, only half of the proposed codes remain for further studies. In 12, the authors of the proposed rank-metric codes have also reported/published an attack that reduces the security level from 256 bits to 200 bits. After this new finding, there is no exact mapping between the parameters of the rank-metric codes and the actual system security level.

Finally, to the best of our knowledge, entry 13 is the only published example of a large self-dual code implemented in a McEliece type cryptosystem. This code has a very small minimum weight and cannot be considered optimal in this sense. Moreover, there is no extensive cryptanalysis and no defined security of the system. The only reason to include it here is that no other example of a self-dual code implementation has been found.

2. Research Questions - detailed description

Having introduced the main terminology and existing results on McEliece type cryptosystems, now we can formulate the main research question and sub-questions in this work.

2.1. Main Question

To what extent can a McEliece type cryptosystem using codes with a high error-correcting capability– derived from self-dual codes– be a secure and practically applicable post-quantum cryptosystem?

2.2. Sub-Questions

The aim of this study is to reduce the key size of the McEliece cryptosystem by using codes with high error-correcting capability while maintaining the security level of the encryption scheme. Applying smaller keys would increase the practical usability of the system. One family of codes with high error-correcting capability is the optimal binary self-dual codes. As there is no study of these codes used in a code-based cryptosystem, their performance in terms of the system security level relative to the key size is unknown. To be able to evaluate such a relation, we first study the security level of a McEliece type cryptosystem, using a known optimal self-dual code of a short length.

RQ1: What is the security level of a McEliece type cryptosystem using a code obtained from an optimal binary [104,56,18] self-dual code? Define the encryption scheme.

The study on this question involves theoretical analysis, programming implementation of the cryptosystem and experiments. After determining the security level of the small example, we focus on a McEliece type cryptosystem providing the currently required bit security level of 80, 128 and 256 bits.

RQ2: Which are the optimal parameters, length and error-correcting capability of a putative self-dual code that, if used in a McEliece type cryptosystem, would provide a bit security level of 80, 128 and 256, respectively?

To create a McEliece type cryptosystem with the recently required 80 bits security, we have to resolve two more challenges: to construct an optimal self-dual code of length of over 1000 and to develop a suitable fast decoding algorithm used in the decryption step. These challenges determine the next two research sub-questions/tasks.

RQ3: Construct an optimal self-dual code with parameters defined in RQ2, which can be used in a McEliece cryptosystem with a security of 80 bits.

In order to define a secure and practically applicable cryptosystem using the code generated in RQ3, we have to determine the algorithms applied in each step of the system.

RQ4: Determine the algorithms in the steps of the McEliece type cryptosystem when using the code obtained in RQ3?

4

McEliece type crytposystem using an optimal self-dual code of length 104

We implement an example of a binary $[104, 52, 18]$ self-dual code in a McEliece type cryptosystem. The code is one of the 18 codes given in [30] that has 23 700 codewords of weight 18. The code is denoted by $\mathcal{C}$[1].

1. Cryptosystem

To define the implementation, we follow the description of the McEliece cryptosystem as given in Section 2.1.

1. *System parameters*:
   (a) $k = 52$ the length of the message m.
   (b) $n = 104$ the length of the ciphertext.
   (c) $t = 8$ the number of the intentionally added errors.
2. *Key generation*: let G be a generator matrix of the $[104, 52, 18]$ self-dual code. Since the public key $G' = SGP$ is expected to be in a systematic form, P is a randomly chosen 104×104 permutation matrix, whereas S is calculated, e.g. from $[GP \mid I_{52}]$ after Gaussian elimination by the following way:
   let GP (52×104) be denoted by $[A_1 \mid A_2]$, where A_1, A_2 are the first and the second 52 columns of matrix GP. Then $[GP \mid I_{52}] = [A_1 \mid A_2 \mid I_{52}]$ after Gaussian elimination is $[I_{52} \mid * \mid S]$, where S is the dense nonsingular matrix in question.
   - Choose a random 104×104 permutation matrix P and compute GP.
     Compute a 52×52 invertible matrix S such that SGP is in a systematic form.
   - Compute $G' = SGP$ and S^{-1} and P^{-1} - the inverse of P and S.
   - *Public key*: (G', t).
   - *Private key*: (G, P, S).
3. *Encryption*: split the data for encryption into k-bit blocks. Then each block m is encrypted as $r = G'm + e$, where e is a random vector of length n and weight t. Stated differently, the message m is encrypted with the public key (G', t) with t errors intentionally introduced by adding the error vector e.

[1] A generator matrix of $\mathcal{C}$ can be found on https://github.com/yorgova/MsThesisData

4. *Decryption:* the decryption steps for the received vector r are:
   (a) Compute $r' = rP^{-1}$, which is $mSG + eP^{-1}$.
   (b) Decode r' into a codeword c' using the decoding Algorithm 5, which will be discussed in Section 2.1. Note that $c' \in \mathcal{C}$ and $c' = mSG$.
   (c) Denote by c the first k bits of c' (since G is in systematic form).
   (d) Return $m = cS^{-1}$.

2. Decoding algorithm and vulnerabilities

2.1. Decoding Algorithm

The decoding algorithm that we apply in the second step of the decryption phase described in Section 1 combines the two algorithms presented in [12] and [37]. From [12], we choose one of the hard-decision deterministic decoding schemes, namely *Algorithm II*, which uses the set of minimum weight codewords of the orthogonal code. This algorithm is generalized in [37] by using any other set of fixed weight dual codewords or a combination of such sets instead of the minimum weight codewords.

First, we define the elements used in the decoding scheme and then the steps of the algorithms. Let $\mathcal{D} \subset \mathbb{F}_2^n$ be an $[n, k, d]$ binary code and $\mathcal{D}^\perp$ be its dual code with minimum distance $d^\perp$. Denote by B the set of all codewords in $\mathcal{D}^\perp$ with weight d_B such that $d_B \geq d^\perp$ (d_B close to $d^\perp$ as in [12]), i.e., $B = \{b \in \mathcal{D}^\perp \mid wt(b) = d_B\}$.

Let $r = c + e$ be the received vector, where $c \in \mathcal{D}$ and $e \in \mathbb{F}_2^n$ is an error vector. Then, for all $b_i \in B$ it follows that the inner product $\langle r, b_i\rangle = \langle c + e, b_i\rangle = \langle c, b_i\rangle + \langle e, b_i\rangle = \langle e, b_i\rangle$. Note that $c \in \mathcal{D}$, $b_i \in B \subset \mathcal{D}^\perp$, hence $\langle c, b_i\rangle = 0$.

Consider

$$WT_B(r) = \sum_{b_i \in B} \langle r, b_i\rangle \tag{4.1}$$

as the sum of all $\langle e, b_i\rangle$, with $b_i \in B$. Stated differently, we count how many codewords in B are not orthogonal to the received vector. Algorithm II in [12] is based on the following observation: given two error vectors e_1 and e_2 with weight $wt(e_1) \leq wt(e_2) \leq \frac{d}{2}$, then $WT_B(e_1) \leq WT_B(e_2)$ is valid in most cases (according to [12]). The steps of this decoding scheme are given in Algorithm 5.

In [37], the considered function is a linear combination of functions as $WT_B(r)$. The dual code $\mathcal{D}^\perp$ is split into sets of codewords with the same weight: B_0, B, ..., B_n for $d_i = 0, 1, \ldots, n$. The counting function is defined as:

$$U(r) = \sum_{d_i=0}^{n} U_{d_i}(r), \text{ where } U_{d_i}(r) = \alpha_{d_i} WT_{B_i}(r), \tag{4.2}$$

and where $\alpha_{d_i} \in \mathbb{R}$, called a *weighted factor*, can be assumed to be only dependent on the weight d_i of the dual codewords in B_{d_i}. The function $U(r)$ is called a *potential function* and U_{d_i} *subpotentials*. According to [37], for efficient decoding, it is not necessary to use all subpotentials in the potential function but only some of them. A decoding example presented in [37] is only using the subpotentials of the maximum and minimum weight vectors in $\mathcal{D}^\perp$.

The decoding schemes that we implement are from Algorithm 5, where instead of $WT_B(r)$, we are using $U(r)$ with only one or two subpotentials and with weighted factors always equal to 1. The number of subpotentials and the value for the factors are determined by experimental evaluation.

Algorithm 5: Hard-decision Decoding using a set of dual codewords

1 Denote $v = r$, r-received vector
Calculate
$X = WT_B(v)$

2 **if** $X = 0$ **then**

3 go to 8)

4 **else**

5 Calculate
$\epsilon_i = WT_B(v + e_i)$ for $i = 1, 2, \ldots, n$,
where $e_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 in the i^{-th} coordinate

6 Find $j \in \{1, 2, \ldots, n\}$ with
$\epsilon_j = min\{\epsilon_i \mid i = 1, 2, \ldots, n\}$

7 $v = v + \epsilon_j$
$X = \epsilon_j$
go to 2)

8 Decode r as the codeword v. Exit.

2.2. Decoding of the [104,52,18] Self-dual Code. Vulnerability and mitigation.

Let B_{18} be the set of all codewords in $\mathcal{C}^{\perp} = \mathcal{C}$ of weight 18. The cardinality of B_{18} is 23 700. Moreover, $rank(B_{18}) = 52$, which means that the set B_{18} spans the entire code.

For decoding, we use Algorithm 5 with potential function $U(r) = U_{18}(r) = WT_{B_{18}}(r)$. A programming implementation is tested on 2 000 random examples of received vectors r, where $r = mG + e$ with m a random message of length 52 and e a random error vector of length 104 and $wt(e) = 8$. All vectors r are correctly decoded.

In the setup, the self-dual $[104, 52, 18]$ code $\mathcal{C}$ is a private key of a McEliece type cryptosystem. Then:

1. The rows of a generator matrix G of $\mathcal{C}$ are orthogonal.
2. The matrix GP, P permutation matrix, generates an equivalent to $\mathcal{C}$ self-dual code.
3. The matrix $G' = SGP$, S being the non-singular matrix, consists of rows which are linear combinations of rows in GP, i.e., SGP generates self-dual code with the same minimum weight as in $\mathcal{C}$.

From the last step, it follows that Algorithm 5 can be applied directly to the public key G' and it will decrypt any ciphertext into a message without any additional knowledge. In order to do it, the set of minimum weight codewords generated by G' are required. This set can be obtained for a self-dual code by computing all linear combinations of $1, 2, \ldots, d/2$ rows in G' and in the parity-check matrix of G' when both matrices are in a systematic form.

An attacker to reveal the structure of the public key only needs to check the self orthogonality of G', and when $k = n/2$, then G' generates a self-dual code. Self orthogonality check includes only computing $k(k-1)/2$ inner products. Generating the set of minimum weight codewords in the public key and in the private key takes the same effort, i.e., the amount of work of the attacker to break the system is equal to the amount of work of the creator of the encryption system.

The number of all linear combinations, generating the set of minimum weight codewords in $\mathcal{C}$,

is:

$$L_{nb} = \sum_{i=1}^{d/2} \binom{k}{i} + \sum_{i=1}^{d/2-1} \binom{k}{i} = \sum_{i=1}^{9} \binom{52}{i} + \sum_{i=1}^{8} \binom{52}{i} \approx 2^{32.36}.$$

We will see later that 32 bits security is much higher than the security level of this system, but this approach breaks the system entirely. Therefore, a McEliece type cryptosystem that uses self-dual codes directly as a private key is vulnerable to a key-recovery attack [2].

To avoid this vulnerability, we consider a $[102, 51, 17]$ punctured code of code $\mathcal{C}$ for the private key, instead of the complete code $\mathcal{C}$. Let matrix G_{short} be obtained from G by removing two columns and one row. Let also $\mathcal{C}_{short}$ be the punctured code of $\mathcal{C}$ generated by G_{short}. The aim is to preserve the error-correcting capability of 8 errors for $\mathcal{C}_{short}$. To achieve this, the set B_{18} must have in the deleted columns only combinations of $[0, 0]$, $[0, 1]$, or $[1, 0]$. The particular set B_{18}, $B_{18} \subset \mathcal{C}$, has 6 column pairs with this property. G_{short} is obtained from G, particularly by removing the first two columns and the first row.

To decode the punctured $[102, 51, 17]$ code $\mathcal{C}_{short}$ with generator matrix G_{short}, we present two strategies: A_1 and A_2. Strategy A_1 is a known procedure to directly decode the punctured code, which is applicable for codes of a small length. The strategy A_2 is new, applicable only when the number of errors is known, and it decodes the punctured code via the complete code. If there exists a fast decoding scheme for the complete code, strategy A_2 is suitable for codes of any length.

A_1 Decoding.

Strategy A_1 performs the decoding via Algorithm 5 with the potential function $U(r) = U_{17}(r) + U_{18}(r)$. The set B_{18} with the first two columns removed is denoted by B_{18_short}. The elements of B_{18_short}, which are orthogonal to $\mathcal{C}_{short}$ and have weight 17 and weight 18, form the sets B'_{17} and B'_{18}, respectively. These two sets are used to calculate the subpotentials as $U_{17}(r) = WT_{B'_{17}}(r)$ and $U_{18}(r) = WT_{B'_{18}}(r)$.

We obtained $|B'_{17}| = 5\,925$, $|B'_{18}| = 11\,850$, $rank(B'_{17}) = 49$, $rank(B'_{18}) = 50$ and together, $rank(B'_{17} \cup B'_{18}) = 51$. Using $B'_{17} \cup B'_{18}$ in the decoding algorithm, we guarantee that each received vector orthogonal to this set will be a codeword of the punctured code.

We tested an implementation of Algorithm 5 with the above-mentioned potential function $U(r)$ on a sample of 2 000 random received vectors r. In this case, $r = mG + e$ with m a random message of length 51 and e a random error vector of length 102 with $wt(e) = 8$. All vectors r are correctly decoded. An experiment shows that using Algorithm 5 only with B_{17} or only with B_{18} instead of both does not always decode. For B_{17}, there are 238 received vectors out of 2 000 which are not decoded, whereas, for B_{18}, there are 9 out of 2 000 also not decoded. It is confirmed that all 347 not decoded vectors are correctly decoded using Algorithm 5 with $B'_{17} \cup B'_{18}$. For an overview, the results of the experiments are included in Table 4.1.

Table 4.1: Performance of Decoding Strategy A_1 on $[102, 51, 17]$ code $\mathcal{C}_{short}$

set	weight i	cardinality	rank	tested r	decoded r	%
B'_{17}	17	5 925	49	2 000	1 762	88.1
B'_{18}	18	11 850	50	2 000	1 991	99.55
$B'_{17} \cup B'_{18}$	17,18	17 775	51	2 000	2 000	100

[2] The private key structure is revealed, and this fact can be used for direct decoding via the public key.

A_2 **Decoding.**

This strategy decodes the punctured code via the complete code. We first define several elements and describe the decoding strategy. Then we present a new decryption algorithm applying this strategy.

Let m be a message of length 51 and $(0 \mid m)$ be m padded with one zero from the left. Denote by P_{short} and S_{short} the permutation and the non-singular matrices used for the public key $G'_{short} = S_{short}G_{short}P_{short}$. The matrix G_{short} is the punctured matrix of G, defined as:

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & g_{1,3} & \cdots & g_{1,104} \\ g_{2,1} & g_{2,2} & & & \\ g_{3,1} & g_{3,2} & & G_{short} & \\ \vdots & \vdots & & & \\ g_{52,1} & g_{52,2} & & & \end{pmatrix}. \tag{4.3}$$

The encoding of $(0 \mid m)$ is:

$$(0 \mid m).G \quad = (m_1^*, m_2^* \mid mG_{short}). \tag{4.4}$$

Thus, the received vector $r' = mG_{short} + e$ of length 102 can be decoded via decoding a padded vector $(*, * \mid r')$ of length 104 by the initial self-dual code $\mathcal{C}$. For $(*, *)$ there are four options, 00, 01, 10, 11, and it is not known which one will decode r'.

The steps of decoding the punctured code via the complete code are:

1. Denote $s = [[0,0],\ [0,1],\ [1,0],\ [1,1]]$, $t = 8$, $k = 51$, $n = 102$.
2. **for** $i = 1$: 4 **do**

   Pad r' into $(** \mid r')$, where $** = s[i]$.

   Decode $(** \mid r')$ into c_1, where $c_1 \in \mathcal{C}$.

   if $c_1[3:104]$ belongs to $\mathcal{C}_{short}$ **then** $//c_1$ without the first 2 coordinates

   Return $c_1[3:104]$. Exit.

3. Return 'Unsuccessful decoding'. Exit.

For the decryption, we define two more elements. Let S and P be extended matrices of S_{short} and P_{short} as follows:

$$S = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & S_{short} & \\ 0 & & & \end{pmatrix} \text{ and } P = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & P_{short} & \\ 0 & 0 & & & \end{pmatrix}. \tag{4.5}$$

One can show that for the matrices defined above the following holds:

$$\begin{aligned}(0 \mid m).S.G.P &= (0 \mid m.S_{short}).G.P = \\ &= (m_1^*, m_2^* \mid m.S_{short}G_{short}P_{short}) = \\ &= (m_1^*, m_2^* \mid mG'_{short}).\end{aligned} \tag{4.6}$$

Thus, we can decode r' of length 102 via decoding a padded $(*, * \mid r')$ of length 104 by the initial self-dual code $\mathcal{C}$.

The decryption, including this decoding strategy, is described in Algorithm 6.

Algorithm 6: Decryption using padded ciphertext.

Denote
$s = [[0,0],[0,1],[1,0],[1,1]]$,
$i = 1$, $t = 8$, $n = 102$, $k = n/2$,

Compute
$r' = rP^{-1}$, r-received vector of length k

while $i < 5$
  Pad
    r' into $(** \mid r')$, where $** = s[i]$
  Decode
    $(** \mid r')$ into c_1, $c_1 \in \mathcal{C}$, by Algorithm 5 with decoding set B_{18}.

  if 5) successful **then**
    Denote
      $c_2 = c_1[3 : n+2]$, $m_2 = c_2[1 : k]$
    Compute
      $m_1 = m_2 * S^{-1}$
    if $(m_1 \in \mathcal{C}'_{short} \quad \wedge \quad \text{weight}(m_1 * G'_{short} - r') == t)$ **then**
      Decrypt r as m_1. Exit.

  $i = i + 1$ increase the index

if $i == 5$ then
  return 'Unsuccessful decryption'. Exit.

An experiment for Algorithm 6 with 2 000 random examples of received vectors r of length 102 shows that all of vectors r are correctly decoded and decrypted.

As decryption is the last step of the McEliece cryptosystem, the scheme is defined. In Figure 4.1 we present the McEliece type encryption scheme using the punctured code $\mathcal{C}_{short}$ of the optimal binary $[104, 52, 18]$ self-dual code $\mathcal{C}$. For the decryption step, they are two choices- using the punctured code or using the complete self-dual code.

1. *System parameters*:
   - $k = 51$ - length of the message m.
   - $n = 102$ - length of the ciphertext r.
   - $t = 8$ - number of the intentionally added errors.
2. *Key generation*:
   - G_{short} - a generator matrix of a $[102, 51, 17]$ code, a shorted code of an optimal $[104, 52, 18]$ code.
   - P - a random $n \times n$ permutation matrix.
   - S - an invertible $k \times k$ matrix such that $SG_{short}P$ is in a systematic form.
   - $G'_{short} = SG_{short}P$, S^{-1} and P^{-1} - the inverse of P and S.
   - *Public key*: $(G'_{short},\ t)$.
   - *Private key*: (G_{short}, G, P, S).
3. *Encryption*:
   - e - a random vector of length n and $wt(e) = t$.
   - $m \to r = G'_{short}m + e$
4. *Decryption:*

1. By private key G_{short}	OR	**2.** By private key G
• $r' = rP^{-1}$ • Decode r' into a codeword c' using *A1 Decoding* - an algorithm described in this sub-section • c - the first k bits of c' • $m = cS^{-1}$		Decrypt by Algorithm 6 using padded ciphertext

Figure 4.1: McEliece type encryption scheme using a $[102, 51, 17]$ punctured code from an optimal $[104, 52, 18]$ self-dual code

3. Cryptanalysis

The attacks described in Section 3.2 are considered against the punctured code of the $[104, 52, 18]$ self-dual code and against the Goppa codes, which would provide a bit security level of the McEliece cryptosystem close to the bit security level provided by the first code. The chosen Goppa codes are small, with a length of $n = 2^m$, $m = 6, 7$ and a number of errors from 4 to 10. The choice of parameters for Goppa codes is also restricted by the information rate $R > 0.4$ $(R = k/n)$ since the code has to be efficient, i.e., $n - k$ check bits do not exceed much the k information bits.

The total cost for each attack is defined in Section 3.2. In Table 4.3, we list the values of log_2 of the total cost for each of the attacks. The notations of the attacks in Table 4.3 are defined in Table 4.2. The value of the parameter β included in the work factor of the attacks A_4 and A_6 is evaluated by experiment, and it equals $29, 05\%$. Figure 4.2 presents the result of the experiment.

Table 4.2: Attacks

Name	Attack
A_1	Brute force attack towards the message
A_2	Brute force attack towards the coset leaders of the private key
A_3	Brute force attack on the error-vector
A_4	Basis Information Set Decoding attack
A_5	Stern's attack
A_6	Basis Quantum Information Set Decoding attack

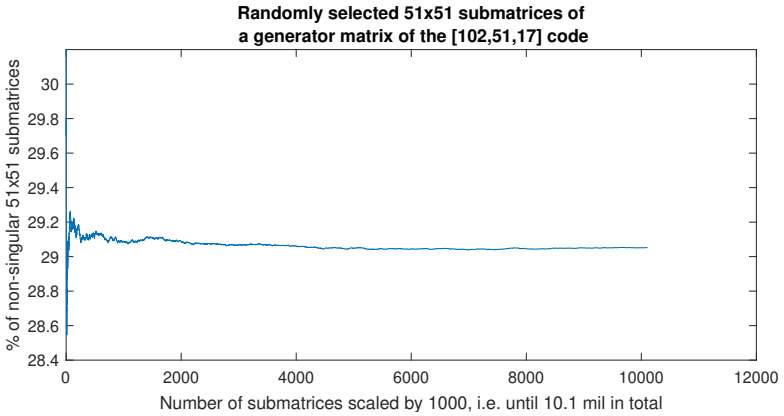

Figure 4.2: Evaluation of the parameter β

Table 4.3: $log_2(Work\ factor)$ of different attacks.
$A = \{A_1, ..., A_5\}$

Goppa codes											
code	n	k	t	k(n-k)	A_1	A_2	A_3	A_4	A_5	A_6	$min(A)$
C_1	128	100	4	2 800	100	28	23.3468	30.7427	20.2171	25.3371	20.5533
C_2	128	93	5	3 255	93	35	27.9791	31.074	21.1199	25.3457	21.1199
C_3	128	86	6	3 612	86	42	32.3366	31.0785	21.9873	25.1787	21.9873
C_4	128	79	7	3 871	79	49	36.46	30.801	22.6618	24.8562	22.6618
C_5	128	72	8	4 032	72	56	40.3789	30.2708	23.19	24.3903	23.3368
C_6	128	65	9	4 095	65	63	44.1158	29.5066	23.5314	23.7869	23.5629
C_7	128	58	10	4 060	58	70	47.6887	28.5193	23.7787	23.0466	23.7787
C_8	128	51	11	3 927	51	77	51.1119	27.3117	23.8866	22.1645	23.8866
C_9	64	52	2	624	52	12	10.9773	23.8201	14.7128	20.4607	10.9773
C_{10}	64	46	3	828	46	18	15.3465	24.0306	16.7361	20.3007	15.3465
C_{11}	64	40	4	960	40	24	19.2773	23.6536	17.9063	19.8097	17.9063
C_{12}	64	34	5	1 020	34	30	22.8622	22.7898	18.5835	19.0261	18.5835
C_{13}	64	28	6	1 008	28	36	26.1599	21.4744	18.9469	17.9482	18.9469
The self-dual code with a punctured code derived from it											
$C_{104,52}$	104	52	8	2 704	52	52	37.9062	27.3062	22.3401	22.2038	22.3401
$C_{102,51}$	102	51	8	2 601	51	51	37.6741	27.2311	22.253	22.1242	22.253

As discussed in the previous section, using a self-dual code for a private key in a McEliece type cryptosystem is not secure. Instead, a punctured code is considered. The values in Table 4.3 show that the classical bit security of the $[102, 51, 17]$ code $C_{102,51}$ is 22.25 bits, and the Goppa

codes with the closest security level are C_3 and C_4. For the quantum security level, our code example is closest to code C_8. Comparing the size of $C_{102,51}$ with the sizes of all three Goppa codes, C_3, C_4, and C_8, one can show that the size of $C_{102,51}$ is at least 28% smaller than the sizes of C_3, C_4, and C_8.

As it was mentioned in Section 3.2, we have implemented Stern's ISD attack for the code $C_{102,51}$ into a SageMath code. We have performed experiments to evaluate the parameters p and l, namely, finding the values for p and l when the attack succeeds for the shortest time. Figure 4.3 shows the outcome of the experiment. The values for $(p,\ l)$ of the fastest Stern's attack are

p	1	1	1	1	1	1	1
l	9	5	7	6	8	4	3
$log_2(time)$	-0.2413	-0.1856	-0.1458	-0.0062	0.0223	0.0968	0.6293

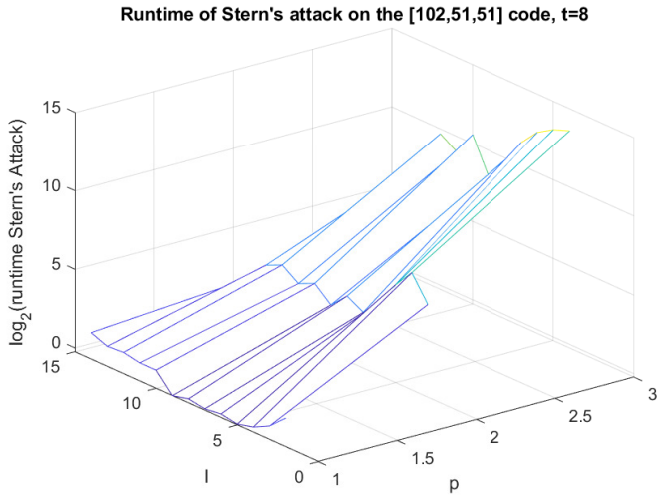

Figure 4.3: Time study of Stern's attack for different (p, l)

Values for $(p,\ l)$ can also be obtained from the work factor of Stern's attack derived from Eq. (2.3). The results are displayed in Figure 4.4. The values for $(p,\ l)$ for the smallest work factor are:

p	1	2	1	2	1	1	1
l	1	9	2	10	3	4	5
$log_2(WF)$	22.25	22.29	22.33	22.33	22.46	22.63	22.81

Note that the smallest values for the $log_2(WF)$ differ by very little, and they all are less than 23. Thus, combining the results from the runtime experiment, we can conclude that a good choice for the parameters (p, l) is $(1, 5)$, $(1, 4)$ and $(1, 3)$.

Figure 4.4: Work factor of Stern's attack with different (p, l)

Remark 1. Structural attacks are not considered because both, the public and the private keys, do not have any specific structure. In order to reconstruct the private key to the initial self-dual code, $2k+(n+2)$ bits have to be restored, which has a much higher work factor than the claimed bit security level requires.

Conclusion RQ1: The security level of the McEliece encryption scheme using a punctured code of an optimal [104,52,18] self-dual code is 22. The system is defined with two choices for the decryption step.

5

Parameters Estimation for Self-dual Codes with Bit Security 80, 128, and 256

To estimate parameters for the self-dual codes, which would provide a security level of 80, 128, and 256 bits, we apply the upper bounds for the work factor of the attacks in the previous section to the known recently proposed Goppa codes with these security levels. Since our attacks are not the best known, we expect to obtain higher values for the upper bounds. We use these higher values further for the estimation of the parameters of the self-dual codes.

The private key of the original McEliece cryptosystem is a $[1\,024, 525]$ Goppa code with the error-correcting capability of 50 errors. It is initially estimated to provide security of 64 bits. Latter, via an improved version of Stern's attack in [8], the security of the system is reduced to 60.5 bits. In the same publication [8], the authors proposed parameters for the Goppa codes, where implementation in the McEliece cryptosystem would provide a security level of 80, 128, and 256 bits. The proposed codes are listed in Table 5.1. The latest proposed codes providing security levels of 128, 196, and 256 bits are in the NIST proposal [6].

From the results listed in Table 5.1, it follows that we have to search for codes providing a bit security level of 83, 148, and 302 to ensure that they would provide at least 80, 128, and 256 bits security against the latest attacks. In Table 5.2, we list the parameters of a few such codes. A larger list is included in Table A.1 in Appendix 2.

Note that these are the parameters of the punctured $[n, k, 2t + 1]$ codes. The corresponding self-dual codes have to be with length $n + 2$ and minimum weight $2t + 3$ to ensure that the

Table 5.1: $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_6$ in Section 3.

Goppa codes							
code	security	n	k	t	k(n-k)	$min_(A_1, ..., A_5)$	A_6
D_1	80 [8]	1 632	1 269	34	460 647	82.231	69.5887
D_2	128 [8]	2 960	2 288	57	1 537 536	129.8371	96.7078
D_3	128 [6]	3 488	2 720	64	2 088 960	147.4275	106.5127
D_4	256 [8]	6 624	5 129	117	7 667 855	259.2255	166.1179
D_5	256 [6]	6 688	5 024	128	8 359 936	265.2662	168.9545
D_6	256 [6]	6 960	5 413	119	8 373 911	266.0612	169.8205
D_7	256 [6]	8 192	6 528	128	10 862 592	302.1663	188.9797

Table 5.2: $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_6$ in Section 3.

Punctured codes						
code	n	k	t	k(n-k)	$min_(A_1, ..., A_5)$	A_6
B_1	1 062	531	75	281 961	87.3248	67.5796
B_2	1 064	532	75	283 024	87.3264	67.5837
B_8	1 076	538	75	289 444	87.2886	67.6079
B_9	1 894	947	134	896 809	147.8721	101.2093
B_{10}	1 896	948	134	898 704	147.869	101.2097
B_{30}	1 940	970	136	940 900	149.8767	102.3316
B_{31}	4 006	2 003	284	4 012 009	303.9682	183.5916
B_{32}	4 008	2 004	284	4 016 016	303.9619	183.5895
B_{42}	4 028	2 014	284	4 056 196	303.8758	183.5694

punctured codes are within the required parameters. The estimation for the self-dual codes is for the minimum weight with 15% less than the upper bounds for the minimum weight of a putative binary self-dual code (see Eq. (2.2)).

This restriction increases the probability that such a code, if it exists, is not unique and could be constructed. The existence of a large number of codes of the same family is a preliminary requirement for the security of the McEliece type cryptosystem.

The size of the putative punctured codes B_1, B_9, and B_{31} is at least 38% smaller than the size of the proposed smallest Goppa codes D_1, D_2, and D_4, providing the security level of 80, 128, and 256 bits, correspondingly. In the next chapter, we will present a possible construction of a self-dual code where the punctured code has the parameters of B_1.

Conclusion RQ2: The optimal parameters of a putative self-dual code as a source for a punctured code used in a McEliece type encryption scheme with a bit security level of 80, 128 and 256, respectively, are:

Self-dual codes			Punctured codes
n	k	d	bit security
1 064	532	154	80
1 066	533	154	80
1 078	539	154	80
1 896	948	272	128
1 898	949	272	128
4 008	2 004	572	256
4 0010	2 005	572	256

6

McEliece Cryptosystem with 80 bits security

To define a McEliece type cryptosystem with 80 bits security, we first construct an example of a binary $[1\,064, 532, d \geq 162]$ self-dual code[1]. To the best of our knowledge, it is the first code of its type and parameters. A punctured code of it is used for a private key of the encryption scheme. As next, an efficient decoding algorithm suitable for the new self-dual code is developed. It is used in the decryption step of the cryptosystem. In addition, we prove that this new decoding scheme is applicable for a large family of codes. Further, we propose a new decryption algorithm for the McEliece type system with a private key a punctured code of the newly constructed self-dual code. The decryption integrates the decoding of the complete self-dual code. Finally, we discuss the bit security level of the McEliece type cryptosystem thus defined.

1. Constructing a binary [1 064, 532, d $\geq$ 162] code

To construct a binary $[1\,064, 532, d \geq 162]$ self-dual code, we use a known algorithm presented in [19] and [67]. We introduce first the main steps of this algorithm and then the construction of the code in question.

1.1. Algorithm for constructing Binary [n, n/2, d] self-dual codes having an automorphism of a particular type.

In Section 1.1, we mentioned when two linear codes in $\mathbb{F}_2^n$ are *permutation equivalent*. In the binary case, two linear codes $\mathcal{C}$ and $\mathcal{C}'$ are *equivalent* if one can be obtained from the other by a permutation of coordinates. That is, if there exists a permutation σ, $\sigma \in S_n$ (S_n- the symmetric group of degree n), such that $\sigma(\mathcal{C}) = \mathcal{C}'$. If $\sigma(\mathcal{C}) = \mathcal{C}$ for some $\sigma \in S_n$, then the permutation σ is an *automorphism* of the code $\mathcal{C}$.

Following the notations in [67], a permutation of order $\mathcal{L}$, having f fixed points and t_1 cycles of length a_1, t_2 cycles of length a_2, ..., t_h cycles of length a_h, with $1 < a_1 < a_2 < \cdots < a_h$, is called a permutation of type $\mathcal{L}$-$(t_1, t_2, \ldots, t_h;\ f)$.

In this chapter, we only consider permutations of type $\mathcal{L}$-$(t_1;\ 0)$, where $\mathcal{L} = pr$, for p and r being odd primes. That is, σ is of order pr and has only cycles of length pr and no fixed points. Thus, without loss of generality, σ can be represented as:

[1] A generator matrix of this example code can be found on https://github.com/yorgova/MsThesisData

$$\sigma = \quad \Omega_1\Omega_2\ldots\Omega_{t_1}, \tag{6.1}$$

where Ω_s is a cycle of length pr for $1 \leq s \leq t_1$.

Let $\mathcal{C}$ be a binary $[n, n/2, d]$ self-dual code with a generator matrix G and σ be an automorphism of $\mathcal{C}$. If $v \in \mathcal{C}$, then v can be presented as

$$v = (v|\Omega_1, v|\Omega_2, \ldots, v|\Omega_{t_1}),$$

where $v|\Omega_i = (v_0, v_1, \ldots, v_{pr-1})$ denotes the coordinates of v in the i^{-th} cycle of σ. Then, the image of v, $\sigma(v)$, is a vector obtained from v by a cyclic shift in each of $v|\Omega_i$. From another side, $\sigma(v) \in \mathcal{C}$ since σ is an automorphism of $\mathcal{C}$. Thus, all vectors obtained from v by cyclic shifts of the coordinates in each Ω_i are also codewords. Therefore, a matrix in the form

$$\begin{pmatrix} G_1 & G_2 & \ldots & G_{t_1} \end{pmatrix}, \tag{6.2}$$

where G_j is a circulant matrix of length pr, generates codewords, and then it can be considered as a sub-matrix of G.

Further, we follow the notations in [67] adjusted to out particular type of automorphism (Eq. (6.1)). The sets $F_\sigma(\mathcal{C})$ and $E_\sigma(\mathcal{C})$ are defined as:

$$F_\sigma(\mathcal{C}) = \{v \in \mathcal{C} | \ \ \sigma(v) = v\} \tag{6.3}$$

and

$$E_\sigma(\mathcal{C}) = \{v \in \mathcal{C} | \ \ wt(v|\Omega_i) \equiv 0 \ (mod\ 2), \ 1 \leq i \leq t_1\}, \tag{6.4}$$

where $v|\Omega_i$ is the restriction of v on Ω_i. Stated differently, $F_\sigma(\mathcal{C})$ in the part of the code which remains fixed under the automorphism σ, whereas $E_\sigma(\mathcal{C})$ contains all the codewords with even weight in each cycle of σ. It is known that both, $F_\sigma(\mathcal{C})$ and $E_\sigma(\mathcal{C})$, are linear subcodes of $\mathcal{C}$ [19]. Moreover, a codeword is fixed under σ if it is a constant in each cycle of σ. This last property implies a map that projects the equal coordinates in each cycle into only one of them. The map is denoted by π, and for our particular type of automorphism is defined as:

$$\pi : \ F_\sigma(\mathcal{C}) \to \mathbb{F}_2^{t_1}, \qquad \pi(v|\Omega_i) = v_j, \tag{6.5}$$

for some $j \in \Omega_i$, $i = 1, \ldots, t_1$.

Proposition 1.1. [67] If $\mathcal{C}$ is a binary self-dual code having an automorphism of type pr-$(t_1;\ 0)$, then $\pi(F_\sigma(\mathcal{C}))$ is a binary self-dual code of length t_1.

Proposition 1.2. [67] A self-dual code $\mathcal{C}$, as described in Proposition 1.1, can be decomposed to $\mathcal{C} = F_\sigma(\mathcal{C}) \oplus E_\sigma(\mathcal{C})$, where $\oplus$ stands for the direct sum of linear subspaces, and $\dim E_\sigma(\mathcal{C}) = \frac{t_1}{2}(pr - 1)$.

Thus, a generator matrix of $\mathcal{C}$ can be decomposed as:

$$G = \quad \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{6.6}$$

where X is a generator matrices of $F_\sigma(\mathcal{C})$ and Y is a generator matrix of $E_\sigma(\mathcal{C})$.

Let $\mathcal{P}$ denote the set of even-weight polynomials in $\mathcal{R} = \mathbb{F}_2[x]/(x^{pr} - 1)$ and map φ be the following:

$$\varphi : \ E_\sigma(\mathcal{C}) \quad \to \quad \mathcal{P}^{t_1}, \tag{6.7}$$

where $v|\Omega_i = (v_0, v_1, \ldots, v_{pr-1})$ is identified with the polynomial $\varphi(v|\Omega_i)(x) = v_0 + v_1 x + \cdots + v_{pr-1}x^{pr-1}$ in $\mathcal{P}$ for $1 \le i \le t_1$.

An *inner product* in $\mathcal{P}^{t_1}$ is defined as:

$$\langle g, h\rangle = g_1(x)h_1(x^{-1}) + \cdots + g_{t_1}(x)h_{t_1}(x^{-1}) \tag{6.8}$$

for every $g, h \in \mathcal{P}^{t_1}$.

Lemma 1.1. [67] If $\mathcal{C}$ is a binary self-dual code having an automorphism σ defined in Eq. (6.1), then $\varphi(E_\sigma(\mathcal{C}))$ is a self-orthogonal code, that is:

$$u_1(x)v_1(x^{-1}) + \cdots + u_{t_1}(x)v_{t_1}(x^{-1}) = 0, \tag{6.9}$$

for $\forall u, v \in \varphi(E_\sigma(\mathcal{C}))$.

The method from [19] and [67] uses the presented above properties of the subcodes $F_\phi(\mathcal{C})$ and $E_\sigma(\mathcal{C})$ and their images, $\pi(F_\sigma(\mathcal{C}))$ and $\varphi(E_\sigma(\mathcal{C}))$, to construct the self-dual code $\mathcal{C}$. The method includes the following steps (Algorithm 7):

Algorithm 7: Construction of a self-dual code having an automorphism.

1 Determine a generator matrix X' of $\pi(F_\sigma(\mathcal{C}))$.

2 Find the generator matrix X of $F_\sigma(\mathcal{C})$ corresponding to X'.

3 Construct a generator matrix Y' of $\varphi(E_\sigma(\mathcal{C}))$.

4 Find the generator matrix Y of $E_\sigma(\mathcal{C})$ corresponding to Y'.

5 **if** $G = \begin{pmatrix} X \\ Y \end{pmatrix}$ generates a code with a minimum weight d, **then**

6 return G (G generates $\mathcal{C}$); Exit.

7 **else**

8 return to 1.

1.2. A Binary [1 064, 532, d ≥ 162] Self-dual Code.

Let B be a self-dual $[1\,064, 532, d \ge 162]$ code having an automorphism σ_1 of order 133 with 8 cycles of length 133 and no fixed points, i.e. σ_1 of type $133 - (8; 0)$. Without loss of generality σ_1 can be represented as:

$$\sigma_1 = \Omega_1\Omega_2\ldots\Omega_8,$$

where Ω_i is a cycle of length 133 for $1 \le i \le 8$.

The sets $F_{\sigma_1}(B)$, $E_{\sigma_1}(B)$, and the images π and φ are defined as in Section 1.1. For the particular code B and permutation σ_1 they are:

- $F_{\sigma_1}(B) = \{v \in B |\ \ v\sigma_1 = v\}$;
- $E_{\sigma_1}(B) = \{v \in B |\ \ \mathrm{wt}(v|\Omega_i) \equiv 0 \,(\mathrm{mod}\, 2),\ \ i = 1, \ldots, 8\}$;
- $\pi\colon F_{\sigma_1}(B) \to \mathbb{F}_2^8, \qquad \pi(v|\Omega_i) = v_j$ for some $j \in \Omega_i, i = 1, \ldots, 8$;
- $\varphi:\ E_{\sigma_1}(B) \ \ \to \ \ \mathcal{P}^8, \qquad \varphi(v|\Omega_i)(x) = v_0 + v_1 x + \cdots + v_{132}x^{132}$ in $\mathcal{P}$ for $1 \le i \le 8$,

where $v|\Omega_i = (v_0, v_1, \ldots, v_{132})$ and $\mathcal{P}$ is the set of even-weight polynomials in $\mathcal{R}_1 = \mathbb{F}_2[x]/(x^{133} - 1)$.

An inner product in $\mathcal{P}^8$ is defined as in Eq. 6.8 for $t_1 = 8$:

$$\langle g, h \rangle = g_1(x)h_1(x^{-1}) + \cdots + g_8(x)h_8(x^{-1}) \tag{6.10}$$

for all $g, h \in \mathcal{P}^8$.

According to Section 1.1, for code B, the following holds:
1. $B = F_{\sigma_1}(B) \oplus E_{\sigma_1}(B)$.
2. The image $\pi(F_{\sigma_1}(B))$ of the fixed subcode $F_{\sigma_1}(B)$ is a binary $[8, 4]$ self-dual code.
3. The image $\varphi(E_{\sigma_1}(B))$ is a self-orthogonal code, that is $\langle g, h \rangle = 0$ for any $g, h \in \varphi(E_{\sigma_1}(B))$, where $\langle g, h \rangle$ is defined in Eq. (6.10).

To construct the code B we follow the steps of Algorithm 7.
1. Determine a generator matrix of $\pi(F_{\sigma_1}(B))$.
   The image $\pi(F_{\sigma_1}(B))$ is a binary $[8, 4]$ self-dual code. Then a possible generator matrix of $\pi(F_{\sigma_1}(B))$ is:
$$X' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$
2. Find the corresponding generator matrix X of $F_{\sigma_1}(B)$.
   The corresponding to X' generator matrix of $F_{\sigma_1}(B)$ is:
$$X = \begin{pmatrix} s & o & o & o & o & s & s & s \\ o & s & o & o & s & o & s & s \\ o & o & s & o & s & s & o & s \\ o & o & o & s & s & s & s & o \end{pmatrix}, \tag{6.11}$$
   where $s = (1, 1, \ldots, 1)$ is the all ones vector and o is the zero vector in $\mathbb{F}_2^{133}$.
3. Construct a generator matrix Y' of $\varphi(E_{\sigma_1}(B))$.
   $\varphi(E_{\sigma_1}(B)) \subset \mathcal{P}^8$, where $\mathcal{P}$ is the set of even-weight polynomials in $\mathcal{R}_1 = \mathbb{F}_2[x]/(x^{133} - 1)$.

   The factorization of the polynomial $x^{133} - 1$ over $\mathbb{F}_2$ is
$$x^{133} - 1 = h_0(x)h_1(x) \ldots h_9(x),$$
   where $h_0 = x - 1$, $deg(h_1(x)) = deg(h_2(x)) = 3$ and $deg(h_j(x)) = 18$ for $j = 3, \ldots, 9$. Denote $g_j(x) = \frac{x^{133}-1}{h_j(x)}$, $I_j = \langle g_j(x) \rangle$ - the ideal of $\mathcal{R}_1$ generated by $g_j(x)$ and, $e_j(x)$ - the generator idempotent of I_j for $j = 0, \ldots, 9$.

   According to Theorem 1.9:
   - $\mathcal{R}_1 = I_0 \oplus I_1 \oplus \cdots \oplus I_s$;
   - I_j is a field with $2^{deg(h_j(x))}$ elements, $j = 0, 1, \ldots, 9$;
   - $e_i(x)e_j(x) = 0$, $i \neq j$.

   Using Theorem 1.7 we calculated the general idempotent $e_j(x)$ of the ideal I_j, for $j = 1, \ldots, 9$. We observe that $e_1(x^{-1}) = e_2(x)$, $e_3(x^{-1}) = e_4(x)$, $e_5(x^{-1}) = e_6(x)$, $e_7(x^{-1}) = e_8(x)$ and $e_9(x^{-1}) = e_9(x)$. The same relations are also valid between the generator polynomials

$g_i(x)$, $1 \le i \le 9$, i.e. $g_1(x^{-1}) = g_2(x)$, $g_3(x^{-1}) = g_4(x)$, etc. Using these relations and the self-orthogonality of the image $\varphi(E_{\sigma_1}(B))$, we construct a generator matrix of $\varphi(E_{\sigma_1}(B))$ in the form:

$$Y' = \begin{pmatrix} Y_1 \\ \vdots \\ Y_9 \end{pmatrix}, \tag{6.12}$$

where Y_j is 4×8 matrix with elements of I_j, for $j = 1, \ldots, 9$. The cells Y_1, Y_3, Y_5 and Y_7 are constructed under certain conditions, which we discuss at the end of this section. The cells Y_2, Y_4, Y_6 and Y_8 are obtained from the previous four cells using the orthogonality condition Eq.(6.10). For example, if Y_1 is the following matrix:

$$Y_1 = \begin{pmatrix} e_1(x) & 0 & 0 & 0 & 0 & g_1(x) & g_1(x) & g_1(x) \\ 0 & e_1(x) & 0 & 0 & g_1(x) & 0 & g_1^2(x) & g_1^3(x) \\ 0 & 0 & e_1(x) & 0 & g_1(x) & g_1^2(x) & 0 & g_1^2(x) \\ 0 & 0 & 0 & e_1(x) & 0 & g_1^2(x) & g_1^3(x) & g_1^4(x) \end{pmatrix},$$

then the corresponding Y_2 is

$$Y_2 = \begin{pmatrix} 0 & g_2(x) & g_2(x) & 0 & e_2(x) & 0 & 0 & 0 \\ g_2(x) & 0 & g_2^2(x) & g_2^2(x) & 0 & e_2(x) & 0 & 0 \\ g_2(x) & g_2^2(x) & 0 & g_2^3(x) & 0 & 0 & e_2(x) & 0 \\ g_2(x) & g_2^3(x) & g_2^2(x) & g_2^4(x) & 0 & 0 & 0 & e_2(x) \end{pmatrix}$$

As $e_1(x^{-1}) = e_2(x)$, $g_1(x^{-1}) = g_2(x)$, $e_i(x)e_j(x) = 0$ and $g_i(x)g_j(x) = 0$ for $i \ne j$, it follows that the orthogonality condition Eq.(6.10) holds for any two rows in Y_1, any two rows in Y_2, and any row of Y_1 or Y_2 with any row of Y_j, $j = 3, \ldots, 9$. For example, the first row of Y_1 is orthogonal to itself since:

$$\begin{aligned} &e_1(x)e_1(x^{-1}) + 0 + 0 + 0 + 0 + g_1(x)g_1(x^{-1}) + g_1(x)g_1(x^{-1}) + g_1(x)g_1(x^{-1}) = \\ &= e_1(x)e_2(x) + g_1(x)g_2(x) + g_1(x)g_2(x) + g_1(x)g_2(x) = 0. \end{aligned}$$

Y_2 is obtained from Y_1 by the orthogonality condition, thus Eq.(6.10) also holds for any row of Y_1 with any row of Y_2. For example, the Eq.(6.10) for the last row of Y_1 and the second row of Y_2 is:

$$0\ g_2(x^{-1}) + 0 + 0\ g_2^2(x^{-1}) + e_1(x)g_2^2(x^{-1}) + 0 + g_1^2(x)e_2(x^{-1}) + g_1^3(x)\ 0 + g_1^4(x)\ 0 = 0$$

or equivalently:

$$e_1(x)g_1^2(x) + g_1^2(x)e_1(x)) = 0$$

The last is satisfied because the operations are in $\mathcal{R}_1 = \mathbb{F}_2[x]/(x^{133} - 1)$.
The particular matrices that are chosen for $Y_3, Y_4, \ldots, Y_8$ are included in Appendix 3, matrix Y'.

Generating a matrix of the cell Y_9 requires a different approach since $e_9(x^{-1}) = e_9(x)$ and $g_9(x^{-1}) = g_9^{512}(x)$. For example, let the first two rows of Y_9 be

$$\begin{aligned} w_1 = (\ & e_9(x), & 0, & & 0, & 0, & a_1, & a_2, & a_3, & a_4\ ) \\ w_2 = (\ & 0, & e_9(x), & & 0, & 0, & a_5, & a_6, & a_7, & a_8\ ), \end{aligned}$$

where a_j is 0 or $g_9^{s_j}$ for $s_j = 0, \ldots, 2^{18} - 1$, $j = 1, \ldots, 8$. The condition $\langle w_1, w_1 \rangle = 0$ is:

$$e_9(x)e_9(x^{-1}) + a_1(x)a_1(x^{-1}) + a_2(x)a_2(x^{-1}) + a_3(x)a_3(x^{-1}) + a_4(x)a_4(x^{-1}) = 0.$$

Using SageMath we obtain that this equation has more than 25 mil. of solutions for (a_1, a_2, a_3, a_4). Combining with $\langle w_1, w_2 \rangle = 0$ and $\langle w_2, w_1 \rangle = 0$ one can verify that for a fixed choice of (a_1, a_2, a_3, a_4) there exist at least 200 solutions for (a_5, a_6, a_7, a_8).
Let the next two rows of Y_9 be:

$$\begin{aligned} w_3 = (0, \quad & 0, & e_9(x), & & 0, & a_9, & a_{10}, & a_{11}, & a_{12}\ ) \\ w_4 = (0, \quad & 0, & 0, & & e_9(x), & a_{13}, & a_{14}, & a_{15}, & a_{16}\ ), \end{aligned}$$

where a_j is 0 or $g_9^{s_j}$ for $s_j = 0, \ldots, 2^{18} - 1$, $j = 9, 10, \ldots, 16$.
There are 5 orthogonality conditions for w_3 and 7 for w_4. For example for w_3 they are: $\langle w_3, w_3 \rangle = 0$, $\langle w_3, w_1 \rangle = 0$, $\langle w_1, w_3 \rangle = 0$, $\langle w_3, w_2 \rangle = 0$ and $\langle w_2, w_3 \rangle = 0$.

Using SageMath, a direct check of them for the possible cases of w_3 could not find a solution for an execution time of 24 hours. Therefore, we have developed a different strategy: consider the third row, w_3, with one variable and the fourth row, w_4, with two variables where we find a solution consequently - first for w_3 and then for w_4. In particular, the strategy includes the following two main steps:

a) Generate by SageMath the following two rows of Y_9:

$$\begin{aligned} v_1 = (\ & e_9(x), & 0, & & 0, & 0, & g_9^{s_1}(x), & g_9^{s_2}(x), & g_9^{s_3}(x), & g_9^{s_4}(x)\ ) \\ v_2 = (\ & 0, & e_9(x), & & 0, & 0, & g_9^{s_5}(x), & g_9^{s_6}(x), & g_9^{s_7}(x), & g_9^{s_8}(x)\ ), \end{aligned}$$

such that $\langle v_1, v_1 \rangle = 0$, $\langle v_1, v_2 \rangle = 0$, $\langle v_2, v_1 \rangle = 0$ and $\langle v_2, v_2 \rangle = 0$, where $s_j = 0, \ldots, 2^{18} - 1$ for $j = 0, 1, \ldots, 8$.

b) Consider the third and fourth rows of Y_9 in the form:

$$\begin{aligned} v_3 = (\ & g_9^{s_1}(x^{-1}), & g_9^{s_5}(x^{-1}), & & g_9^{y_1}(x), & 0, & e_9(x), & 0, & 0, & 0)\ ) \\ v_4 = (\ & g_9^{s_2}(x^{-1}), & g_9^{s_6}(x^{-1}), & & g_9^{y_2}(x), & g_9^{y_3}(x), & 0, & e_9(x), & 0, & 0)\ ), \end{aligned}$$

where s_1, s_2, s_5 and s_6 are from v_1 and v_2, and y_i is unknown, $0 \leq y_i \leq 2^{18} - 1$ for $i = 1, 2, 3$. Then:

(i) find y_1 from $\langle v_3, v_3 \rangle = 0$;
(ii) find y_2 from $\langle v_3, v_4 \rangle = 0$ and $\langle v_4, v_3 \rangle = 0$;
(iii) find y_3 from $\langle v_4, v_4 \rangle = 0$.

Using that $g_9(x^{-1}) = g_9^{512}(x)$ we rewrite v_3 and v_4 as:

$$\begin{aligned} v_3 &= (\ g_9^{512s_1}(x),\ g_9^{512s_5}(x),\ g_9^{y_1}(x),\ 0,\ e_9(x),\ 0,\ 0,\ 0)\ ) \\ v_4 &= (\ g_9^{512s_2}(x),\ g_9^{512s_6}(x),\ g_9^{y_2}(x),\ g_9^{y_3}(x),\ 0,\ e_9(x),\ 0,\ 0)\ ). \end{aligned}$$

Then the orthogonality conditions from b) for finding y_1, y_2 and y_3 acquire the following form:

$$\begin{aligned} &g_9^{512s_1}(x)g_9^{512s_1}(x^{-1}) + g_9^{512s_5}(x)g_9^{512s_5}(x^{-1}) + g_9^{y_1}(x)g_9^{y_1}(x^{-1}) + e_9(x)e_9(x^{-1}) = 0 \\ &g_9^{512s_1}(x)g_9^{512s_2}(x^{-1}) + g_9^{512s_5}(x)g_9^{512s_6}(x^{-1}) + g_9^{y_1}(x)g_9^{y_2}(x^{-1}) = 0 \\ &g_9^{512s_1}(x^{-1})g_9^{512s_2}(x) + g_9^{512s_5}(x^{-1})g_9^{512s_6}(x) + g_9^{y_1}(x^{-1})g_9^{y_2}(x) = 0 \\ &g_9^{512s_2}(x)g_9^{512s_2}(x^{-1}) + g_9^{512s_6}(x)g_9^{512s_6}(x^{-1}) + g_9^{y_2}(x)g_9^{y_2}(x^{-1}) + g_9^{y_3}(x)g_9^{y_3}(x^{-1}) + e_9(x)e_9(x^{-1}) = 0 \end{aligned}$$

or equivalently:

$$\begin{aligned} g_9^{513s_1}(x) + g_9^{513s_5}(x) + e_9(x) &= g_9^{513y_1}(x) \\ g_9^{512s_1+s_2}(x) + g_9^{512s_5+s_6}(x) &= g_9^{y_1+512y_2}(x) \\ g_9^{s_1+512s_2}(x) + g_9^{s_5+512s_6}(x) &= g_9^{512y_1+y_2}(x) \\ g_9^{513s_2}(x) + g_9^{513s_6}(x) + g_9^{513y_2}(x) + e_9(x) &= g_9^{513y_3}(x)\ . \end{aligned} \tag{6.13}$$

In particular, in a) we consider the following solution for $s_1, \ldots, s_8$:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
133	1	319	233 370	266	2	1	49

Solving the equations (6.13) for y_1 , y_2 and y_3 we obtain:

- $y_1 = 117 + 511k_1$, $k_1 = 0, 1, \ldots, 512$, where we choose $k_1 = 2$, $y_1 = 1139$.
- $y_2 = 149579$.
- $y_3 = 338 + 511k_3$, $k_3 = 0, 1, \ldots, 512$, where we choose $k_3 = 0$, $y_3 = 338$.

Hence, we have defined an example for the cell Y_9 of Y' (Eq. (6.12)). The complete matrix Y' of $\varphi(E_{\sigma_1}(B))$ is given in Appendix 3.

Following Algorithm 7, the next step is:

4. Find the corresponding generator matrix Y of $E_{\sigma_1}(B)$.
The matrix Y' defines the generator matrix of the subcode $E_{\sigma_1}(B)$ as

$$Y = \begin{pmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,8} \\ \vdots & \vdots & & \vdots \\ y_{36,1} & y_{36,2} & \cdots & y_{36,8} \end{pmatrix}, \tag{6.14}$$

where each entry of the first 8 rows is a right circulant 3×133 matrix, as I_j is a cyclic $[133, 3]$ code for $j = 1, 2$, and each entry of the rest of the rows is a right circulant 18×133

since I_j is a cyclic $[133, 18]$ code for $j = 3, \ldots, 9$. The first rows of the circulants correspond to the polynomials in the matrix Y' (Eq. (A.1))

In step 3. it was mentioned that the cells Y_1, Y_3, Y_5 and Y_7 are constructed under certain conditions. The matrix Y', $i = 1, \ldots, 9$, specifies the generator matrix Y of the subcode $E_{\sigma_1}(B)$. The minimum weight of the code B has to be greater than or equal to 162, thus, the same has to hold for the minimum weight of Y. In this regard, we construct the Y_i cells according to the following requirements:

- each row of Y_i, $i = 1, \ldots, 9$, has at least four non zero elements, i.e. each row has weight greater than or equal to four;
- the weight of Y_1, $\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$, Y_3, $\begin{pmatrix} Y_3 \\ Y_4 \end{pmatrix}$, Y_5, $\begin{pmatrix} Y_5 \\ Y_6 \end{pmatrix}$, Y_7, $\begin{pmatrix} Y_7 \\ Y_8 \end{pmatrix}$ and Y_9 is at least 3.

In step 4. the matrix Y has to satisfy the following requirements:

- the first 24 rows, corresponding to $\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}$, have a minimum weight of at least 168;
- each next 18 rows corresponding to a row in Y_s, $s = 3, \ldots, 9$, have a minimum weight of at least 168;
- the linear combinations of up to 8 rows of the 144 rows of Y corresponding to $\begin{pmatrix} Y_3 \\ Y_4 \end{pmatrix}$, $\begin{pmatrix} Y_5 \\ Y_6 \end{pmatrix}$, $\begin{pmatrix} Y_7 \\ Y_8 \end{pmatrix}$ and the 72 rows corresponding to Y_9, have a weight of at least 168.

Once we have constructed generator matrices for the subcodes $F_{\sigma_1}(B)$ and $E_{\sigma_1}(B)$, we can proceed to step 5. of Algorithm 7.

5. **if** $G = \begin{pmatrix} X \\ Y \end{pmatrix}$ generates a code with a minimum weight $d \geq 162$, where X and Y are given in Eq. (6.11) and Eq. (6.14), then G generates the code B.

Software implementation: For constructing a generator matrix of B, we developed a software in C^{++} performing the following:

- Constructing sub-matrices of Y corresponding to Y_s, $s = 1, \ldots, 8$, which hold the requirements in steps 3 and 4.
- Creating the sub-matrix of Y corresponding to Y_9 defined in step 3.
- Creating the matrix G defined in Eq.(6.6).
- Creating the parity-check matrix H of G.
- Calculating the weight of all linear combinations up to 8 rows of G and of H. For this we implement the algorithm for efficiently computing the codewords of fixed weight in linear codes (for the binary case) presented in [13].

Due to computation time, the exact minimum weight of the code is not calculated. Instead, all linear combinations of up to 8 vector rows of G and of the corresponding parity check matrix are computed. They all have a weight greater than or equal to 168. A random linear combination of a random number of rows of G on a single 16 RAM Intel7 PC for 30 days did not result in a vector with a weight smaller than 168.

Moreover, note that the requirements for steps 3. and 4. are held during the construction of the cells of Y' and the matrix Y. Therefore, the first $24 \times 1\,064$ and every next $18 \times 1\,064$ sub-matrix of Y has a minimum weight of at least 168.

Based on the listed evidence, we expect the matrix G with the defined above sub-matrices X and Y to generate a self-dual $[1\,064, 532, d \geq 162]$ code.

Conclusion RQ3: The matrix

$$G = \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{6.15}$$

where X and Y are given in Eq. (6.11) and Eq. (6.14), generates a self-dual code of length 1 064. A punctured code of it used in a McEliece type encryption scheme is expected to provide a security level of 80 bits.

2. A new decoding algorithm

The decoding Algorithm 5 applied for the small code example of length 102 is using the set of the minimum weight dual codewords or union of sets with chosen weights from the dual code. For code B_1 to find all the codewords with the minimum weight is a computationally difficult problem. Moreover, the set can be very large, i.e., it requires a large memory, which is a limitation for practical implementation in the current communication systems.

This chapter presents a decoding scheme suitable for self-dual codes having an automorphism of a specific type. We first prove a polynomial orthogonal property of such codes. Then an algorithm for decoding of cyclic codes introduced by Bossert in [11] is described. Next, we combine the basic idea of the cyclic codes decoding with the polynomial orthogonal condition for the specific type of self-dual codes into a new hard decision iterative decoding algorithm for these self-dual codes. Three examples demonstrate the efficiency of this new decoding strategy. At last, we discuss an application of the new algorithm on the binary $[1\,064, 532, d > 160]$ self-dual code constructed in Section 1.2.

2.1. Polynomial orthogonal property of self-dual codes of a specific type.

Consider all elements defined in Section 1.1 by the same names and notations. Let u and v are two codewords of $\mathcal{C}$. In polynomials they can be representes as $u = (u_1(x), u_2(x), \cdots, u_{t_1}(x))$ and $v = (v_1(x), v_2(x), \cdots, v_{t_1}(x))$, where $u_j(x)$, $v_j(x) \in \mathcal{R}$ for $1 \leq j \leq t_1$. An *inner product* of u and v is defined similarly to Eq. (6.8) by:

$$\langle u, v \rangle = u_1(x)v_1(x^{-1}) + \cdots + u_{t_1}(x)v_{t_1}(x^{-1}) \tag{6.16}$$

for every $u, v \in \mathcal{C}$.

According to Lemma 1.1, this inner product is equal to 0 if $u, v \in \varphi(E_\sigma(\mathcal{C}))$. We will prove that this orthogonality holds for every u and v in $\mathcal{C}$, where $\mathcal{C}$ is a self-dual code with an automorphism σ of type $pr - (t_1;\ 0)$, with p, r being odd primes.

Lemma 2.1. Let $\mathcal{C}$ be a binary $[n, n/2, d]$ self-dual code possessing an automorphism σ of type pr-$(t_1;\ 0)$, where p and r are odd primes and $n = prt_1$. Then, Eq. (6.9) holds for every $u,\ v \in \mathcal{C}$.

To prove Lemma 2.1, we first prove that Eq. (6.9) holds for every $u,\ v \in F_\sigma(\mathcal{C})$, then for every $u \in F_\sigma(\mathcal{C})$ and every $v \in E_\sigma(\mathcal{C})$, and at last, the statement in the lemma, for $\forall u, v \in \mathcal{C}$.

Proposition 2.1. Let $\mathcal{C}$ be a binary self-dual code having an automorphism σ of type $pr-(t_1;\ 0)$. Let each $u \in F_\sigma(\mathcal{C})$ be presented as $u = (u_1(x), u_2(x), \ldots, u_{t_1}(x))$. Then, for $u,\ v \in F_\sigma(\mathcal{C})$ it follows:

$$u_1(x)v_1(x^{-1}) + \cdots + u_{t_1}(x)v_{t_1}(x^{-1}) = 0. \tag{6.17}$$

Proof. The codewords of the subcode $F_\sigma(\mathcal{C})$ can be seen as one row circulant matrices in Eq. (6.2). Moreover, the coordinates of a fixed codeword are constant in each pr cycle, i.e., $v|\Omega_i = (0, 0, \ldots, 0)$ or $v|\Omega_i = (1, 1, \ldots, 1)$ for $v \in F_\sigma(\mathcal{C})$, $1 \leq i \leq t_1$.

If a map π is defined as in Eq. (6.5), then $\pi(F_\sigma(\mathcal{C}))$ is a binary $[t_1, t_1/2]$ self-dual code. Therefore, t_1 must be even and the weight of each element in $\pi(F_\sigma(\mathcal{C}))$ must also be even. Thus, each fixed codeword in $\mathcal{C}$ has an even number of $v|\Omega_i = (1, 1, \ldots, 1)$. Hence, written in polynomials in $\mathcal{R}$ it is:

• each $g = (g_1(x), g_2(x), \ldots, g_{t_1}(x)) \in F_\sigma(\mathcal{C})$ has an even number of coordinates $g_s(x)$ of the form $1 + x + x^2 + \cdots + x^{pr-1}$.

If $1 + x + x^2 + \cdots + x^{pr-1}$ is denoted by $g_0(x)$, then it can be obtained that $g_0(x^{-1}) = g_0(x) \mod (x^{pr} - 1)$ and $g_0(x)g_0(x^{-1}) = g_0(x) \mod (x^{pr} - 1)$.

Thus, we can conclude that Eq. (6.17) holds for two codewords in $F_\sigma(\mathcal{C})$ if they intersect in even positive number of cycle positions with full one vector or, they intersect in 0 cycle positions with full one vector.

Let us assume that there exist two codewords $v,\ v' \in F_\sigma(\mathcal{C})$ that intersect in an odd number of cycle positions with a full one vector, and this odd number is z. As the length of the cycles is pr, then the regular inner product of v and v', $v.v' = v_1v'_1 + v_2v'_2 + \cdots + v_{prt_1}v'_{prt_1} \mod (2)$, will be equal to zpr, where z is odd, and p and r are odd primes. Thus, $v.v' = zpr \equiv 1 \mod(2)$, which is a contradiction to $\mathcal{C}$ being a self-dual code. Therefore, Eq. (6.9) holds for any two codewords in $F_\sigma(\mathcal{C})$. □

Let now $u \in F_\sigma(\mathcal{C})$ and $v \in E_\sigma(\mathcal{C})$, where $u = (u_1(x), u_2(x), \ldots, u_{t_1}(x))$ with coordinates $u_i(x) = 0$ or $u_i(x) = 1 + x + x^2 + \cdots + x^{pr-1}$, and $v = (v_1(x), v_2(x), \ldots, v_{t_1}(x))$, where $v_i(x) \in \mathcal{P}$.

Since the weight of $\pi(u)$ is even, then u contains even number of coordinates equal to $1 + x + x^2 + \cdots + x^{pr-1}$. Let $u_i(x) = 1 + x + x^2 + \cdots + x^{pr-1}$ for $i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}$. Then, $\langle u, v\rangle$ and $\langle v, u\rangle$ are:

$$\begin{aligned}
\langle \boldsymbol{u}, \boldsymbol{v} \rangle &= \sum_{i=1}^{t_3} u_i(x) v_i(x^{-1}) = \\
&= \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} (1 + x + x^2 + \cdots + x^{pr-1})\ v_i(x^{-1}) = \\
&= \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} \tfrac{x^{pr}-1}{x-1}\ v_i(x^{-1}) = \\
&= \tfrac{x^{pr}-1}{x-1} \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x^{-1})\ , \\
\langle \boldsymbol{v}, \boldsymbol{u} \rangle &= \sum_{i=1}^{t_3} v_i(x) u_i(x^{-1}) = \\
&= \sum_{i=1}^{t_3} v_i(x) u_i(x) = \\
&= \tfrac{x^{pr}-1}{x-1} \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x)\ .
\end{aligned} \tag{6.18}$$

Thus,

$$\begin{aligned}
&\langle \boldsymbol{u}, \boldsymbol{v} \rangle \equiv 0 \mod(x^{pr} - 1) \quad \text{if and only if} \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x^{-1}) \equiv 0 \mod(x - 1)\ , \\
&\langle \boldsymbol{v}, \boldsymbol{u} \rangle \equiv 0 \mod(x^{pr} - 1) \quad \text{if and only if} \sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x) \equiv 0 \mod(x - 1)\ .
\end{aligned} \tag{6.19}$$

We consider the coordinates $v_i(x)$ of $v \in E_\sigma(\mathcal{C})$. By definition $E_\sigma(\mathcal{C}) = \{v \in \mathcal{C} |\ \mathrm{wt}(v|\Omega_i) \equiv 0 \,(\mathrm{mod}\, 2),\ \ i = 1, \ldots, t_1\}$, which implies that the weight of $v_i(x)$ is even for $1 \le i \le t_1$. When $v_i(x)$ has even number of nonzero coefficients, then $v_i(1) = 0$ in $\mathbb{F}_2$, which means 1 is a root of $v_i(x)$. Therefore, $x - 1$ divides $v_i(x)$ for $1 \le i \le t_1$, i.e.

$$v_i(x) \equiv 0 \mod(x - 1). \tag{6.20}$$

Thus,

$$\sum_i v_i(x) \equiv 0 \mod(x - 1)$$

for any i and in particular, this will also be satisfied for $i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}$. Hence,

$$\sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x) \equiv 0 \mod(x - 1). \tag{6.21}$$

Does $\sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x^{-1}) \equiv 0 \mod(x - 1)$ also hold?

One can show that the following holds:

- $v_i(x^{-1}) \equiv v_i(x^{pr-1})$ in $\mathcal{R}$, for $1 \le i \le t_1$;

- $v_i(x) = (x-1)v_i'(x)$, for some $v_i'(x) \in \mathcal{R}$, $1 \le i \le t_1$ (from Eq.(6.20)).
- $x^{pr-1} - 1 \equiv 0 \bmod(x-1)$, since $x \equiv 1 \bmod(x-1) \implies x^{pr-1} \equiv 1 \bmod(x-1)$.

Then,

$$v_i(x^{-1}) = v_i(x^{pr-1}) = (x^{pr-1} - 1)\ v_i'(x^{pr-1}) \equiv 0 \bmod(x-1)\ ,$$

for some $v_i'(x) \in \mathcal{R}$, for $1 \le i \le t_1$. In particular, this will also hold for $i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}$ and therefore

$$\sum_{i \in \{\alpha_1, \alpha_2, \ldots, \alpha_{2s}\}} v_i(x^{-1}) \equiv 0 \bmod(x-1). \tag{6.22}$$

Combining Eqs. (6.19), (6.21), and (6.22), we derive the following proposition.

Proposition 2.2. Let $\mathcal{C}$ be a binary self-dual code having an automorphism σ of type $pr-(t_1;\ 0)$. Then $u \in F_\sigma(\mathcal{C})$, $v \in E_\sigma(\mathcal{C})$ are orthogonal, namely $\langle u, v\rangle \equiv 0 \bmod(x^{pr}-1)$ and $\langle v, u\rangle \equiv 0 \bmod(x^{pr}-1)$.

Summarizing Proposition 2.1, Proposition 2.2, and Lemma 1.1, we can prove Lemma 2.1.

Proof. Let $u,\ v \in \mathcal{C}$, where $\mathcal{C} = F_\sigma(\mathcal{C}) \oplus E_\sigma(\mathcal{C})$. Then,

$$u = u' + u'', \quad v = v' + v'',$$

where $u',\ v' \in F_\sigma(\mathcal{C})$ and $u'',\ v'' \in E_\sigma(\mathcal{C})$. The inner product of u and v is

$$\langle u, v\rangle = \langle u' + u'', v' + v''\rangle =$$

$$= \langle u', v'\rangle + \langle u', v''\rangle + \langle u'', v'\rangle + \langle u'', v''\rangle.$$

The first term $\langle u', v'\rangle = 0$ according to Proposition 2.1. The next two terms are also 0 because of Proposition 2.2. The last term $\langle u'', v''\rangle$ is also 0 because $\varphi(E_\sigma(\mathcal{C}))$ is a self-orthogonal code (Lemma 1.1). Therefore, $\langle u, v\rangle = 0$ for any $u,\ v \in \mathcal{C}$. □

Remark 2. *In case the binary self-dual code $\mathcal{C}$ has an automorphism γ of odd prime order p with c cycles and no fixed points (type $p-(c,0)$), the fixed subcode $F_\gamma(\mathcal{C})$ is also self-dual [31] and, the image $\varphi(E_\gamma(\mathcal{C}))$ of the even subcode $E_\gamma(\mathcal{C})$ is also self-orthogonal [65]. Following the proofs of Proposition 2.1 and Proposition 2.2, one can conclude that they also hold for p instead of pr. Therefore, Lemma 2.1 is also a valid statement for self-dual codes with an automorphism of type $p-(c,0)$.*

2.2. Shift-sum Decoding for Cyclic codes

A decoding concept introduced in [11] is on decoding cyclic codes by using the set of the minimal weight codewords of the dual code. Later in [64], this concept is called *Shift-Sum Decoding*.

Cyclic codes are introduced in detail in Section 1.6. Here we recall only properties used for the cyclic code decoding in [11] and, we point out a difference in one of the general notations.

Let C_1 be a cyclic $[n,k]_2$ code generated by polynomial $g(x) = g_0 + g_1x + g_2x^2 + \cdots + g_rx^r$, where $g(x) \in \mathcal{R}_1 = \mathbb{F}_2[x]/(x^n-1)$ and $r = n-k$. Then C_1 can be generated by the matrix

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots & g_{r-1} & g_r & \ldots & 0 \\ & & & \vdots & & & & \\ 0 & \ldots & 0 & g_0 & g_1 & \ldots & & g_r \end{pmatrix}.$$

Regarding the code C_1, it is known that:

- the polynomial $g(x)$ divides $x^n - 1$;
- the polynomial $h(x)$, $h(x) = \frac{x^n-1}{g(x)}$, is the check polynomial of the cyclic code C_1;
- every codewords $c(x)$ in C_1 is divisible by the generator polynomial of C_1, i.e., $c(x) \equiv 0 \bmod(g(x))$;
- $c(x)h(x) \equiv 0 \bmod(x^n - 1)$;
- the dual code ${C_1}^\perp$ is also cyclic, generated by the reverse of the check polynomial $h(x)$, i.e. ${C_1}^\perp$ is generated by $h^R(x) = x^{deg(h(x))}h(x^{-1})$.

Note that *only in this subsection* a polynomial $b(x)$ of ${C_1}^\perp$ refers to the polynomial, which is in reverse order of the codeword as a vector in ${C_1}^\perp$. Considering $b(x)$ in this way, $b(x)$ will be divisible by $h(x)$ instead of $h^R(x)$, i.e. $b(x) = h(x)q_1(x)$ for some polynomial $q_1(x)$. Then, for every $c(x)$ of C_1 and every $b(x)$ of ${C_1}^\perp$ it holds that $c(x)b(x) \equiv 0 \bmod(x^n - 1)$. It is because $b(x) = h(x)q_1(x)$, $c(x) = g(x)q_2(x)$ and then

$$c(x)b(x) = g(x)q_2(x)h(x)q_1(x) = (x^n - 1)q_1(x)q_2(x) = 0 \mod(x^n - 1) \ .$$

This notation of $b(x)$ of ${C_1}^\perp$ is in correspondence to [11].

The *support* $supp(f(x))$ of any polynomial $f(x) = \sum_{l=0}^{n-1} f_i x^i$ in $\mathcal{R}$ is the set of indices for which the coefficients of $f(x)$ are nonzero, i.e. $supp\ f(x) = \{i \mid f_i \neq 0,\ \forall i\}$. Then, the weight of $f(x)$ is $wt(f(x)) = |supp\ f(x)|$.

Let ${C_1}^\perp$ have a minimum weight $d^\perp$ and $b(x) = x^{b_1} + x^{b_2} + \cdots + x^{b_{d^\perp}}$ be a minimum weight codeword of ${C_1}^\perp$. Thus, its support is $supp(b(x)) = \{b_1, b_2, \cdots, b_{d^\perp}\}$, where $b_i \in \{0, 1, \ldots, n-1\}$. As ${C_1}^\perp$ is cyclic and $b(x) \in C^\perp$, then each cyclic shift of $b(x)$ is also in ${C_1}^\perp$, i.e. $x^s b(x) \in C^\perp$ for $s = 1, \ldots, pr$. In particular, the polynomial $x^{-b_1}b(x) \in {C_1}^\perp$ and it starts with x^0 as $x^{-b_1}b(x) = x^0 + x^{b_2-b_1} + \cdots + x^{b_{d^\perp}-b_1}$. Thus, instead of $b(x)$ of $C^\perp$ we can use $x^{-b_1}b(x)$ of $C^\perp$. Hence, we can always assume that $b_1 = 0$, which leads to $supp(b(x)) = \{0, b_2, \cdots, b_{d^\perp}\}$.

Encoding by the cyclic code C_1 is defined by its generator polynomial $g(x)$, that is: a message $m(x) = m_0 + m_1x + m_2x^2 + ...m_{k-1}x^{k-1}$ of length k is encoded into $c(x) = g(x)m(x)$. Let the encoded message $c(x)$ be transmitted and $r(x) = g(x)a(x) + e(x)$ be received where $e(x)$ is the error polynomial. Then, the product of the received $r(x)$ with every dual codeword is:

$$r(x)b(x) = (c(x) + e(x))\, b(x) = c(x)b(x) + e(x)b(x) = e(x)b(x) \quad mod(x^n - 1)$$

The product $r(x)b(x)$ is denoted by $w(x)$, and it can be considered as a syndrome because its value only depends on the error [11].

Note that if $e(x) \notin C_1$, then $w(x) = r(x)b(x) = e(x)b(x)$ is a nonzero linear combination of $b(x)$ and shifted $b(x)$ (or only shifted $b(x)$), where $b(x)$ as well as the shifted $b(x)$ are in ${C_1}^\perp$ because ${C_1}^\perp$ is cyclic. This concludes that $w(x)$ is a nonzero codeword of ${C_1}^\perp$ and therefore $wt(w(x)) \geq d^\perp$.

If the error is $e(x) = x^{e_1} + x^{e_2} + \cdots + x^{e_r}$, or equivalently, the errors are on positions e_1,

e_2,...,e_r, then $w(x)$ can be written as:

$$\begin{aligned} w(x) = e(x)b(x) &= (x^{e_1} + x^{e_2} + \cdots + x^{e_r})(1 + x^{b_2} + \cdots + x^{b_{d^\perp}}) \quad mod(x^n - 1) \\ &= x^{e_1} + x^{e_2} + \cdots + x^{e_r} + \\ &\quad x^{e_1+b_2} + x^{e_2+b_2} + \cdots + x^{e_r+b_2} + \\ &\quad x^{e_1+b_3} + x^{e_2+b_3} + \cdots + x^{e_r+b_3} + \\ &\quad \vdots \\ &\quad x^{e_1+b_{d^\perp}} + x^{e_2+b_{d^\perp}} + \cdots + + x^{e_r+b_{d^\perp}}, \end{aligned} \tag{6.23}$$

where the exponents $e_i + b_j$ are calculated *mod* n. This representation can be seen as: the first row contains all error positions of $r(x)$, the second row - all error positions shifted with b_2 positions, the next row - all error positions shifted with b_3 positions and so on, the last row - all error positions shifted with $b_{d^\perp}$ positions. Thus, any nonzero coefficient in $w(x)$ is an error or a shifted error with b_2, b_3,..., or $b_{d^\perp}$ positions.

In order to move the shifted error positions in $w(x)$ into the original error positions, we can shift back by b_2, b_3,..., $b_{d^\perp}$ positions, which is equivalent to the multiplication of $w(x)$ with x^{-j}, for $j \in \{b_1, b_2, ..., b_{d^\perp}\}$. Considering all $d^\perp$ polynomials $w(x), x^{-b_2}w(x), x^{-b_3}w(x), ..., x^{-b_{d^\perp}}w(x)$ $mod(x^n - 1)$, one can conclude that they are all of the same weight, and any nonzero coefficient of $w(x)$ is at the original error position in one of these shifts. Therefore, the polynomials all together have at least $wt(w(x))$ errors in their original positions. They have at least $wt(w(x))$ errors in their original positions since a shifted error can eventually also be at another error position in some shift [11].

Example 2.1. *Shifts of $w(x)$ for BCH(15,5,7) code*[2]

The BCH (15,5,7) code is a cyclic code with generator polynomial $g = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$ and generator matrix

$$G_{15} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The parity-check polynomial is $h(x) = \frac{x^{15}-1}{g(x)} = x^5 + x^3 + x + 1$ and the dual of this BCH code is a [15, 10, 4] code.

Let codeword $c(x) = 1 + x^3 + x^4 + x^6 + x^8 + x^9 + x^{10} + x^{11}$ be sent and let $r(x) = x^3 + x^4 + x^5 + x^6 + x^8 + x^9 + x^{10} + x^{11} + x^{12}$ be the received vector. Then the error vector is $e(x) = 1 + x^5 + x^{12}$.

Consider a codeword $b(x) = 1 + x^{10} + x^{11} + x^{13}$ of the dual code with minimum weight $d^\perp = 4$. Then the polynomial $w(x)$ from Eq. 6.23 is

$$\begin{aligned} w(x) = e(x)b(x) &= (1 + x^5 + x^{12})(1 + x^{10} + x^{11} + x^{13}) \quad mod(x^{15} - 1) \\ &= 1 + x^5 + x^{12} + \\ &\quad x^{10} + 1 + x^7 + \\ &\quad x^{11} + x + x^8 + \\ &\quad x^{13} + x^3 + x^{10} \\ &= 1 + x^3 + x^5 + x^7 + x^8 + x^{11} + x^{12} + x^{13}, \end{aligned}$$

[2]This example is created for this work and not being a part of [11].

where the error positions 1, x^5 *and* x^{12} *are all in* $w(x)$. *The other non zero positions in* $w(x)$ *are shifted error positions, like* x^3 *shifted by* x^{-13} *is* x^5, x^7 *shifted by* x^{-10} *is* x^{12}, *i.e. shifted by* x^{-j}, $j = 10, 13$ *where* $j \in supp(b(x)) = \{0, 10, 11, 13\}$.

The vectors $w(w)$, $x^{-10}w(x)$, $x^{-11}w(x)$, $x^{-13}w(x)$ *are:*

$$\begin{aligned} w(x) &= 1 + x^3 + x^5 + x^7 + x^8 + x^{11} + x^{12} + x^{13}, \\ x^{-10}w(x) &= x + x^2 + x^3 + x^6 + x^8 + x^{10} + x^{12} + x^{13}, \\ x^{-11}w(x) &= 1 + x + x^2 + x^5 + x^7 + x^9 + x^{11} + x^{12}, \\ x^{-13}w(x) &= 1 + x^3 + x^5 + x^7 + x^9 + x^{10} + x^{13} + x^{14}. \end{aligned}$$

The main idea in the decoding concept in [11] is in all the polynomials $w(x)$, $x^{-b_2}w(x)$, $x^{-b_3}w(x)$,...,$x^{-b_{d^\perp}}w(x)$ $\mathrm{mod}(x^n - 1)$ to count the frequency of 1 at position j. From Eq. 6.23, it is expected that the frequency in the error positions will be higher than in any other position, which implies that *a larger frequency is an indicator for an error* in the corresponding position. The frequency is denoted by Φ_j and calculated by:

$$\Phi_j = \sum_{i \in \mathrm{sup}(b(x))} w_{i+j \bmod(n)}, \quad j = 0, 1, 2, \cdots, n-1,$$

where $w_{b_i+j \bmod(n)}$ is counted in position j since the shift of $w(x)$ by $-b_i$ moves the position $b_i + j \bmod(n)$ into j.

Definition 2.1. [11] Two different codewords $c_1(x)$, $c_2(x) \in C_1$ are called *cyclically different* if no shift i exist, such that $x^i c_1(x) = c_2(x) \bmod(x^n - 1)$.

Example 2.1 continuation
Counting the frequency Φ_j *for the above example:* 1 *appears 3 times,* x *appears 2 times,* x^2 *appears 2 times and so on: for* $j = 0, 1, ..., 14$, Φ_j *is:*

position j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Φ_j	**3**	2	2	**3**	0	**3**	1	**3**	2	2	2	2	**3**	**3**	1

The positions 0, 3, 5, 7, 12, 13 have the highest frequencies, and all the error positions are among them.

We obtain that the BCH (15,5,7) code has 7 cyclically different minimum weight dual codewords. One set of them contains the following vectors:

1	(1,0,1,0,1,1,0,0,0,0,0,0,0,0,0)	5	(1,1,0,0,0,1,0,0,1,0,0,0,0,0,0)
2	(1,0,0,0,0,1,1,1,0,0,0,0,0,0,0)	6	(1,0,0,1,1,0,0,0,0,1,0,0,0,0,0)
3	(1,1,0,1,0,0,0,1,0,0,0,0,0,0,0)	7	(1,0,0,0,1,0,0,0,1,0,1,0,0,0,0)
4	(1,0,0,1,0,0,1,0,1,0,0,0,0,0,0)		

Assume that the dual code $C_1{}^\perp$ has L cyclically different minimum weight codewords $b^{(l)}(x)$, $l = 1, 2, \cdots, L$. The described process above can be applied to each of these codewords and accumulate the sums per position. Then, the sum Φ_j becomes:

$$\Phi_j = \sum_{l=1}^{L} \sum_{i \in \mathrm{sup}(b^{(l)}(x))} w^{(l)}_{i+j \bmod n}, \quad j = 0, 1, 2, \cdots, n-1, \tag{6.24}$$

where the values of Φ_j are bounded by $0 \leq \Phi_j \leq Ld^\perp$ [11].

Example 2.1 continuation
The results for Φ_j calculated for the set of 7 cyclically different minimum weight dual codewords listed above and the received vector $r(x)$ are:

position j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
total Φ_j	**18**	12	12	16	12	**18**	12	12	12	12	14	14	**18**	12	14

The highest frequency is 18 *and it is in the positions* 0, 5 *and* 12 *which are exactly the error positions since in our example* $e(x) = 1 + x^5 + x^{12}$.

The decoding strategy in [11] can be summarized in the following steps:

1. Generate all or almost all cyclically different codewords of minimum weight $d^\perp$ in the dual code ${C_1}^\perp$.
2. $r(x)$ - received polynomial
   Compute $r(x)h(x)$. If $r(x)h(x) = 0$ then $r(x)$ belongs to the code C_1 - end, otherwise 3.
3. For every minimum weight dual codeword $b(x)$ generated in step 1 and for $r(x)$ compute $w(x) = b(x)r(x)$, $x^{-b_2}w(x)$, $x^{-b_3}w(x), \ldots,$ $x^{-b_{d^\perp}}w(x)$ $\mathrm{mod}(x^n - 1)$, for all $b_j \in supp(b(x))$.
4. Compute Φ_j defined in (6.24) for $j = 0, 1, ..., n-1$ and determine the maximum value.
5. Determine the index j_m (or indexes) with the maximal value Φ_j and flip the coordinate j_m of $r(x)$ by adding x^{j_m} to $r(x)$, i.e. $r(x)$ becomes $r(x) + x^{j_m}$.
6. If $r(x) + x^{j_m}$ belongs to the code C_1 - end, otherwise repeat from step 3.

The idea of shifting and summing up in cyclic codes we use in a new algorithm for decoding self-dual codes having an automorphism of type $pr - (t_1;\ 0)$.

2.3. Hard-decision Iterative Decoding of Self-dual Codes of a specific type

Consider all elements of Section 1.1 by the same names and notations.

Let a codeword $c \in \mathcal{C}$ be transmitted and $r = c + e$ be received where e is the error vector. In polynomial representation it is:

$$(r_1(x), r_2(x), \ldots, r_{t_1}(x)) = (c_1(x), c_2(x), \ldots, c_{t_1}(x)) + (e_1(x), e_2(x), \ldots, e_{t_1}(x)),$$

where $r_i(x)$, $c_i(x)$, $e_i(x)$ are in $\mathcal{R} = \mathbb{F}_2[x]/(x^{pr} - 1)$.

We denote by w the inner product of the received r with any minimum weight codeword $b \in \mathcal{C}$:

$$w(x) = \langle r, b\rangle = \sum_{i=1}^{t_1} r_i(x)b_i(x^{-1}) \mod (x^{pr} - 1)\ . \tag{6.25}$$

As $r = c + e$, for $w(x)$ it follows that:

$$w(x) = \langle r, b\rangle = \sum_{i=1}^{t_1} c_i(x)b_i(x^{-1}) + \sum_{i=1}^{t_1} e_i(x)b_i(x^{-1}) \mod (x^{pr} - 1)$$

On this step we use Lemma 2.1 proved in Section 2.1.

According Lemma 2.1, the term $\sum_{i=1}^{t_1} c_i(x)b_i(x^{-1})$ is equal to zero. Therefore,

$$w(x) = \langle r, b\rangle = \sum_{i=1}^{t_1} e_i(x)b_i(x^{-1}) \mod (x^{pr} - 1).$$

It is clear that, if the error vector e is zero, i.e., $e_i(x) = 0$ for $1 \leq i \leq t_1$, then also w is zero. In the case when e is nonzero, $w(x)$ is a polynomial of degree at most $pr - 1$. Further, we define an indicator that determines the flipping positions in the received vector so that $w(x)$ becomes zero.

Let the support of $b_1(x)$, $supp(b_1(x))$, be the set $\{\beta_1, \beta_2, \ldots, \beta_d\}$ which means that $b_1(x) = x^{\beta_1} + x^{\beta_2} + \cdots + x^{\beta_d}$. Then w can be written as:

$$\begin{aligned} w(x) &= \sum_{i=1}^{t_1} e_i(x) b_i(x^{-1}) \mod (x^{pr} - 1) \\ &= e_1(x) b_1(x^{-1}) + \sum_{i=2}^{t_1} e_i(x) b_i(x^{-1}) \mod (x^{pr} - 1) \\ &= e_1(x) x^{-\beta_1} + e_1(x) x^{-\beta_2} + \cdots + e_1(x) x^{-\beta_d} + \sum_{i=2}^{t_1} e_i(x) b_i(x^{-1}) \mod (x^{pr} - 1) \end{aligned}$$

If $e_1(x) = x^{\epsilon_1} + x^{\epsilon_2} + \cdots + x^{\epsilon_r}$, then the expression for w can be further reorganized as:

$$\begin{aligned} w(x) = \; & x^{\epsilon_1 - \beta_1} + x^{\epsilon_2 - \beta_1} + \cdots + x^{\epsilon_r - \beta_1} + \\ & x^{\epsilon_1 - \beta_2} + x^{\epsilon_2 - \beta_2} + \cdots + x^{\epsilon_r - \beta_2} + \\ & x^{\epsilon_1 - \beta_3} + x^{\epsilon_2 - \beta_3} + \cdots + x^{\epsilon_r - \beta_3} + \\ & \vdots \\ & x^{\epsilon_1 - \beta_d} + x^{\epsilon_2 - \beta_d} + \cdots + + x^{\epsilon_r - \beta_d} + \\ & + \sum_{i=2}^{t_1} e_i(x) b_i(x^{-1}) \qquad \mod (x^{pr} - 1), \end{aligned} \tag{6.26}$$

where all $\epsilon_i - \beta_j$ are calculated in mod(pr).

Eq. (6.26) can be seen as: the first row contains all error positions of $e_1(x)$ shifted by $-\beta_1$ positions; the second row - all error positions of $e_1(x)$ shifted by $-\beta_2$ positions and so on, the d^{th} row- all error positions of $e_1(x)$ shifted by $-\beta_d$ positions. The rest can be considered in the same way regarding the error positions of e_2, e_3, $\ldots$, e_{t_1} with shifts corresponding to the supports of $b_2, b_3, \ldots, b_{t_1}$. Note that many of these terms can be cancelled out since these shifted error positions can be the same for different $e_i b_i$.

To move the shifted error positions of $e_1(x)$ in $w(x)$ into the original error positions of $e_1(x)$, we use the idea presented in [11] to multiply $w(x)$ by x^{β_j}, where $\beta_j \in supp(b_1(x))$. When w in Eq. (6.26) is multiplied by x^{β_1}, the first row of $x^{\beta_1} w(x)$ becomes exactly $e_1(x)$. Multiplying w with x^{β_2}, the second row in Eq. (6.26) will become $e_1(x)$ and so on.

Thus, each of the polynomials:

$$x^{\beta_1} w(x),\ x^{\beta_2} w(x),\ \ldots,\ x^{\beta_d} w(x) \quad \mod (x^{pr} - 1) \tag{6.27}$$

contains $e_1(x)$, $x^{\beta_j - \beta_s} e_1(x)$, and $x^{\beta_j} \sum_{i=2}^{t_1} e_i(x) b_i(x^{-1})$ (all in $\mathcal{R}$), which are the original and shifted error positions of $e_1(x)$ and shifted error positions of $e_i(x)$, $i = 2, 3, \ldots, t_1$. Some of the original or shifted error positions of $e_1(x)$ can be cancelled out with some of the shifted error positions of $e_2(x), e_3(x), \ldots, e_{t_1}(x)$. If in all polynomials $x^{\beta_s} w(x)$ in Eq. (6.27) we count the number of 1s in position j, for $j = 1, 2, \ldots, pr$, it is expected that in some of the error positions, the number of 1s will be greater than the number of 1s in the other positions.

The same process can be repeated for the polynomials

$$e_2(x) b_2(x^{-1}),\ e_3(x) b_3(x^{-1}),\ \ldots,\ e_{t_1}(x) b_{t_1}(x^{-1}).$$

In general, if we consider $e_s(x)b_s(x^{-1})$, where $supp(b_s(x)) = \{\beta_1^{(s)}, \beta_2^{(s)}, \ldots, \beta_{d_s}^{(s)}\}$, then each of the polynomials:

$$x^{\beta_1^{(s)}} w(x),\ x^{\beta_2^{(s)}} w(x),\ \ldots,\ x^{\beta_{d_s}^{(s)}} w(x) \mod (x^{pr} - 1) \tag{6.28}$$

contains $e_s(x)$, $x^{\beta_j^{(s)} - \beta_i^{(s)}} e_s(x)$, $i \neq j$ and $1 \leq i \leq d_s$, and $\beta_j^{(s)} \sum_{\substack{i=1 \\ i \neq j}}^{t_1} e_i(x) b_i(x^{-1})$, all in $\mathcal{R}$. The last are the original error positions of $e_s(x)$, shifted error positions of $e_s(x)$ and shifted error positions of $e_i(x)$, $i = 1, 2, \ldots, t_1$, $i \neq s$. The original error positions (or a part of them which are not cancelled out) are in all d_s polynomials in Eq. (6.28), while the shifted error positions are different in each of these polynomials. Thus, counting the number of 1s in each position from 1 to pr in the elements in Eq. (6.28) will lead to a greater number of 1s in the original error positions (or some of them). Stated differently, a larger number of 1s will be an indicator for an error in this position. The polynomials in Eq. (6.28) are created for every s, $1 \leq s \leq t_1$, and the number of 1s in each position is included in $\Phi_j^{(s)}$ (defined in Eq. (6.29)).

The total number of positions in the supports $supp(b_s(x))$, for $1 \leq s \leq t_1$, for a codeword b is equal to $wt(b)$. Thus, $d_1 + d_2 + \cdots + d_{t_1} = wt(b) = d$ when b is a minimum weight codeword. Then *for all* s, $1 \leq s \leq t_1$, *the total number of elements in Eq.* (6.28) *is also* $wt(b)$.

Let $\mathcal{C}$ has L cyclically different minimum weight codewords. In our case, *cyclically different* codewords mean that one cannot be obtained from the other by applying σ^t, for some t, that is, $b \neq \sigma^t(c)$ for $1 \leq t \leq pr - 1$, $\forall b,\ c \in \mathcal{C}$. The counting of 1s in each position from 1 till pr in the cycles Ω_s is repeated for the polynomials in Eq. (6.28) for all L cyclically different minimum weight codewords b. The number is denoted by $\Phi_j^{(s)}$:

$$\Phi_j^{(s)} = \sum_{l=1}^{L} \sum_{i \in supp(b_s^{(l)}(x))} w_{i+j \mod pr}^{(l)}\ , \qquad j = 0, 1, 2, \cdots, pr - 1, \quad s = 1, 2, \ldots, t_1. \tag{6.29}$$

where $w_{i+j \mod pr}^{(l)}$ is counted in position j since the shift of $w(x)$ by $-i$ moves the position $i + j \mod pr$ into j.

Thus, for L cyclically different minimum weight codewords, we calculate $t_1 pr$ sums:

$$\Phi_0^{(1)}, \Phi_1^{(1)}, \ldots, \Phi_{pr-1}^{(1)},\ \Phi_0^{(2)}, \Phi_1^{(2)}, \ldots, \Phi_{pr-1}^{(2)},\ \ldots,\ \Phi_0^{(t_1)}, \Phi_1^{(t_1)}, \ldots, \Phi_{pr-1}^{(t_1)}\ .$$

As it was mentioned, a higher number of 1s in a position from 1 till pr in the polynomials in Eq. (6.28) is an indicator for an error in this position. Thus, an error is expected in the j^{-th} position in cycle Ω_s if the value of $\Phi_j^{(s)}$ is greater than the sum for any other position in the same cycle Ω_s, and greater than the sum for any position in the other $t_1 - 1$ cycles.

Note that the idea of shifting and counting, introduced in [11] and described in Section 2.2, is specifically and only for cyclic codes. The presence of cyclic cells in the generator matrix of code $\mathcal{C}$ (Eq. (6.2)) and, moreover, the orthogonality in the polynomial representation of each two codewords (Lemma 2.1) makes it possible to use shifting and counting in a similar way.

Example 2.2. *Let $\mathcal{D}_{90}$ be a binary [90,45,14] self-dual code with an automorphism ϕ of type* $15 - (6;\ 0)$. *Without loss of generality, ϕ can be represented as:*

$$\phi = \quad \Omega_1\Omega_2\ldots\Omega_6, \tag{6.30}$$

where Ω_s is a cycle of length 15 for $1 \le s \le 6$. The code $\mathcal{D}_{90}$ holds the conditions in Lemma 2.1 and therefore,

$$u_1(x)v_1(x^{-1}) + u_2(x)v_2(x^{-1}) + \cdots + u_6(x)v_6(x^{-1}) = 0, \tag{6.31}$$

$\forall u, v \in \mathcal{D}_{90}$.

Let the codeword $c = (c_1(x), c_2(x), \ldots, c_6(x))$ be sent and $r = (r_1(x), r_2(x), \ldots, r_6(x))$ be received, where the polynomials $c_i(x)$ and $r_i(x)$ are given in Table 6.1. Then the error vector is $e = (e_1(x), \ldots, e_6(x)) = (0,\ 0,\ 0,\ x^8,\ x^9+x,\ x^{14}+x^3+x^2)$. Thus, there are 6 errors: 1 error in position 8 of Ω_4, 2 errors in positions 1 and 9 in Ω_5 and 3 errors in positions 2, 3 and 14 of Ω_6.

We would like to compute $\Phi_j^{(s)}$, defined in (6.29), for the given r.

i	$r_i(x)$	$c_i(x)$
1	$x^{13}+x^9+x^7+x^6+x^5+x^3+x^2+x+1$	$x^{13}+x^9+x^7+x^6+x^5+x^3+x^2+x+1$
2	$x^{14}+x^{10}+x^9+x^7+x^6+x^5+x^4+x^2+x$	$x^{14}+x^{10}+x^9+x^7+x^6+x^5+x^4+x^2+x$
3	$x^{13}+x^{12}+x^{10}+x^9+x^8+x^7+x^6+x+1$	$x^{13}+x^{12}+x^{10}+x^9+x^8+x^7+x^6+x+1$
4	$x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^6+1$	$x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+1$
5	$x^{14}+x^9+x^8+x^3+1$	$x^{14}+x^8+x^3+x+1$
6	$x^{13}+x^{12}+x^{10}+x^8+x^7+x^2+x$	$x^{14}+x^{13}+x^{12}+x^{10}+x^8+x^7+x^3+x$

Table 6.1: Elements of $\mathbb{F}_2[x]/(x^{15}-1)$

To calculate $\Phi_j^{(s)}$, the set of all cyclically different codewords of weight 14 in $\mathcal{D}_{90}$ are required. In Appendix 5.1, construction of a generator matrix of the code $\mathcal{D}_{90}$ is included. Using this matrix, we obtain all minimum weight codewords. They are 375, where only 25 of them are cyclically different.

Consider a codeword $b = (x^2,\ x^{10}+x^6,\ x^{14},\ x^{12}+x^9+x^5+x,\ x^9+x^7+x^6+x^5,\ x^6+x^2)$. Then, the corresponding $w(x)$

$$w(x) = \langle r, b\rangle = r_1(x)b_1(x^{-1}) + \cdots + r_2(x)b_6(x^{-1}) \mod (x^{15}-1),$$

is $w(x) = x^{14}+x^{11}+x^{10}+x^9+x^8+x^4+x^2+x$.

The supports of $b_i(x)$ are:

$supp(b_1(x))$	$supp(b_2(x))$	$supp(b_3(x))$	$supp(b_4(x))$	$supp(b_5(x))$	$supp(b_6(x))$
2	6, 10	14	1, 5, 9, 12	5, 6, 7, 9	2, 6

Using them, we calculate the 14 polynomials $x^{\beta_j^{(s)}}w(x)$ in Eq. (6.28) for $s = 1, 2, \ldots, 6$. They are 14 since their number is equal to the $wt(b)$, which is 14. The results for $x^{\beta_j^{(s)}}w(x)$ are:

s		$x^{\beta_j^{(s)}}w(x)$	s		$x^{\beta_j^{(s)}}w(x)$
1	x^2w	$x^{13}+x^{12}+x^{11}+x^{10}+x^6+x^4+x^3+x$	4	$x^{12}w$	$x^{14}+x^{13}+x^{11}+x^8+x^7+x^6+x^5+x$
2	x^6w	$x^{14}+x^{10}+x^8+x^7+x^5+x^2+x+1$	5	x^5w	$x^{14}+x^{13}+x^9+x^7+x^6+x^4+x+1$
2	$x^{10}w$	$x^{14}+x^{12}+x^{11}+x^9+x^6+x^5+x^4+x^3$	5	x^6w	$x^{14}+x^{10}+x^8+x^7+x^5+x^2+x+1$
3	$x^{14}w$	$x^{13}+x^{10}+x^9+x^8+x^7+x^3+x+1$	5	x^7w	$x^{11}+x^9+x^8+x^6+x^3+x^2+x+1$
4	x^1w	$x^{12}+x^{11}+x^{10}+x^9+x^5+x^3+x^2+1$	5	x^9w	$x^{13}+x^{11}+x^{10}+x^8+x^5+x^4+x^3+x^2$
4	x^5w	$x^{14}+x^{13}+x^9+x^7+x^6+x^4+x+1$	6	x^2w	$x^{13}+x^{12}+x^{11}+x^{10}+x^6+x^4+x^3+x$
4	x^9w	$x^{13}+x^{11}+x^{10}+x^8+x^5+x^4+x^3+x^2$	6	x^4w	$x^{14}+x^{10}+x^8+x^7+x^5+x^2+x+1$

Similarly, we calculate such 14 polynomials for each of the other 24 cyclically different minimum weight codewords in $\mathcal{D}_{90}$. All together, they are 350 polynomials. Then, in the polynomials corresponding to $s = 1$, we count the frequency of 1, x, x^2 and so on till x^{14}. The frequencies are denoted by $\Phi_0^{(1)}, \Phi_1^{(1)}, \ldots, \Phi_{14}^{(1)}$. Similarly, we count the frequency of 1, x, x^2 and so on till x^{14} in all the polynomials corresponding to $s = 2$. They form the values of $\Phi_0^{(2)}, \Phi_1^{(2)}, \ldots, \Phi_{14}^{(2)}$. And so on until $s = 6$. The values of $\Phi_j^{(s)}$ are given in Table 6.2

s \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	18	32	24	28	25	24	26	26	27	26	22	35	29	33	23
2	25	19	26	24	24	24	26	21	28	23	16	27	23	28	24
3	23	21	25	20	24	22	18	23	23	27	18	25	19	24	24
4	22	23	24	20	28	25	19	21	33	23	33	24	27	26	22
5	34	40	31	32	29	31	23	32	32	**43**	30	32	31	34	30
6	25	28	36	33	22	31	25	31	27	27	25	24	32	31	37

Table 6.2: $\Phi_j^{(s)}$

The maximum value of $\Phi_j^{(s)}$ is 43, which appears in row 5, column $j = 9$. This indicates that an error can be on cycle 5, position 9. Since $e_5(x) = x^9 + x$, indeed an error occurs on the last cycle, position 9 (and position 1).

By flipping the bit on cycle 5, position 9 in r, we correct one error in the received vector. Iteration of this process results in a decoding algorithm, which we describe next.

Here we define our new Hard-decision Iterative Decoding of Self-dual Codes of a particular type. They have an automorphism of type $pr - (t_1;\ 0)$ or $p - (c,\ 0)$ which determines their structure. For this type of binary self-dual codes, Lemma 2.1 holds, and this is used in the decoding scheme.

Hard-decision Iterative Decoding of Self-dual Codes [3]

We suppose that $\mathcal{C}$, σ and G are defined as in Section 1.1.

1) generate all or almost all cyclically different codewords of weight d or $d + o$ for some small o, for example, 2 or 4 or 6. Denote the set by D_1;

2) for the received vector r compute rG. If $rG = 0$, then r belongs to the code $\mathcal{C} \rightarrow$ end, otherwise $\rightarrow$ 3);

3) split r into t_1 polynomials of $\mathbb{F}_2[x]/(x^{pr} - 1)$, i.e., $r = (r_1(x), r_2(x), \ldots, r_{t_1}(x))$;

4) for $r = (r_1(x), r_2(x), \ldots, r_{t_1}(x))$ compute:

- $w(x) = \langle r, b\rangle = r_1(x)b_1(x^{-1}) + r_2(x)b_2(x^{-1}) + \cdots + r_{t_1}(x)b_{t_1}(x^{-1}) \mod(x^{pr} - 1)$

  for $\forall\ b \in D_1$, $b = (b_1(x), b_2(x), \ldots, b_{t_1}(x))$

[3] A pseudocode of the decoding scheme is provided in Appendix 4.

- $x^{\beta_i^{(1)}} w(x) \mod(x^{pr}-1)$ for $\forall\ \beta_i^{(1)} \in supp(b_1(x))$

  $x^{\beta_i^{(2)}} w(x) \mod(x^{pr}-1)$ for $\forall\ \beta_i^{(2)} \in supp(b_2(x))$

  $\vdots$

  $x^{\beta_i^{(t_1)}} w(x) \mod(x^{pr}-1)$ for $\forall\ \beta_i^{(t_1)} \in supp(b_{t_1}(x))$

Note that: (a) for the chosen code $\mathcal{C}$ if $r \in \mathcal{C}$, then $w(x) = 0$, and (b) the products $x^{\beta_i^{(s)}} w(x)$ are cyclic shifts of w with a number of positions which values are from the support of $b_s(x)$.

5) compute $\Phi_j^{(s)}$ defined in Eq. (6.29) for $j = 0, 1, \cdots, pr-1,\ s = 1, 2, \ldots, t_1$;

6) determine $\Phi_{max} = max\{\Phi_j^{(s)} \mid j = 0, 1, \cdots, pr-1,\ s = 1, 2, \ldots, t_1\}$ and find the position with the value Φ_{max}, i.e., find the cycle s_1 and position(s) j_1 such that $\Phi_{j_1}^{(s_1)} = \Phi_{max}$;

7) flip the coordinate of $r_{s_1}(x)$ by adding x^{j_1} to $r_{s_1}(x)$, i.e., $r_{s_1}(x)$ becomes $r_{s_1}(x) + x^{j_1}$;

8) for the modified r repeat from 2).

This algorithm could use a set of cyclically different codewords of weight d or weight slightly higher than d. It is because if in the first set the codewords have only 0 coordinates in some of the cycles of σ, it is clear that no error in this cycle can be corrected. Also, if the first set is very small, the algorithm does not perform error correction close to the error-correction capability of the code. Therefore, experiments are required to find a suitable number of low weight codewords for efficient decoding performance. In Section 2.4, we give three examples and the decoding performance of different sets of codewords.

2.4. Examples

The decoding algorithm is applied on three examples of self-dual code with the required structure, where two of the codes are known, whereas the third one is new. The last is constructed to demonstrate that self-dual codes with an automorphism of the particular type exist, and it is not hard to generate them when their minimum weight is not close to its upper bound Eq. (2.2).

The first is the binary $[90, 45, 14]$ self-dual code $\mathcal{D}_{90}$ from Example 2.2.

A. Decoding of a Binary [90,45,14] Self-dual Code

Example 2.3. *Consider the binary* $[90, 45, 14]$ *self-dual code* $\mathcal{D}_{90}$ *defined in Example 2.2.* $\mathcal{D}_{90}$ *has an automorphism* ϕ *of type* $15-(6;\ 0)$. *Note that* $\mathcal{D}_{90}$ *is an optimal code since the upper bound (Eq. (2.2)) for the minimum distance is* 16. *The code* $\mathcal{D}_{90}$ *holds the conditions in Lemma 2.1, and therefore, the new iterative decoding is applicable to it. The number of errors that* $\mathcal{D}_{90}$ *is capable to correct is* $t \leq \lfloor \frac{d-1}{2} \rfloor$ *errors, i.e., maximum 6 errors.*

We mentioned that construction of a generator matrix of this code is included in Appendix 5.1 and that the code has 375 codewords of weight 14 where 25 of them are cyclically different. Using the same generator matrix, we obtain the sets of codewords with weight 16 and weight 18. They contain 11 745 and 215 915 codewords, respectively. The rank of the sets with codewords of weight 14, 16 and 18 is accordingly 45, 44, and 45. The sets with cyclically different codewords are denoted by B_{14}, B_{16}, and B_{18}, where $|B_{14}| = 25$, $|B_{16}| = 783$ and $|B_{18}| = 14\,399$ (Table 6.3).

Table 6.3: Sets of codewords in the $[90, 45, 14]$ s-d code $\mathcal{D}$

weight i	A_i	rank	cyclically different	in simulations
14	375	45	B_{14}, $\lvert B_{14}\rvert = 25$	D_1, $\lvert D_1\rvert = 25$
16	11 745	44	B_{16}, $\lvert B_{16}\rvert = 783$	D_2, $\lvert D_2\rvert = 450$
18	215 915	45	B_{18}, $\lvert B_{18}\rvert = 14\,399$	D_3, $\lvert D_3\rvert = 340$

In simulations, we use three different sets of cyclically different codewords and compare their performance. The sets are D_1, D_2, and D_3, which are also given in Table 6.3.

We perform simulations on 2 000 random error vectors and random encoded messages for each number of errors t, for $t = 1, 2, \ldots, 8$. The simulation steps are:

- generate a random error vector e of length 90 and weight t;
- generate a random message vector m of length 45;
- encode the message vector m into $c = mG$;
- compute $r = e + c$;
- decode the received vector r using the iteration steps 2 till 8 of Algorithm 9;

The results of the simulations are presented in Table 6.4. They show that the set D_1 of the cyclically different codewords of weight 14 is too small for a good decoding performance. Using each of the other two sets, D_2 and D_3, the decoding algorithm reaches the error-correcting capability of the code and corrects 100% of the errors, where errors are in the range up to 6. Moreover, these two sets in the experiments correct 99.95% and 95.35% of the cases with 7 errors, and 96.8% and 60.35% of the cases with 8 errors.

The results of the experiments indicate a high error-correcting capability of our new algorithm, a capability beyond the upper bound for t.

Note that the rank of the set of weight 16 codewords is 44, which means that it is possible in step 4) to obtain a vector $r(x)$ such that the corresponding $w(x) = 0$ but $rG \neq 0$. That is the reason we also consider the set D_3 with rank 45, where the described exception is not possible.

Table 6.4: Decoding performance of the $[90, 45, 14]$ self-dual code $\mathcal{D}$

Decoding set D_1, $\lvert D_1\rvert = 25$				Decoding set D_2, $\lvert D_2\rvert = 450$				Decoding set D_3, $\lvert D_3\rvert = 340$			
t	*tested*	*corrected*	%	t	*tested*	*corrected*	%	t	*tested*	*corrected*	%
1	90	90	100	1	90	90	100	1	90	90	100
2	2 000	1 932	96.6	2	2 000	2 000	100	2	2 000	2 000	100
3	2 000	1 928	96.4	3	2 000	2 000	100	3	2 000	2 000	100
4	2 000	1 955	97.75	4	2 000	2 000	100	4	2 000	2 000	100
5	2 000	1 930	96.5	5	2 000	2 000	100	5	2 000	2 000	100
6	2 000	1 829	91.45	6	2 000	2 000	100	6	2 000	2 000	100
7	2 000	1 268	63.4	7	2 000	1 999	99.95	7	2 000	1 907	95.35
8	2 000	560	28	8	2 000	1 936	96.8	8	2 000	1 207	60.35

B. Decoding of a Binary [78,39,14] Self-dual Code

Example 2.4. *Let $\mathcal{T}$ be a binary $[78, 39, 14]$ self-dual code with an automorphism ϕ_1 of type $39-(2;\ 0)$. Note that $\mathcal{T}$ is an extremal code [20]. Since $d = 14$, the number of errors that $\mathcal{T}$ can correct is up to 6. As in the previous example, the conditions in Lemma 2.1 are satisfied for the code $\mathcal{T}$, and therefore, the decoding Algorithm 9 is applicable to $\mathcal{T}$.*

A generator matrix of $\mathcal{T}$ is available in Appendix 5.2. We obtain that this example has 3 081 minimum weight codewords and 46 116 codewords of weight 16. The rank of the sets is 39 and 38, respectively. Among them, the cyclically different codewords are 79 with a weight of 14 and 1 644 with 16.

The set of 79 elements is denoted by T_1, and the set of 79 elements of weight 14 together with 244 elements of weight 16 is denoted by T_2 (Table 6.5). The sets T_1 and T_2 are used in the decoding simulations.

Table 6.5: Sets of codewords in the $[78, 39, 14]$ self-dual code $\mathcal{T}$

weight i	A_i	rank	cyclically different	in simulations
14	3 081	39	B_{14}, $\lvert B_{14}\rvert = 79$	T_1, $\lvert T_1\rvert = 79$
16	64 116	38	B_{16}, $\lvert B_{16}\rvert = 1\,644$	T_2', $\lvert T_2'\rvert = 244$
14,16	67 197	39	$B_{14} \cup B_{16}$	$T_2 = T_1 \cup T_2'$

Similarly to the first decoding example, for both sets T_1 and T_2, for each t, $1 \le t \le 8$, we perform simulations on 2 000 random error vectors and random encoded messages. The simulation steps are the same. The results are included in Table 6.6.

Table 6.6: Decoding performance of the $[78, 39, 14]$ self-dual code $\mathcal{T}$

Decoding set T_1, $\lvert T_1\rvert = 79$				Decoding set T_2,$\lvert T_2\rvert = 323$			
t	*tested*	*corrected*	%	t	*tested*	*corrected*	%
1	78	78	100	1	78	78	100
2	2 000	2 000	100	2	2 000	2 000	100
3	2 000	2 000	100	3	2 000	2 000	100
4	2 000	2 000	100	4	2 000	2 000	100
5	2 000	2 000	100	5	2 000	2 000	100
6	2 000	2 000	100	6	2 000	2 000	100
7	2 000	1 974	98.7	7	2 000	1 995	99.75
8	2 000	1 530	76.5	8	2 000	1 725	86.25

The values in Table 6.6 show that the set of 14 weight codewords are sufficient for the complete decoding of 6 errors which is the upper bound for the error capability of this code example. Differently from the first code, here both, the set T_1 and T_2 have a high error-correcting performance beyond the upper bound for t.

In both decoding examples, the complete set of 16 (or 16 and 18) weight cyclically different codewords is not considered. There is a trade-off between the speed and memory of the decoder from one side and the decoding performance of the algorithm from another side. To achieve decoding up to $\frac{d-1}{2}$ errors for the first example, it is sufficient to use set D_3 with 340 codewords and set T_1 with 79 codewords for the second example.

C. Decoding of a Binary [266,133,36] Self-dual Code

Example 2.5. *Let B_{266} be a binary $[266, 133, 36]$ self-dual code with an automorphism ϕ_2 of type $133-(2;0)$. The upper bound for the minimum weight of self-dual codes of length 266 is*

48 *(Eq. (2.2)). To the best of our knowledge, there is no example of a self-dual* $[266, 133, d \geq 36]$ *code. The code* B_{266} *has an error-correcting capability of* 17 *errors. Since the code possesses the specific automorphism of type* $pr - (t_1;\ 0)$*, then the conditions in Lemma 2.1 are satisfied, and the decoding Algorithm 9 is applicable to* B_{266}*.*

First, we describe the construction of the code and then present the decoding experiments.

The code B_{266} possesses an automorphism ϕ_2 of type $133 - (2; 0)$. Then:
1) $B_{266} = F_{\phi_2}(B_{266}) \oplus E_{\phi_2}(B_{266})$;
2) the fixed subcode $\pi(F_{\phi_2}(B_{266}))$ is a binary $[2, 1]$ self-dual code;
3) the vectors of the image $\varphi(E_{\phi_2}(B_{266}))$ are from $\mathcal{P}^2$, where $\mathcal{P} \subset \mathbb{F}_2/(x^{133} - 1)$.

From 2), it follows that the generator matrix of $F_{\phi_2}(B_{266})$ is $X = \begin{pmatrix} l & l \end{pmatrix}$, where $l = (1, 1, \ldots, 1)$ is the all-ones vector in $\mathbb{F}_2^{133}$.

Applying 3) by computing experiment, we obtain an example for the generator matrix Y of $F_{\phi_2}(B_{266})$:

$$Y = \begin{pmatrix} y_{1,1} & y_{1,2} \\ \vdots & \vdots \\ y_{9,1} & y_{9,2} \end{pmatrix},$$

where $y_{i,j}$ are right-circulant 3×133 cells for the first 2 rows in Y and $y_{i,j}$ are right-circulant 18×133 cells for the next 7 rows. The first rows of these circulant matrices are given in Table A.5 in Appendix 5.

For code B_{266}, only part of the codewords with weights 36, 38 and 40 are generated.They are reduced to cyclically different codewords and are divided into three sets, denoted by L_{36}, L_{38} and L_{40} (Table 6.7). To decode code B_{266}, we use eight sets M_i, for $1 \leq i \leq 8$. Each of them consists of a different number of elements from L_{36}, L_{38} and L_{40}. The set of codewords of L_j, $j = 36,\ 38,\ 40$, included in the set M_i, is denoted by $M_{i,j}$, $1 \leq i \leq 8$. Details are given in Table 6.8.

Table 6.7: Sets of cyclically different codewords in the $[266, 133, 36]$ s-d code B_{266}

weight i	cyclically different	rank
36	L_{36}, $\lvert L_{36}\rvert = 26$	26
38	L_{38}, $\lvert L_{38}\rvert = 275$	133
40	L_{40}, $\lvert L_{40}\rvert = 6\,583$	132

Table 6.8: Decoding performance of the $[266, 133, 36]$ self-dual code B_{266}

M_i	$\lvert M_i\rvert$	$\lvert M_{i,36}\rvert$	$\lvert M_{i,38}\rvert$	$\lvert M_{i,40}\rvert$	rank(M_i)	Correction of t errors in % for t equals to						
						18	17	16	15	14	13	12
M_1	260	22	238	0	132	0.5	4.5	13	32.5	63.5	93.5	99
M_2	300	26	274	0	133	1.5	5	15	46	76.5	98.5	100
M_3	300	0	0	300	132	0	0	4	20.5	47	83	98
M_4	601	0	0	601	132	1	6.5	15.5	44	73.5	97.5	100
M_5	602	26	275	301	133	1.5	10	29.5	52.5	88	100	100
M_6	1489	13	22	1454	133	13.33	27	52	78	97	99	100
M_7	3450	18	94	3338	133	21.5	54	87.5	97.5	100	100	100
M_8	6835	26	275	6534	133	53.33	83.50	99.23	100	100	100	100

As in the previous two examples, for the sets M_i, for $i = 1, \ldots, 8$, and each number of errors t, in this case, $12 \leq t \leq 18$, we are conducting a decoding experiment on 2 000 received encoded messages with t errors. The simulation steps described in Example 2.3 are followed, where the length of the error vector is 266, and the length of the message is 133. The results of the simulations are provided in Table 6.8.

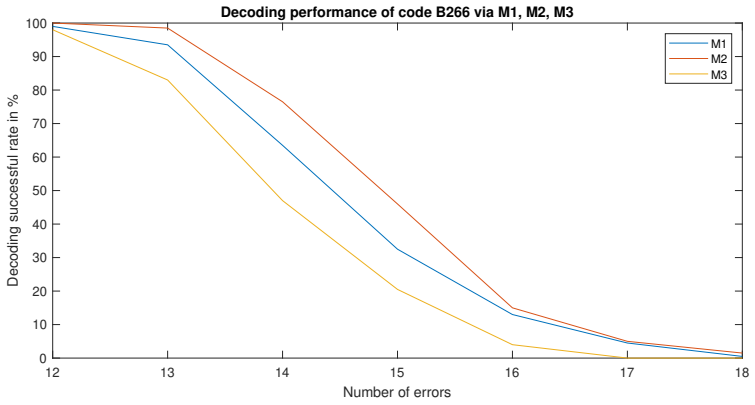

Figure 6.1: Decoding via M_1, M_2, M_3

We first perform the decoding experiment using the sets M_1, M_2 and M_3. Note that M_1 and M_2 consist of cyclically different codewords of weight d and $d+2$ while M_3 - of weight $d+4$. Moreover, M_2 and M_3 have the same cardinality, while the set M_1 is smaller. The results (see Fig. 6.1) show that the error correction efficiency depends on both the weight of the codewords in the decoding set and the number of elements in the set. Using M_2, the decoding Algorithm 9 corrects 15 errors ($15 = t - 2$) in 46% of the cases and 13 errors in 98.5% of the cases, while the decoding using $M3$ corrects 15 and 13 errors in 20% and in 83% of the cases, respectively. Thus, the error-correction performance of a decoding set of codewords of weight d and $d+2$ significantly exceeds the performance of a set of codewords of weight $d+4$, when both sets are of the same cardinality.

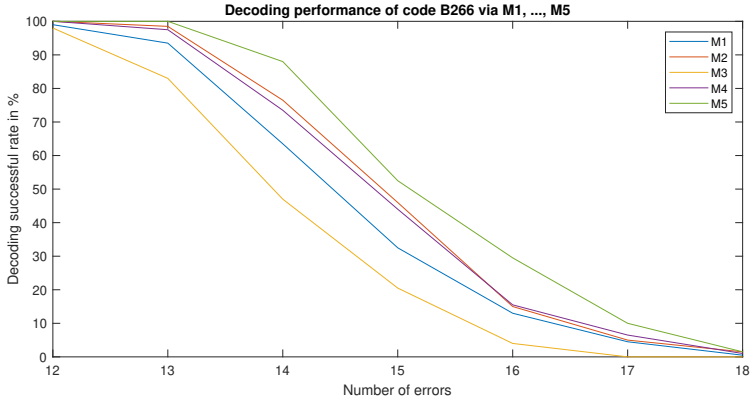

Figure 6.2: Decoding via M_1, ..., M_5

By experiment, we obtain that decoding to achieve error-correction results close to the results

of M_2, when only codewords of weight 40 are used, requires twice as many codewords. This is illustrated by the set M_4 in Fig. 6.2. M_4 consists of twice as many codewords as in M_2, and its error-correction efficiency is very close to that of M_2. For comparison, the performance of set M_5 is also displayed on the same figure. This set is with the same cardinality as M_4, but half of it is of codewords of weight 40, and the other half is codewords of weight 36 and 38. It can be seen that the performance of M_5 outweighs the performance of M_4.

At last, in Fig. 6.3, we present three larger decoding sets. All of them consists of codewords of weight 36, 38 and 40. The last one, M_8, reaches the error-correcting capability of $t = 17$ errors of code B_{266} in 83.5% of the cases, whereas $t - 1$ errors are corrected in 99.23% of the cases. Moreover, the decoding performance of Algorithm 9 using set M_8 goes beyond the error-correcting capability of the code as it corrects 18 errors in 53.33% of the cases.

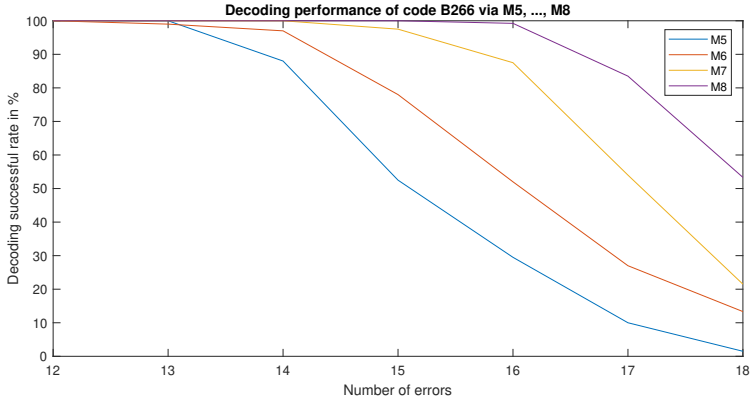

Figure 6.3: Decoding via $M_5, \ldots, M_8$

To conclude, for the correction of 17 errors in 100% of the cases, decoding set with a larger number of cyclically different codewords of weight 36 and 38 has to be generated. Note that L_{36}, L_{38} and L_{40} are only part of the sets of codewords with weights 36, 38 and 40.

Then, it can be expected that a set including all the cyclically different codewords of weight 36 and 38 together with a small subset of L_{40} will achieve correction of 17 errors in 100% of the cases.

3. McEliece Type Cryptosystem Using the New Code Example

Let B_1 be a punctured $[1\,062, 531, d' \geq 160]$ code obtained from the self-dual code B with a generator matrix G, from Section 1.2, by removing the first two columns and the first row of G. Let this generator matrix of B_1 be denoted by G_{short}.

As mentioned at the beginning of this section, the decoding Algorithm 5 is not suitable for large codes. In contrast, the new hard-decision iterative decoding of specific self-dual codes is applicable to code B. We implement the decoding scheme, as described in Algorithm 4, with a set L_{mix} of cyclically different codewords of code B. L_{mix} containing codewords of weight 168, 180, 184 and 188.

Applying the iterative decoding in the decryption Algorithm 6, we obtain the decryption with padding Algorithm 8. The latter is used in the decryption step when defining a McEliece cryptosystem applying code B_1.

The McEliece type cryptosystem applying the code B_1 is specified as follows:

1. *System parameters*:
   - $k = 531$ - length of the message m.
   - $n = 1\,062$ - length of the ciphertext r.
   - $t = 75$ - number of the intentionally added errors.
2. *Key generation*:
   - G_{short} - a generator matrix of a $[1\,062, 531, 75]$ code, a punctured code of a self-dual $[1\,064, 532, d \geq 162]$ code.
   - P - a random $n \times n$ permutation matrix.
   - S - an invertible $k \times k$ matrix such that $SG_{short}P$ is in a systematic form.
   - $G'_{short} = SG_{short}P$, S^{-1} and P^{-1} - the inverse of P and S.
   - *Public key*: $(G'_{short},\ t)$.
   - *Private key*: (G_{short}, G, P, S).
3. *Encryption*:
   - e - a random vector of length n and $wt(e) = t$.
   - $m \to r = G'_{short}m + e$
4. *Decryption:*
   - Decrypt via Algorithm 8.

   Algorithm 8 includes the decoding Algorithm 4 with decoding set L_{mix} containing cyclically different codewords of B of weight 168, 180, 184 and 188.

Remark 3. Due to time limitations, we could not complete the simulations to determine the optimal decoding set L_{mix} of codewords. The codewords of weight 168, 180, 184 and 188 from all linear combinations of up to 8 rows of the 532×1064 generator matrix G and the corresponding parity-check matrix are computed. The sets of cyclically different codewords of weight 168, 180, 184 and 188 contain 45, 3, 30 and 501 elements. These sets are very small and do not perform efficient decoding of 77 errors.

The self-dual $[266, 133, 36]$ code B_{266} of Example 2.5 is constructed via an automorphism of order 133 as the code B. Using a set of 6 835 codewords, the mentioned decoding algorithm corrects $t - 1$ errors in 99.23% of the cases, where $t = 17$. The t errors are corrected in 83.5%.

Since increasing the number of codewords with weight d and $d + o$, for $o = 2, 4$, in the decoding set increases the decoding performance of the algorithm, there will be a set that decodes t errors in 100% of the cases.

Note that the minimum weight of the punctured code B_1 is 160, which means B_1 has an error-correcting capability of up to 79 errors. According to the estimation in Section 5 for a security level of 80 bits, code B_1 is required to correct 75 errors, which is $t - 4$.

In such a setup, we expect that the new algorithm will perform decoding of code B with the same or close to the efficiency of decoding of B_{266} when using a large enough decoding set of codewords.

Algorithm 8: Decryption using padded ciphertext for McEliece with a self-dual $[1064, 532, d' \geq 160]$ code.

Denote
$s = [[0,0],[0,1],[1,0],[1,1]]$,
$i = 1$, $t = 75$, $n = 1062$, $k = n/2$,

Compute
$r' = rP^{-1}$, r-received vector of length k

while $i < 5$
  Pad
    r' into $(** \mid r')$, where $** = s[i]$
  Decode
    $(** \mid r')$ into c_1, $c_1 \in \mathcal{C}$, by Algorithm 9 with decoding set B_{mix}.

  if 5) successful **then**
    Denote
      $c_2 = c_1[3 : n+2]$, $m_2 = c_2[1 : k]$
    Compute
      $m_1 = m_2 * S^{-1}$
    if $(m_1 \in B_1 \quad \wedge \quad \text{weight}(m_1 * G'_{short} - r') == t)$ **then**
      Decrypt r as m_1. Exit.

  $i = i + 1$ increase the index

if $i == 5$ then
  return 'Unsuccessful decryption'. Exit.

The public key, punctured code B_1, is not a self-dual code and has no specific structure. Thus, it does not belong to any specific family of codes. Therefore, the problem of decoding B_1 is expected to be as difficult as the problem of decoding a random code, that is $\mathcal{NP}$-complete, according to Theorem 1.3.

Conclusion RQ4: The algorithms in the Key Generation, Encryption and Decryption of the McEliece type cryptosystem when using code B, obtained in RQ3, are defined. Hard-decision iterative decoding is specially developed to specify the decoding integrated into the Decryption. An algorithm using padded ciphertext determines the Decryption. Further discussion on RQ4 results is in Section 1.

7

Discussion, limitations and future work

As concluding remarks, here we answer the research questions, discuss the limitations of our results, and formulate open problems for further research. Most results are included in two submissions, one to a conference and another to a journal. Details are provided in section Preface 1.

1. Discussion

In this work, we presented our study and research on the first Code-based cryptosystem, the McEliece cryptosystem. It is one of the most studied and proven secure public key encryption schemes. Still, there is a practical limitation for broad use in the current communication systems, namely the large size of its public key.

We attempted to reduce the public key size of the McEliece cryptosystem by using codes with a minimum distance larger than the distance of the codes adopted until now. We considered large minimum distance self-dual codes (optimal self-dual codes) and punctured codes derived from them. To the best of our knowledge, this family of codes has not previously been used in the McEliece encryption scheme. The reasons can be that the optimal self-dual codes are only known for the lengths up to 130, which are too small for the current security requirements, and that there is no fast decoding algorithm for these codes (excepting the extended Golay code).

In this project, we focused on answering the question: *To what extent can a McEliece type cryptosystem using codes with a high error-correcting capability– derived from self-dual codes– be a secure and practically applicable post-quantum cryptosystem?*

First, we defined an example of the McEliece scheme using a known small optimal self-dual code of length 104. For this, we specified the algorithm of each step of the scheme. We discovered that a McEliece type cryptosystem using a self-dual code directly for a private key is vulnerable to structural attacks. Therefore, a punctured code of a self-dual code is used instead.

We proposed a different strategy for the decryption step: decryption using padded ciphertext and decoding via the complete self-dual code. As proof of concept, we implemented in SageMath code the thus defined McEliece encryption scheme with a private key - a punctured code of the small optimal self-dual code. Via cryptanalysis, we estimated that the classical bit security of the small McEliece encryption scheme is 22 bits. The public key of this system is around 28% smaller than the key size of the original McEliece encryption scheme of the same security level.

With this, we answered our first research question:
RQ1: *What is the security level of a McEliece type cryptosystem using a code obtained from an optimal binary [104,56,18] self-dual code? Define the encryption scheme.*

Next, using cryptanalysis, we estimated parameters for punctured codes obtained from optimal self-dual codes, which, if used in McEliece type system, would provide a bit security level of 80, 128 and 256. It is shown (Chapter 5) that the size of putative punctured codes providing bit security of 80, 128 and 256 bits is at least 38% smaller than the size of the proposed smallest Goppa codes[1] providing the same bit security levels. In particular, a punctured code of a putative $[1064, 532, d \geq 162]$ self-dual code would provide a bit security level of 80 when used as a private key of a McEliece type system. This completed the work on our second research question:
RQ2: *Which are the optimal parameters, length and error-correcting capability of a putative self-dual code that, if used in a McEliece type cryptosystem, would provide a bit security level of 80, 128 and 256, respectively?*

RQ3: *Construct an optimal self-dual code with parameters defined in RQ2, which can be used in a McEliece cryptosystem with a security of 80 bits.*

This question requires constructing an example of a binary $[1064, 532, d \geq 162]$ self-dual code. Then a McEliece cryptosystem using a punctured code derived from it would be secured up to the level of 80 bits.

To generate an example of such a code, we applied a known algorithm. It uses properties of self-dual codes possessing an automorphism of a specific type. In our case, we used a permutation of 8 cycles of length 133. Construction of a generator matrix of a self-dual code of length 1064 is derived in Section 1.2. A particular example of a $[1064, 532]$ self-dual code is also presented.

Due to computation time, we could not calculate the exact minimum weight of the example code. Instead, we completed a number of other computations discussed at the end of Section 1.2. Based on them, we expect that the constructed code has a minimum distance greater than or equal to 168.

RQ4: *Determine the algorithms in the steps of the McEliece type cryptosystem when using the code obtained in RQ3?*

The McEliece type cryptosystem has four steps described in Section 2.1. In the encryption scheme using the code of length 1 064, only the decryption step is not defined. More precisely, decoding the private key is not specified. The decoding algorithm implemented in the McEliece small example in RQ1 uses a complete set of codewords with a particular weight or a few such sets. For the self-dual code of length 1064 to find all the codewords with a certain weight is a computationally difficult problem. In addition, the set can be very large, i.e., it requires a large memory, which is a practical limitation for the usability of the system.

To create the missing decryption step, we developed a hard-decision iterative decoding algorithm. We proved that the decoding is suitable for a large group of self-dual codes, more precisely, self-dual codes with a specific structure. The code of length 1064 from RQ3 has such a structure.

We included three examples of decoding optimal self-dual codes with different parameters. The results of the experiments showed that the error-correcting performance of our new algorithm is beyond the upper bound for the error-correction capability ($t \leq (d-1)/2$, t maximum number of errors) of the code.

Further, we specified the missing decryption algorithm of the McEliece encryption scheme using a punctured code of the self-dual code of length 1064. The decryption is decryption with

[1] Goppa codes are the family codes used in the original McEliece cryptosystem.

padding with integrated iterative decoding for the self-dual code.

Note that in this case, the system has two private keys- the punctured code of the self-dual code and the self-dual code by itself. The reason is that the punctured code is used for creating the public key. It defines the parameters of the system: the length of the message, the length of the ciphertext and the number of intentionally added errors. The self-dual code is used for decoding in the decryption step via the new hard-decision iterative decoding.

Thus, we defined all the algorithms of the McEliece cryptosystem using the self-dual code of length 1064.

Note that the punctured code is not a self-dual code and has no specific structure. This code is the private key generating the corresponding public key of the cryptosystem. Then, also the public key has no specific structure. Thus, the public key and the punctured private key do not belong to any specific family of codes. Therefore, the problem of decoding the public key is expected to be as difficult as the problem of decoding a random code that is NP-complete (see Theorem 1.3). The full self-dual code is the second private key and is used for decoding.

To reverse engineer the process of creating the public key in order to retrieve the private key, one has to perform the following operations:

1. Guess the inverse of the permutation applied in the key generation;
2. Extend the punctured code to the complete self-dual code, which requires creating two columns and one row. This operation has a work factor of $2^{2k}2^n = 2^{2n}$. For our large code example, this is equal to 2^{2128}, which is far beyond the claimed security level of 80 bits.

To attack the system, an attacker can:

1. Search for a self-dual code of length 1064 with a generator matrix in the form derived in Section 1.2 (Eq.(6.6), (6.11), (6.14), (6.12)).
   One can calculate that the number of choices for the generator matrix in the form given in Section 1.2 is at least $16 * 2^3 + 4 * 16 * 2^{18}$, which is greater than 2^{24}.
2. Search for a relation between such a code and the public key.
   Finding a relation between a self-dual code of length 1 064 and the public key leads to searching for a permutation that maps 1 062 coordinates of the self-dual code into the public key. The number of choices is at least 1 032!, which is much bigger than the claimed security level of 80 bits.

With this, we finished the discussion on RQ4.

Summarising the results from all four research questions, we can conclude that self-dual codes with a large minimum weight can be used as a source for private keys of a McEliece type encryption scheme. The public key size is reduced by around 30%, comparable to the key size of the original McEliece scheme. Thus, the system is practically applicable due to the smaller public key and due to the new efficient decoding scheme integrated into the decryption. Also, the system is secure up to the claimed level according to the discussion on RQ4 and the results in Chapter 5.

Therefore, we concluded that:
A McEliece type cryptosystem using a code with a high error-correcting capability obtained from a self-dual code can be a secure and practically applicable post-quantum cryptosystem to a great extent.

2. Limitations

McEliece type cryptosystem using self-dual codes and punctured codes derived from them is proposed for the first time. Although the system is defined and analysed, there are limitations we

encountered during the project. An open question still is what is the exact minimum distance of the self-dual code of length 1064 generated in Section 1.1.

An exhaustive search for the existence of a codeword of weight d in a binary self-dual code requires calculating the weights of at most

$$2\sum_{i=1}^{d/2-1}\binom{k}{i}+\binom{k}{d/2}$$

linear combinations of basis vectors, where k is the dimension of the code. For the code of length 1 064 and expected minimum distance of 168, this number is greater than 2^{152}. We can consider the work factor of Stern's attack (Section 3.2) as another much better upper bound for the work factor of calculating the minimum weight of the code. It is at most 2^{87} (see Table 5.2).

Instead of calculating the exact minimum distance of the code, we could generate as large as possible sets of cyclically different codewords with weight d, $d+2$, $d+4$. Then, we could experiment with the error-correcting performance of different decoding sets obtained from these sets of codewords. If we find a decoding set such that a ciphertext with t errors is correctly decoded in more than 99% of the examples, then the self-dual code with the particular decoding set could be used in a McEliece type cryptosystem.

Finding suitable decoding set for the self-dual code of RQ3 is the other limitation in our work. Generating all codewords of weight d, $d+2$, $d+4$ takes a long computation time, but the decoding algorithm has the advantage that requires only a subset of these sets. Therefore, we consider this limitation as an open problem for future work.

3. Future work

The proposed McEliece type cryptosystem in this project is the first attempt to apply optimal self-dual codes and punctured codes derived from them in such a scheme. We have implemented a proof of concepts of the McEliece scheme for the small example using a code of length 104 and for the code of RQ3. Because the decoding set for the large code was very small, the implementation of the system could only decrypt ciphertext with up to 15 errors. Thus, the limitations discussed above evoke the following question for future work:

(a) Develop a method for generating new codewords of weight close to d from existing such codewords. For this, the particular structure of the code could help. The required new codewords must be cyclically different from the existing ones.

(b) Define an optimal decoding set of codewords of the self-dual code of length 1 064 from RQ3.

(c) Estimate the minimum weight of the code from RQ3 by its decoding performance using different decoding sets. For example, if there is a decoding set that decodes more than 99% of the encoded messages with 83 errors successfully, then we can assume that the minimum weight is $d \geq 167$. The number of examples included in the simulations should be relevant to the cardinality of the code.

Note that for a security level of 80 bits, the code must correct up to 77 errors, i.e. the minimum distance of 156 fulfils the requirement.

Finding a solution for (a) will optimise the computation time for the process of creating sets with codewords required for the problem (b). When an optimal decoding set of codewords is derived in (b), then the minimum weight of the code could be estimated (i.e. solving (c)), and even more, (b) provides the input data needed for the decryption of the cryptosystem.

Further, a programming implementation of the McEliece type cryptosystem with optimised computation time would be a logical continuation of the results in this project. The proof of concept code of the cryptosystem is now on SageMath. For a faster implementation, we recommend using $C++$ or *Rust*. From a security perspective for implementation on a real network, *Rust* is by default secure, while $C++$ requires much more tests to achieve the same.

The cryptosystem could be analysed via side-channel attacks when there are solutions for all of the above open problems, and a software implementation is developed.

We introduced a decryption algorithm (Algorithm 8) that integrates decoding a padded ciphertext via the full self-dual code. As a new algorithm, it requires further analysis for eventual vulnerabilities. This raises the next open research problem: Analyse the decryption Algorithm 8 for vulnerabilities.

A new decoding algorithm was developed specifically for self-dual codes of a certain type. It uses the cyclic structure of cells of the code and an orthogonality condition. QC codes also consist of cells with cyclic structure, which leads to the question of whether the algorithm can be adjusted for QC codes and, in particular, for QC-MDPC codes. This is an interesting question since QC-MDPC codes are used in the NIST submission *BIKE* [1]. If the answer is yes, the next open problem is to compare this algorithm to the existing efficient decoding algorithms for such codes.

References

[1] N. Aragon et al. BIKE: bit flipping key encapsulation. accessed 26 July 2021. 2021. URL: http://bikesuite.org.

[2] M Baldi and F. Chiaraluce. "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes". In: IEEE International Symposium on Information Theory (ISIT 2007). 2007, pp. 2591–2595.

[3] M. Baldi. QC-LDPC Code-Based Cryptography (5.4 Cryptanalysis of the McEliece and Niederreiter Cryptosystems). Springer, 2014.

[4] M. Bardet et al. "An Algebraic Attack on Rank Metric Code-Based Cryptosystems". In: Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III. Vol. 12107. Lecture Notes in Computer Science. Springer, 2020, pp. 64–93.

[5] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg. "On the inherent intractability of certain coding problems http://authors.library.caltech.edu/5607/1/BERieeetit78.pdf". In: IEEE Transactions on Information Theory 24.3 (1978), pp. 384–386.

[6] D. Bernstein et al. "Classic McEliece: conservative code-based cryptography". In: 2020. URL: https://classic.mceliece.org/nist/mceliece-20201010.pdf.

[7] D. J. Bernstein. "Grover vs. Mceliece". In: PQCrypto'10 (2010), pp. 73–80. URL: https://doi.org/10.1007/978-3-642-12929-2_6.

[8] D. J. Bernstein, T. Lange, and Ch. Peters. "Attacking and Defending the McEliece Cryptosystem". In: Post-Quantum Cryptography. Ed. by Johannes Buchmann and Jintai Ding. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–46. ISBN: 978-3-540-88403-3.

[9] D. J. Bernstein, T. Lange, and Ch. Peters. "Wild McEliece". In: Selected Areas in Cryptography. Springer Berlin Heidelberg, 2011, pp. 143–158.

[10] D. J. Bernstein, T. Lange, and Ch. Peters. "Wild McEliece Incognito". In: Post-Quantum Cryptography. Springer Berlin Heidelberg, 2011, pp. 244–254.

[11] M. Bossert. "On Decoding Using Codewords of the Dual Code". In: CoRR (Computing Research Repository) (Jan. 2020). URL: http://arxiv.org/abs/2001.02956.

[12] M. Bossert and F. Hergert. "Hard- and soft-decision decoding beyond the half minimum distance—An algorithm for linear codes (Corresp.)" In: IEEE Transactions on Information Theory 32.5 (1986), pp. 709–714.

[13] I. Bouyukliev and V. Bakoev. "A method for efficiently computing the number of codewords of fixed weights in linear codes". In: Discrete Applied Mathematics 156.15 (2008), pp. 2986–3004. DOI: https://doi.org/10.1016/j.dam.2008.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0166218X08000255.

[14] P.-L. Cayrel et al. "Critical attacks in code-based cryptography". In: International Journal of Information and Coding Theory 3 (Jan. 2015), pp. 158–176. DOI: 10.1504/IJICOT.2015.072639.

[15] A. Couvreur, M. Lequesne, and J. P. Tillich. “Recovering Short Secret Keys of RLCE in Polynomial Time”. In: Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019, Chongqing, China, May 8-10, 2019 Revised Selected Papers. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 133–152.

[16] A. Couvreur, A. Otmani, and J .P. Tillich. “Polynomial Time Attack on Wild McEliece over Quadratic Extensions”. In: Advances in Cryptology – EUROCRYPT 2014. Springer Berlin Heidelberg, 2014, pp. 17–39.

[17] A. Couvreur, A. Otmani, and J. P. Tillich. “Polynomial time attack on wild McEliece over quadratic extensions”. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2014, pp. 17–39.

[18] P. Domosi, C. Hannusch, and G. Horvàth. “A Cryptographic System Based on a New Class of Binary Error-Correcting Codes”. In: Tatra Mountains Mathematical Publications 73 (2019), pp. 83–96.

[19] R. Dontcheva, A. J. van Zanten, and S. M. Dodunekov. “Binary self-dual codes with automorphisms of composite order”. In: IEEE Trans.of Inf.Theory 50.2 (2004), pp. 311–318.

[20] S. T. Dougherty, T. A. Gulliver, and M. Harada. “Extremal binary self-dual codes”. In: IEEE Transactions on Information Theory 43.6 (1997), pp. 2036–2047. DOI: 10.1109/18.641574.

[21] D. Engelbert, R. Overbeck, and A. Schmidt. “A Summary of McEliece-Type Cryptosystems and their Security”. In: Journal of Mathematical Cryptology 1.2 (2007), pp. 151–199. DOI: doi:10.1515/JMC.2007.009.

[22] J. C. Faugère, L. Perret, and F. de Portzamparc. “Algebraic Attack against Variants of McEliece with Goppa Polynomial of a Special Form”. In: Advances in Cryptology – ASIACRYPT 2014. Springer Berlin Heidelberg, 2014, pp. 21–41.

[23] E. Fujisaki and T. Okamoto. “Secure Integration of Asymmetric and Symmetric Encryption Schemes”. In: Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34.

[24] E. M. Gabidulin. “On public-key cryptosystems based on linear codes”. In: Proc. of 4th IMA Conference on Cryptography and Coding 1993. IMA Press, Southend-on Sea, 1995, pp. 482–489.

[25] 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. Springer Berlin Heidelberg, 1991, pp. 482–489.

[26] P. Gaborit. “Shorter keys for code based cryptography”. In: International Workshop on Coding and Cryptography (WCC 2005). 2005, pp. 81–91.

[27] J. K. Gibson. “Severely denting the Gabidulin version of the McEliece Public Key Cryptosystem”. In: Designs, Codes and Cryptography 6.1 (1995), pp. 37–45.

[28] J. K. Gibson. “The Security of the Gabidulin Public Key Cryptosystem”. In: Advances in Cryptology — EUROCRYPT ’96. Springer Berlin Heidelberg, 1996, pp. 212–223.

[29] Y. Hamdaoui and N. Sendrier. A Non Asymptotic Analysis of Information Set Decoding. Cryptology ePrint Archive, Report 2013/162. https://eprint.iacr.org/2013/162. 2013.

[30] M. Harada et al. “New Binary Singly Even Self-Dual Codes”. In: IEEE Transactions on Information Theory 56.4 (2010), pp. 1612–1617.

[31] W. Huffman. "Automorphisms of codes with applications to extremal doubly even codes of length 48". In: IEEE Transactions on Information Theory 28.3 (1982), pp. 511–521. DOI: 10.1109/TIT.1982.1056499.

[32] J. Katz and Y. Lindel. Introduction to Modern Cryptography. Ed. by Douglas R. Stinson. CRC Press,Taylor & Francis Group, 2015.

[33] K. Kobara and H. Imai. "Semantically Secure McEliece Public-Key Cryptosystems - Conversions for McEliece PKC -". In: Public Key Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 19–35.

[34] G. Landais and J. P. Tillich. "An efficient attack of a McEliece cryptosystem variant based on convolutional codes". In: CoRR abs/1302.5120 (2013). URL: http://arxiv.org/abs/1302.5120.

[35] P. J. Lee and E. F. Brickell. "An Observation on the Security of McEliece's Public-Key Cryptosystem". In: Advances in Cryptology — EUROCRYPT '88. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 275–280.

[36] Y. Li, R. Deng, and X. Wang. "The equivalence of McEliece's and Niederreiter's public-key cryptosystems". In: IEEE Transactions on Information Theory 40 (1994), pp. 271–273.

[37] R. Lohnert. "Potential-decoding, error correction beyond the half minimum distance for linear block codes". In: Proceedings of 1995 IEEE International Symposium on Information Theory. 1995, pp. 52–.

[38] C. Löndahl and T. Johansons. "A new version of McEliece PKC based on convolutional code". In: Information and Communications Security (ICICS) LNCS 7168 (2012), pp. 461–470.

[39] F. J. Mac Williams and N. J. A. Sloane. The Theory of Error-Correcting Codes. Amsterdam, The Netherlands: Elsever, North - Holland, 1977.

[40] R. McEliece. "A public-key cryptosystem based on algebraic coding theory". In: (1978), p. 25. URL: https://ipnpr.jpl.nasa.gov/progress_report/42-44/44N.PDF.

[41] Carlos Aguilar Melchor et al. ROLLO-Rank-Ouroboros, LAKE & LOCKER. accessed 15 June 2021. 2019. URL: https://pqc-rollo.org.

[42] L. Minder and A. Shokrollahi. "Cryptanalysis of the Sidelnikov Cryptosystem". In: Advances in Cryptology - EUROCRYPT 2007. Springer Berlin Heidelberg, 2007, pp. 347–360.

[43] R. Misoczki et al. "MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes". In: IEEE International Symposium on Information Theory (ISIT 2013). 2013, pp. 2069–2073.

[44] H. Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory". In: Probl. Control and Inform. Theory (1986), pp. 159–166.

[45] NIST. Recommendation for Key Management. NIST. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r3.pdf. 2012.

[46] A. Otmani, J. P. Tillich, and L. Dallot. "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes". In: Mathematics in Computer Science 3.2 (2010), pp. 129–140.

[47] R. Overbeck and N. Sendrier. Code-based cryptography. Ed. by D.J. Bernstein, J. Buchmann, and E. Dahmen. Springer, 2009, pp. 95–145.

[48] Ch. Peters. "Information-Set Decoding for Linear Codes over Fq". In: Post-Quantum Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 81–94.

[49] V. Pless. "Decoding the Golay codes". In: IEEE Transactions on Information Theory 32.4 (1986), pp. 561–567.

[50] V. Pless. Introduction to the Theory of Error-Correcting Codes. New York: A Wiley - Interscience Publ, 1990.

[51] V. S. Pless and W. C. (Eds.) Huffman. Handbook of Coding Theory. Amsterdam, The Netherlands: Elsever, 1998.

[52] V.S. Pless and W.C. Huffman. Fundamentals of Error-Correcting Codes. Amsterdam, The Netherlands: Cambridge University Press, 2003.

[53] D Pointcheval. "Chosen-Ciphertext Security for Any One-Way Cryptosystem". In: Public Key Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 129–146.

[54] E. Prange. "The use of information sets in decoding cyclic codes". In: IRE Transactions on Information Theory vol. 8, no. 5, pp. 5-9 (1962).

[55] E. M. Rains. "Shadow bounds for self-dual codes". In: IEEE Trans. Inform. Theory 44 (1998), pp. 134–139.

[56] E. M. Rains and N. J. A. Sloane. Self-Dual Codes in Handbook of Coding Theory. Amsterdam, The Netherlands: Elsever, 1998, pp. 177–294. URL: arXiv:math/0208001.

[57] V. M. Sidelnikov. "A public-key cryptosystem based on binary Reed-Muller codes". In: Discrete Mathematics and Applications 4.3 (1994), pp. 191–208.

[58] V. M. Sidelnikov and S. O. Shestakov. "On insecurity of cryptosystems based on generalized Reed-Solomon codes". In: Discrete Mathematics and Applications 2.4 (1992), pp. 439–444.

[59] N. P. Smart. Cryptography Made Simple. Ed. by Springer. Springer International Publishing AG Switzerland, 2016.

[60] J. Stern. "A method for finding codewords of small weight". In: Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings. Vol. 388. Lecture Notes in Computer Science. Springer, 1988, pp. 106–113.

[61] S. de Vries. "A public-key cryptosystem based on algebraic coding theory". In: (2016). URL: http://essay.utwente.nl/70677/.

[62] Y. Wang. Quantum Resistant Public Key Encryption Scheme RLCE and IND-CCA2 Security for McEliece Schemes. Cryptology ePrint Archive, Report 2017/206. https://eprint.iacr.org/2017/206. 2017.

[63] Y. Wang. "Quantum resistant random linear code based public key encryption scheme RLCE". In: 2016 IEEE International Symposium on Information Theory (ISIT). 2016, pp. 2519–2523.

[64] J. Xing et al. "Iterative Decoding of Non-Binary Cyclic Codes Using Minimum-Weight Dual Codewords". In: 2020, pp. 333–337. DOI: 10.1109/ISIT44484.2020.9174258.

[65] V. Y. Yorgov. "Binary self-dual codes with automorphisms of odd order". In: Problems Inform. Transmission 19.4 (1983), pp. 260–270.

[66] R. Yorgova. "Binary self-dual extremal codes of length 92". In: 2006 IEEE International Symposium on Information Theory. 2006, pp. 1292–1295. DOI: 10.1109/ISIT.2006.262034.

[67] R. Yorgova. "On Binary Self-Dual Codes With Automorphisms". In: IEEE Trans.of Inf.Theory 54.7 (2008), pp. 3345–3351.

[68] S. Zhang. "On the nonexistence of extremal self-dual codes". In: Discrete Applied Math. 91 (1999), pp. 277–286.

A

Appendix

1. Algebraic Structures

In this section we present a sequence of definitions of algebraic structures which are used in Chapter 2 Preliminaries. We start with a list of properties regarding a set and two operations in it.

Let $(A,\ \circ,\ *)$ be a set A with defined two operations $\circ$ and $*$. One can define the following properties:

1. Operation $\circ$ is closed:
   $\forall a,\ b \in A \Rightarrow a \circ b \in A$.
2. Operation $\circ$ is associative:
   $\forall a,\ b,\ c \in A \Rightarrow (a \circ b) \circ c = a \circ (b \circ c)$.
3. A has an identity regarding $\circ$:
   $\exists\, e_1 \in A$ such that $\forall a \in A \Rightarrow a \circ e_1 = e_1 \circ a = a$.
4. every element in A has an inverse regarding $\circ$:
   $\forall a \in A\ \ \exists\, a' \in A$ such that $a \circ a' = a' \circ a = e_1$.
5. Operation $\circ$ is commutative:
   $\forall a,\ b \in A \Rightarrow a \circ b = b \circ a$.
6. Operation $*$ is closed:
   $\forall a,\ b \in A \Rightarrow a * b \in A$.
7. Operation $*$ is associative:
   $\forall a,\ b,\ c \in A \Rightarrow (a * b) * c = a * (b * c)$.
8. The operations $\circ$ and $*$ satisfy the distributive low:
   $\forall a,\ b,\ c \in A \Rightarrow a * (b \circ c) = a * b \circ a * c$.
9. A has an identity regarding $*$:
   $\exists\, e_2 \in A$ such that $\forall a \in A \Rightarrow a * e_2 = e_2 * a = a$.
10. Operation $*$ is commutative:
    $\forall a,\ b \in A \Rightarrow a * b = b * a$.
11. every element in A has an inverse regarding $*$:
    $\forall a \in A\ \ \exists\, a'' \in A$ such that $a \circ a'' = a'' \circ a = e_2$.

Definition 1.1. The set G with a defined operation $\circ$ is called a *group* if for $(G, \circ)$ the properties from 1 till 4 are satisfied, i.e. $\circ$ is a closed and associative operation, G has an identity element and each element is invertible.

If $\circ$ is commutative (property 5) then G is called an *abelian or commutative group*.

Definition 1.2. The set R with defined operations $\circ$ and $*$ is called a *ring* if $(R, \circ)$ is an abelian group, the operation $*$ is associative and both operations satisfy the distributive low. That is, $(R,\ \circ,\ *)$ is a ring if the properties from 1 till 8 are satisfied.

If in addition $*$ is commutative (property 10), R is called a *commutative ring* and when R has also an identity regarding $*$ (property 9), the ring is called a *commutative ring with unity*.

Definition 1.3. The set F with defined operations $\circ$ and $*$ is called a *field* if F is a commutative ring with unity where every element is invertible regarding $*$, i.e. $(F,\ \circ,\ *)$ is a field if all the properties, from 1 to 11, are satisfied.

A subset of a group is a *subgroup* if it forms a group regarding the same operation in the given group. That is, if $G_1 \subset G$ where $(G, \circ)$ is a group, then G_1 is a subgroup if $(G_1, \circ)$ is a group.

Definition 1.4. Let $(R,\ \circ,\ *)$ be a ring and $I \subset R$. I is called an *ideal* in R if $(I, \circ)$ is a subgroup of $(R,\ \circ)$ and $\forall i \in I, \forall r \in R \Rightarrow i * r \in I$.

An ideal consisting of all multiples of a fixed element is called a *principal ideal*, i.e. there exists $b \in I$ such that $I = \{rb \mid r \in \mathcal{R}\}$. The principal ideal I is denoted by $I = \langle b \rangle$ and b is called a *generator of* I. If a ring has no other ideals than principal ideals, it is called a *principal ideal ring*.

An ideal I is called a *prime ideal* if $ab \in I$ implies $a \in I$ or $b \in I$.

An ideal I in a ring $\mathcal{R}$ is called *maximal* if there is no ideal between I and R, i.e. for every ideal S, $I \subset S \subset R$, it follows that $S = I$ or $S = \mathcal{R}$.

Similarly, a *minimal ideal* I in $\mathcal{R}$ is a nonzero ideal containing no other nonzero ideal of $\mathcal{R}$. That is, for every ideal $S \subset I$ it follows that $S = 0$ or $S = I$.

Definition 1.5. Let $(V,\ +)$ be an abelian group, $(F,\ +,\ *)$ a field and let a multiplication $F \times V \to V$ satisfy the following properties:

1. $\forall a \in V \Rightarrow e_2 * a = a$, where e_2 is the identity elements of F regarding the second operation.
2. $\forall \alpha, \beta \in F$ and $\forall a \in V \Rightarrow (\alpha * \beta) * a = \alpha * (\beta * a)$.
3. $\forall \alpha \in F$ and $\forall a, b \in V \Rightarrow \alpha * (a + b) = \alpha * a + \alpha * b)$.
4. $\forall \alpha, \beta \in F$ and $\forall a \in V \Rightarrow (\alpha + \beta) * a = \alpha * a + \beta * a$.

Then the triple $(V,\ +,\ F)$ is called a *vector space* over the field F. The identity element of $(V,\ +)$ is denoted by $\boldsymbol{0}$.

2. Parameters of Punctured Codes Derived from Self-dual Codes for Bit Security 80, 128, and 256

In this section, we give work factors for the attacks $A_1, \ldots, A_5$. The results are listed in Table A.1.

Table A.1: $\min(Log_2(Workfactor))$ of the attacks $A_1, \ldots, A_5$ in Section 3.
The horizontal lines delimit 80, 128, and 256 bit security levels.
$A = \{A_1, ..., A_5\}$.

Punctured codes											
code	n	k	t	k(n-k)	$min(M)$	code	n	k	t	k(n-k)	$min(A)$
B_1	1 062	531	75	281 961	87.3248	B_{22}	1 924	962	136	925 444	149.9394
B_2	1 064	532	75	283 024	87.3264	B_{23}	1 926	963	136	927 369	149.9266
B_3	1 066	533	75	284 089	87.3118	B_{24}	1 928	964	136	929 296	149.9236
B_4	1 068	534	75	285 156	87.3136	B_{25}	1 930	965	136	931 225	149.9108
B_5	1 070	535	75	286 225	87.299	B_{26}	1 932	966	136	933 156	149.9078
B_6	1 072	536	75	287 296	87.3009	B_{27}	1 934	967	136	935 089	149.8952
B_7	1 074	537	75	288 369	87.2865	B_{28}	1 936	968	136	937 024	149.8922
B_8	1 076	538	75	289 444	87.2886	B_{29}	1 938	969	136	938 961	149.8796
B_9	1 894	947	134	896 809	147.8721	B_{30}	1 940	970	136	940 900	149.8767
B_{10}	1 896	948	134	898 704	147.869	B_{31}	4 006	2 003	284	4 012 009	303.9682
B_{11}	1 898	949	134	900 601	147.8561	B_{32}	4 008	2 004	284	4 016 016	303.9619
B_{12}	1 900	950	134	902 500	147.853	B_{33}	4 010	2 005	284	4 020 025	303.9509
B_{13}	1 902	951	134	904 401	147.8402	B_{34}	4 012	2 006	284	4 024 036	303.9446
B_{14}	1 904	952	134	906 304	147.8371	B_{35}	4 014	2 007	284	4 028 049	303.9336
B_{15}	1 906	953	134	908 209	147.8244	B_{36}	4 016	2 008	284	4 032 064	303.9273
B_{16}	1 908	954	134	910 116	147.8214	B_{37}	4 018	2 009	284	4 036 081	303.9163
B_{17}	1 910	955	134	912 025	147.8088	B_{38}	4 020	2 010	284	4 040 100	303.9101
B_{18}	1 912	956	134	913 936	147.8058	B_{39}	4 022	2 011	284	4 044 121	303.8991
B_{19}	1 918	959	136	919 681	149.9586	B_{40}	4 024	2 012	284	4 048 144	303.8929
B_{20}	1 920	960	136	921 600	149.9554	B_{41}	4 026	2 013	284	4 052 169	303.8819
B_{21}	1 922	961	136	923 521	149.9425	B_{42}	4 028	2 014	284	4 056 196	303.8758

3. Details on the construction of a binary $[1\,064, 532, d \geq 162]$ Self-dual Code

A generator matrix Y' of $\varphi(E_{\sigma_1}(B))$ is the following one:

$$Y' = \begin{pmatrix}
e_1(x) & 0 & 0 & 0 & 0 & g_1(x) & g_1(x) & g_1(x) \\
0 & e_1(x) & 0 & 0 & g_1(x) & 0 & g_1^2(x) & g_1^3(x) \\
0 & 0 & e_1(x) & 0 & g_1(x) & g_1^2(x) & 0 & g_1^2(x) \\
0 & 0 & 0 & e_1(x) & 0 & g_1^2(x) & g_1^3(x) & g_1^4(x) \\
0 & g_2(x) & g_2(x) & 0 & e_2(x) & 0 & 0 & 0 \\
g_2(x) & 0 & g_2^2(x) & g_2^2(x) & 0 & e_2(x) & 0 & 0 \\
g_2(x) & g_2^2(x) & 0 & g_2^3(x) & 0 & 0 & e_2(x) & 0 \\
g_2(x) & g_2^3(x) & g_2^2(x) & g_2^4(x) & 0 & 0 & 0 & e_2(x) \\
e_3(x) & 0 & 0 & 0 & 0 & g_3(x) & g_3^2(x) & g_3^3(x) \\
0 & e_3(x) & 0 & 0 & g_3^2(x) & 0 & g_3^3(x) & g_3^5(x) \\
0 & 0 & e_3(x) & 0 & g_3^7(x) & g_3^{13}(x) & 0 & g_3^{17}(x) \\
0 & 0 & 0 & e_3(x) & g_3^5(x) & g_3^{21}(x) & g_3^{23}(x) & 0 \\
0 & g_4^2(x) & g_4^7(x) & g_4^5(x) & e_4(x) & 0 & 0 & 0 \\
g_4(x) & 0 & g_4^{13}(x) & g_4^{21}(x) & 0 & e_4(x) & 0 & 0 \\
g_4^2(x) & g_4^3(x) & 0 & g_4^{23}(x) & 0 & 0 & e_4(x) & 0 \\
g_4^3(x) & g_4^5(x) & g_4^{17}(x) & 0 & 0 & 0 & 0 & e_4(x) \\
e_5(x) & 0 & 0 & 0 & 0 & g_5(x) & g_5^2(x) & g_5^3(x) \\
0 & e_5(x) & 0 & 0 & g_5^2(x) & 0 & g_5^3(x) & g_5^7(x) \\
0 & 0 & e_5(x) & 0 & g_5^5(x) & g_5^{11}(x) & g_5^{13}(x) & g_5^{17}(x) \\
0 & 0 & 0 & e_5(x) & g_5^7(x) & g_5^{21}(x) & g_5^{23}(x) & 0 \\
0 & g_6^2(x) & g_6^5(x) & g_6^7(x) & e_6(x) & 0 & 0 & 0 \\
g_6(x) & 0 & g_6^{11}(x) & g_6^{21}(x) & 0 & e_6(x) & 0 & 0 \\
g_6^2(x) & g_6^3(x) & g_6^{13}(x) & g_6^{23}(x) & 0 & 0 & e_6(x) & 0 \\
g_6^3(x) & g_6^7(x) & g_6^{17}(x) & 0 & 0 & 0 & 0 & e_6(x) \\
e_7(x) & 0 & 0 & 0 & 0 & g_7(x) & g_7^2(x) & g_7^7(x) \\
0 & e_7(x) & 0 & 0 & g_7^{13}(x) & 0 & g_7^{27}(x) & g_7^{31}(x) \\
0 & 0 & e_7(x) & 0 & g_7^3(x) & g_7^5(x) & 0 & g_7^{11}(x) \\
0 & 0 & 0 & e_7(x) & g_7^{17}(x) & g_7^7(x) & g_7(x) & 0 \\
0 & g_8^{13}(x) & g_8^3(x) & g_8^{17}(x) & e_8(x) & 0 & 0 & 0 \\
g_8(x) & 0 & g_8^5(x) & g_8^7(x) & 0 & e_8(x) & 0 & 0 \\
g_8^2(x) & g_8^{27}(x) & 0 & g_8(x) & 0 & 0 & e_8(x) & 0 \\
g_8^7(x) & g_8^{31}(x) & g_8^{11}(x) & 0 & 0 & 0 & 0 & e_8(x) \\
e_9(x) & 0 & 0 & 0 & g_9^{133}(x) & g_9(x) & g_9^{319}(x) & g_9^{233370}(x) \\
0 & e_9(x) & 0 & 0 & g_9^{266}(x) & g_9^2(x) & g_9(x) & g_9^{49}(x) \\
g_9^{68096}(x) & g_9^{136192}(x) & g_9^{1139}(x) & 0 & e_9(x) & 0 & 0 & 0 \\
g_9^{512}(x) & g_9^{1024}(x) & g_9^{149579}(x) & g_9^{338}(x) & 0 & e_9(x) & 0 & 0
\end{pmatrix}, \quad \text{(A.1)}$$

where the coefficients of the polynomials $e_i(x)$ and $g_i(x)$ for $i = 1, 2, \ldots, 9$ are given in Table A.2. Each of the entry polynomials in Y' generates a right circulant 3×133 matrix for the first 8 rows in Y' and a 18×133 right circulant matrix for the rest of 28 rows in Y'.

Table A.2: The coefficients of $a(x) = a_0x^0 + a_1x^1 + \cdots + a_{132}x^{132}$ in $\mathbb{F}_2[x]/(x^{133} - 1)$

Pol.	$(a_0, a_1, a_2, \ldots, a_{132})$
$e_1(x)$	11101001110100111010011101001110100111010011101001110100111010011101001110
	10011101001110100111010011101001110100111010011101001110100111010011101001110100
$g_1(x)$	10011101001110100111010011101001110100111010011101001110100111010011101001110100111010011101001
	11010011101001110100111010011101001110100111010011101001110100111010011101001110
$e_2(x)$	10010111001011100101110010111001011100101110010111001011100101110010111001011
	01110010111001011100101110010111001011100101110010111001011100101110010111001011
$g_2(x)$	10111001011100101110010111001011100101110010111001011100101110010111001011
	10010111001011100101110010111001011100101110010111001011100101110010111001011100
$e_3(x)$	00010011010111100110011110101001001110010111111111010110011001011100
	01011110101100010101111111010111000111100001110010011001010100000
$g_3(x)$	01111100010100000011011100100111100011111011101111111101111000101
	10110001001010110010000110000010101000010011001100100110010111010
$e_4(x)$	00000001010100110010011100001111000111010111111101010001101011110
	00011101001100110111111111101001110010010101111001100111101011001
$g_4(x)$	00101111010011001001100110010000101010000011000010011010100100011
	10101000111101111111011110111111000111100100111011000000101000111
$e_5(x)$	01111011100011111100010011101110101101000010000011101100101110011
	11010011101010100110000010100101110011111010011001011110100111010
$g_5(x)$	10011101110011011101010101010010100111010000100000100100001111001
	00100101010010110010111011010011011101011001001000010111001000001
$e_6(x)$	00010111001011110100110010111110011101001010000011001010101110010
	01110011101001101110000010000101101011101110010001111110001110111
$g_6(x)$	11100000100111010000100100110101110110010110111010011010010100100
	11001111000010010000010000101111001010010101010101110110011101100
$e_7(x)$	01101101101000101000110001011001110100001110010101100111000011101
	10110000001011011100101110110011011000110101111010101001111101101
$g_7(x)$	10101000010100000010011101110011110111010001111001111100010010011111
	00111011001111100000000011111101010001010001110111011111001111001
$e_8(x)$	01101101111100101010111101011000110110011011101001110110100000011101
	10111100001111001101010011100001011100110100011000101000101101101
$g_8(x)$	11001111001111110111011100010100010101111110000000001111100110111
	01111001001000111110011110001011101111001110111001000000101000010
$e_9(x)$	01111111111111111111011111111111111111111011111111111111111110111111111
	11111111101111111111111111111101111111111111111111011111111111111111
$g_9(x)$	10110110000101000101011011000010100010101101100001010001010110110000
	01010001010110110000101000101011011000010100010101101100001010010

4. Decoding Algorithm

Algorithm 9 gives the pseudocode of decoding self-dual codes with a specific structure defined in Section 1.1. Let G be a generator matrix of a self-dual code $\mathcal{C}$ with this specific structure and r be a given received vector.

Algorithm 9: Decoding self-dual codes having an automorphism

1 Generate the set D_1 of all or almost all cyclically different codewords of weight d or $d+o$ for some small o (2, 4, or 6)

2 Compute rG
if $rG = 0$ ($r \in \mathcal{C}$) **then** return r. Exit.
else 3).

3 Split r into t_1 polynomials of $\mathbb{F}_2[x]/(x^{pr}-1)$, i.e., $r = (r_1(x), r_2(x), \ldots, r_{t_1}(x))$

4 Compute:

- $w(x) = \langle r, b\rangle = r_1(x)b_1(x^{-1}) + r_2(x)b_2(x^{-1}) + \cdots + r_{t_1}(x)b_{t_1}(x^{-1}) \bmod (x^{pr}-1)$

  for $\forall\ b \in D_1$, $b = (b_1(x), b_2(x), \ldots, b_{t_1}(x))$

- $x^{\beta_i^{(1)}} w(x) \mod (x^{pr}-1)$ for $\forall\ \beta_i^{(1)} \in supp(b_1)$

- $x^{\beta_i^{(2)}} w(x) \mod (x^{pr}-1)$ for $\forall\ \beta_i^{(2)} \in supp(b_2)$

- $\vdots$

- $x^{\beta_i^{(t_1)}} w(x) \mod (x^{pr}-1)$ for $\forall\ \beta_i^{(t_1)} \in supp(b_{t_1})$

5 Compute $\Phi_j^{(s)}$ defined in Eq. (6.29) for $j = 0, 1, \cdots, pr-1,\ s = 1, 2, \ldots, t_1$.

6 Determine $\Phi_{max} = max\{\Phi_j^{(s)}\}$ and, the cycle s_1 and position(s) j_1 such that $\Phi_{j_1}^{(s_1)} = \Phi_{max}$

7 Compute $r_{s_1}(x) + x^{j_1}$.
$r_{s_1}(x) \leftarrow r_{s_1}(x) + x^{j_1}$

8 Repeat from 2) for the modified r.

5. Generator matrices of the examples in Section 2.3

This section provides generator matrices for the examples as discussed in Section 2.4.

5.1. Example 2.2

Let $\mathcal{D}_{90}$ be the binary $[90, 45, 14]$ self-dual code of Example 2.2. The generator matrix of $\mathcal{D}_{90}$ defined in Eq. (6.6) requires the matrices X and Y generating the subcodes $F_\phi(\mathcal{D}_{90})$ and $E_\phi(\mathcal{D}_{90})$, respectively.

For X a possible choice is

$$X = \begin{pmatrix} l & o & l & o & o & o \\ o & l & o & l & o & o \\ o & o & o & o & l & l \end{pmatrix},$$

where $l = (1, 1, \ldots, 1)$, $o = (0, 0, \ldots, 0)$, i.e., the all ones vector and the zero vector in $\mathbb{F}_2^{15}$.

The subcode $E_\phi(\mathcal{D}_{90})$ it generated via its image $\varphi(E_\phi(\mathcal{D}_{90}))$. A full description of how the subcode $\varphi(E_\phi(\mathcal{D}_{90}))$ is constructed is available in [66]. Here, we present one example of a generator matrix of $\varphi(E_\phi(\mathcal{D}_{90}))$, namely:

$$A' = \begin{pmatrix} e_1(x) & 0 & 0 & 0 & \mu_1(x) & \mu_1^2(x) \\ 0 & e_1(x) & 0 & \mu_1(x) & \mu_1(x) & \mu_1^{12}(x) \\ 0 & 0 & e_1(x) & \mu_1^2(x) & \mu_1^{12}(x) & \mu_1^8(x) \\ e_2(x) & 0 & 0 & 0 & \mu_2(x) & \mu_2(x) \\ 0 & e_2(x) & 0 & \mu_2(x) & \mu_2(x) & 0 \\ 0 & 0 & e_2(x) & \mu_2(x) & \mu_2(x) & \mu_2(x) \\ 0 & \mu_3(x) & \mu_3(x) & e_3(x) & 0 & 0 \\ \mu_3(x) & \mu_3(x) & \mu_3(x) & 0 & e_3(x) & 0 \\ \mu_3(x) & 0 & \mu_3(x) & 0 & 0 & e_3(x) \\ e_4(x) & 0 & 0 & 0 & 0 & \mu_4(x) \\ 0 & e_4(x) & 0 & 0 & \mu_4^2(x) & 0 \\ 0 & 0 & e_4(x) & \mu_4^2(x) & 0 & 0 \end{pmatrix},$$

where e_i, μ_i for $1 \leq i \leq 4$ are given in Table A.3. All the polynomials in A' are even weight polynomials of $\mathbb{F}_2[x]/(x^{15}-1)$.

Table A.3: Elements of $\mathbb{F}_2[x]/(x^{15}-1)$

e_1	$x^{14}+x^{13}+x^{12}+x^{11}+x^9+x^8+x^7+x^6+x^4+$ x^3+x^2+x
e_2	$x^{14}+x^{13}+x^{12}+x^{11}+x^9+x^7+x^6+x^3$
e_3	$x^{12}+x^9+x^8+x^6+x^4+x^3+x^2+x$
e_4	$x^{14}+x^{13}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x$
μ_1	$x^{11}+x^{10}+x^6+x^5+x+1$
μ_2	$x^{14}+x^{13}+x^{11}+x^9+x^8+x^5+x+1$
μ_3	$x^{14}+x^{10}+x^7+x^6+x^4+x^2+x+1$
μ_4	$x^{13}+x^{12}+x^{10}+x^9+x^7+x^6+x^4+x^3+x+1$

The corresponding generator matrix of the subcode $E_\phi(\mathcal{D}_{90})$ is

$$Y = \begin{pmatrix} y_{1,1} & \cdots & y_{1,6} \\ \vdots & \ddots & \vdots \\ y_{12,1} & \cdots & y_{12,6} \end{pmatrix},$$

where $y_{i,j}$ are right-circulant 4×15 cells for the first 9 rows in Y and $y_{i,j}$ are right-circulant 2×15 cells for the last 3 rows. The first rows of these circulant matrices are corresponding to the given polynomials in matrix A'.

5.2. Example 2.4

Let $\mathcal{T}$ be the binary $[78, 39, 14]$ self-dual code with an automorphism ϕ_1 of type $39-(2;\ 0)$ considered in Section 2.4.

The the matrices X and Y of the generator matrix of $\mathcal{T}$ defined in Eq. (6.6) are:

$$X = \begin{pmatrix} l & l \end{pmatrix},$$

where $l = (1, 1, \ldots, 1)$ is the all ones vector in $\mathbb{F}_2^{39}$ and

$$Y = \begin{pmatrix} y_{1,1} & y_{1,2} \\ y_{2,1} & y_{2,2} \\ y_{3,1} & y_{3,2} \\ y_{4,1} & y_{4,2} \end{pmatrix},$$

where $y_{i,j}$ are right-circulant 12×38 cells for the first 3 rows in Y and $y_{i,j}$ are right-circulant 2×39 cells for the last row. The first rows of these circulant matrices are given in Table A.4.

Table A.4: Coefficients of polynomials in $\mathbb{F}_2[x]/(x^{39} - 1)$

$y_{1,1}$	000100110100101101110101100111110111111
$y_{1,2}$	110011010100011111000101000011010111100
$y_{2,1}$	100111101011000010100011111000101011001
$y_{2,2}$	011111101111100110101110110100101100100
$y_{3,1}$	011010000010001101000001000110100000100
$y_{3,2}$	011111111111101111111111110111111111111
$y_{4,1}$	011011011011011011011011011011011011011
$y_{4,2}$	011011011011011011011011011011011011011

5.3. Example 2.5

We present details for the construction of the binary $[266, 133, 36]$ self-dual code B in Secytion 2.4. B has an automorphism ϕ_2 of type $133 - (2; 0)$. The generator matrix Y of $F_{\phi_2}(B)$ is

$$Y = \begin{pmatrix} y_{1,1} & y_{1,2} \\ \vdots & \vdots \\ y_{9,1} & y_{9,2} \end{pmatrix},$$

where $y_{i,j}$ are right-circulant 3×133 cells for the first 2 rows in Y and $y_{i,j}$ are right-circulant 18×133 cells for the next 7 rows. The first rows of these circulant matrices are given in Table A.5.

Table A.5: Coefficients of polynomials in $\mathbb{F}_2[x]/(x^{133} - 1)$

$y_{1,1}$	11101001110100111010011101001110100111010011101001110100111010011101001110 10011101001110100111010011101001110100111010011101001110100111010011101001110100
$y_{1,2}$	00000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000
$y_{2,1}$	00000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000
$y_{2,2}$	10010111001011100101110010111001011100101110010111001011100101110010111001 01110010111001011100101110010111001011100101110010111001011100101110010111001011
$y_{3,1}$	00010011010111100110011110101001001110010111111111011001100101110000 01011110101100010101111111010111000111100001110010011001010100000
$y_{3,2}$	00000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000
$y_{4,1}$	10011101110011011101010101010010100111101000100000100100001111001 1 00100101010010110010111011010011011101011001001000010111001000000 11

$y_{4,2}$	0000000101010011001001110000111100011101011111110101000110101111010 000111010011001101111111110100111001001010111100110011110101100100
$y_{5,1}$	01111011100011111100010011101110101101000010000011101100101110011101 10100111010101001100000101001011100111110100110010111101001110100
$y_{5,2}$	1001110111001101110101010101001010011110100001000001001000011110011 001001010100101100101110110100110111010110010010000101110010000011
$y_{6,1}$	1110000010011101000010010011010111011001011011101001101001010100100 110011110000100100000100001011110010100101010101011101100111011100
$y_{6,2}$	0001011100101111010011001011111001110100101000001100101010111001011 011100111010011011100000100001011010111011100100011111100011101111
$y_{7,1}$	0110110110100010100011000101100111010000111001010110011110000111101 101100000010110111001011101100110110001101011110101010011111011011
$y_{7,2}$	10001101110001010010011101010100000001000001010100111111101011011101 100101011001100000011111110010111101111110101011000011101011010111
$y_{8,1}$	11110101101011100001101010111111011110100111111100000001100110101001 10111011010111111100101010000001000000010101011100100101000111011000
$y_{8,2}$	0110110111110010101011110101100011011001101110100111011010000001101 101111000011110011010100111000010111001101000110001010001011011011
$y_{9,1}$	0111111111111111110111111111111111110111111111111111110111111111 111111111011111111111111111011111111111111111011111111111111111
$y_{9,2}$	1001101110000101110100110111000010111010011011100001011101001101110 000101110100110111000010111010011011100001011101001101110000101110