UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Markus Punnar

Cryptosystem for Post-Quantum Age Based on Moderate-Density Parity-Check (MDPC) Codes

Bachelor's Thesis (9 ECTS)

Supervisor: Vitaly Skachek, PhD
Co-supervisor: Irina Bocharova, PhD

Tartu 2020

Cryptosystem for Post-Quantum Age Based on Moderate-Density Parity-Check (MDPC) Codes

Abstract:

With the technology for quantum computers being actively developed by researchers worldwide, new methods for encrypting of sensitive data are needed. As a consequence of invention of Shor's algorithm, all cryptographic schemes based on finding the prime factors will become insecure, which include various asymmetric cryptosystems used today. The McEliece cryptosystem is based on the difficulty to distinguish structured linear codes from random linear codes. As it is believed to be immune to known attacks possible with a quantum computer, the McEliece cryptosystem is one of the main candidates for ensuring the confidentiality of sensitive data in a post-quantum environment. However, the construction of McEliece suffers from a large key size which makes using the scheme inefficient. There have been numerous variations to the original construction of the McEliece cryptosystem, but most of them have been proven to be insecure. One of the best candidates is the McEliece cryptosystem variation based on moderate density parity-check codes and its quasi-cyclic variant, which has not been successfully attacked while reducing the key size drastically. In this work, an overview of both the original construction of the McEliece cryptosystem and its modern variant is given, and iterative decoding algorithms used in decrypting messages in the cryptosystem are presented and analyzed.

Keywords:

Coding theory, McEliece cryptosystem, cryptography, decoding, iterative algorithms

CERCS: P175 Informatics, systems theory

Kvantjärgne krüptosüsteem mõõduka tihedusega paarsuskontrolli (MDPC) koodide baasil

Lühikokkuvõte:

Kuna kogu maailmas arendatakse aktiivselt kvantarvutite tehnoloogiat, on vaja uusi meetodeid tundlike andmete krüpteerimiseks. Shori algoritmi tõttu muutuvad kõik algtegurite leidmisel põhinevad krüptograafilised skeemid ebaturvaliseks, hõlmates erinevaid tänapäeval kasutatavaid asümmeetrilisi krüptosüsteeme. McEliece'i krüptosüsteem põhineb struktureeritud lineaarsete koodide eristamatusel juhuslikest lineaarsetest koodidest. Kuna usutakse, et McEliece skeem on teadaolevate kvantarvutirünnakute suhtes immuunne, on see üks peamisi kandidaate andmete konfidentsiaalsuse tagamiseks kvantjärgses keskkonnas. McEliece'i skeemi peamiseks puudujäägiks on suur võtme suurus, mis muudab skeemi kasutamise ebaefektiivseks. McEliece krüptosüsteemi algsest konstruktsioonist on välja töötatud mitmeid variatsioone, kuid enamik neist on osutunud ebaturvaliseks. Üks arimatest kandidaatidest on McEliece'i variatsioon, mis põhineb mõõduka tihedusega paarsuskontrollkoodidel ja selle kvartsüklilisel variandil, mida pole edukalt rünnatud ning mis vähendab märgatavalt võtme suurust. Selles töös antakse ülevaade nii McEliece krüptosüsteemi algsest konstruktsioonist kui ka selle kaasaegsest variandist ning esitatakse ja analüüsitakse krüpteeritud sõnumite dekrüpteerimisel kasutatavaid iteratiivseid dekodeerimise algoritme.

Võtmesõnad:

Kodeerimisteooria, McEliece krüptosüsteem, krüptograafia, dekodeerimine, iteratiivsed algoritmid

CERCS: P175 Informaatika, süsteemiteooria

Table of contents

Glossary

Notations

- $\mathbb{F}_q$ - A finite field with q elements
- $\mathcal{C}$ - A (linear) code
- $\mathtt{d}(\mathbf{x}, \mathbf{y})$ - The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$
- d_{min} - Minimum distance of the code
- G - A generator matrix of the code
- H - A parity-check matrix of the code
- P - A permutation matrix

Abbreviations

- LDPC - Low-density parity-check
- MDPC - Moderate-density parity-check
- QC-LDPC - Quasi-cyclic low-density parity-check
- QC-MDPC - Quasi-cyclic moderate-density parity-check
- DFR - Decoding failure rate
- BG - the Black-Gray
- CBBF - Candidate based bit flipping
- UPC - Unsatisfied parity check

Introduction

The possibility of building a quantum computer has been talked about since the beginning of 1980s when Paul Benioff published his paper on a quantum model of the classical Turing machine [1]. Over the next few decades it became clear that the computing power of a quantum computer is magnitudes larger than that of a classical computer. One of the most important results in the field of quantum computers is an algorithm developed by Peter Shor which is able to find the factors of large composite numbers in polynomial time [2]. Since the time when Shor published his algorithm in 1994, very active research in the field of quantum computers has been conducted worldwide and there has been a rapid progress.

Shor's results imply that cryptosystems based on factoring of large integers or on a discrete logarithm problem become not secure when quantum computers become practical. Examples of such cryptosystems are the RSA cryptosystem [3], ECC [4] and the Diffie-Hellman key exchange [5], which are all widely used today. Since powerful quantum computers may become the reality over the next few decades, active research has been initiated all over the world to develop cryptosystems for the post-quantum age. There are multiple different approaches to this problem but one of the most promising is a cryptosystem based on linear codes. The first code-based cryptosystem was constructed by Robert McEliece in 1978 [6]. The original construction of the cryptosystem is still not broken even with the use of quantum technology, but the main problem of the scheme is a large key size, which makes this scheme not practical. It is an important research objective to find ways to make the key smaller without compromising the security of the cryptosystem. Many different kinds of code families have been suggested over the years, but most of them have been shown to be insecure. However, quasi-cyclic moderate density parity-check (QC-MDPC) codes is one of the promising candidates

for post-quantum cryptographic applications. The focus of this work is the analysis of the use of the QC-MDPC codes in the McEliece cryptosystem, an overview of the known results and improvements to the efficiency of iterative decoding methods for the MDPC codes.

The work is divided into four chapters. In the first chapter, an overview is given about the preliminary coding theory concepts which are needed to understand the theory behind the McEliece cryptosystem. A description of the original construction by McEliece is presented next. In the second chapter, some weaknesses of the original construction are analyzed. The QC-MDPC codes and their suitability for the McEliece cryptosystem are discussed. Overview of iterative decoding algorithms and their variations, which are used to decode the QC-MDPC codes, is given in the third chapter. Optimizations for implementation and parameters for the algorithms along with decoding simulation results are presented in the final chapter.

1 Preliminaries

In this chapter, an overview is given of basic concepts and results which are needed to understand the construction of the McEliece cryptosystem. Additionally, a brief introduction into the original construction by Robert McEliece is presented. This construction is based on binary Goppa codes which is still believed to be secure against quantum computers.

1.1 Linear codes

Definition 1 *Let $\mathbb{F}$ be a finite field. A linear $[n, k]$-code $\mathcal{C}$ over $\mathbb{F}$ is a subspace of a vector space $\mathbb{F}^n$ of dimension k. In other words, for every two codewords $\mathbf{c_1}, \mathbf{c_2} \in \mathcal{C}$ and scalars $a_1, a_2 \in \mathbb{F}$ we have $a_1\mathbf{c_1} + a_2\mathbf{c_2} \in \mathcal{C}$.*

In the sequel, we consider only linear codes over the field $\mathbb{F} = \mathbb{F}_2$. The codewords of such codes can be viewed as vectors of bits. The following definitions assume that we are working with binary codes.

Definition 2 *Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ be vectors of length n. The Hamming distance $\mathsf{d}(\mathbf{x}, \mathbf{y})$ is defined as the number of coordinates which are pairwise different in $\mathbf{x}$ and $\mathbf{y}$. Hence, we have*

$$\mathsf{d}(\mathbf{x}, \mathbf{y}) := |\{i : x_i \neq y_i\}|$$

In addition to the Hamming distance we will note that the Hamming weight of a codeword $\mathbf{c} \in \mathcal{C}$ is defined as the Hamming distance between $\mathbf{c}$ and the all-zero vector of length n. Formally, the Hamming weight of a codeword is denoted as $w(\mathbf{c}) := \mathsf{d}(\mathbf{0}, \mathbf{c})$, where $\mathbf{0}$ denotes the vector of zeroes.

Definition 3 *Let $\mathcal{C}$ be a linear $[n,k]$-code. The minimum distance of $\mathcal{C}$ is denoted as d_{min} and is defined as*

$$d_{min} := \min_{\substack{\mathbf{c_1} \neq \mathbf{c_2} \\ \mathbf{c_1}, \mathbf{c_2} \in \mathcal{C}}} \mathtt{d}(\mathbf{c_1}, \mathbf{c_2}).$$

It is easy to see that the minimum distance of a linear code $\mathcal{C}$ is equal to the minimum weight of any non-zero codeword of $\mathcal{C}$. If $\mathcal{C}$ is a linear $[n,k]$-code of minimum distance d_{min}, then we call it an $[n,k,d_{min}]$-linear code. Linear codes are used to encode information vectors $\mathbf{x} \in \mathbb{F}^k$ to codewords $\mathbf{c} \in \mathcal{C} \subset \mathbb{F}^n$. Since we are considering binary codes, then the information vectors can be viewed as binary representation of the plaintext. Additionally, we have $|\mathcal{C}| = 2^k$. Since the dimension of $\mathcal{C}$ is k, it follows that every basis of the code contains k vectors which span the entire vector space. Every vector in the vector space can be written as a linear combination of the basis vectors and there are 2^k different linear combinations. By using the basis vectors we can define a one-to-one mapping between the information words and codewords.

Definition 4 *Let $\mathcal{C}$ be a linear $[n,k,d_{min}]$-code over $\mathbb{F}$. A generator matrix G of $\mathcal{C}$ is a $k \times n$ matrix with entries in $\mathbb{F}$ whose rows form a basis of the code.*

Usually a code has many different generator matrices since generally a vector space has multiple sets of basis vectors. By using the generator matrix we define a mapping $\varphi : \mathbb{F}^k \to \mathcal{C}$ such that

$$\mathbf{x} \mapsto \mathbf{x} \cdot G =: \mathbf{c}.$$

Since the rows of G form a basis of the code, they are linearly independent and hence the mapping is one-to-one. It is convenient to use row operations on G to obtain the matrix in a systematic form. In that case the generator matrix is of the form

$$G = [I_k \mid Q],$$

where I_k is a $k \times k$ identity matrix and Q is a $k \times (n-k)$ matrix. A generator matrix which has this form is called a systematic generator matrix. The encoding using a systematic generator matrix is simplified, as it becomes

$$\mathbf{x} \mapsto \mathbf{x} \cdot G = (\mathbf{x} \mid \mathbf{x} \cdot Q).$$

A generator matrix is one way to describe the code $\mathcal{C}$. Another way to describe a code is through the parity-check matrix of the code.

Definition 5 *Let $\mathcal{C}$ be a linear $[n, k, d_{min}]$-code over $\mathbb{F}$. A parity-check matrix H of $\mathcal{C}$ is an $(n-k) \times n$ matrix with entries in $\mathbb{F}$ such that for every $\mathbf{c} \in \mathbb{F}^n$*

$$\mathbf{c} \in \mathcal{C} \iff H \cdot \mathbf{c}^T = \mathbf{0}.$$

One of the key properties of linear codes is their error-correcting capability. This means that it is possible to restore the original information even if the communication channel has introduced errors in the received vector.

Definition 6 *Let $\mathcal{C}$ be a linear $[n, k, d_{min}]$-code over $\mathbb{F}$ whose generator matrix is G. Code $\mathcal{C}$ can correct up to t errors if there exists a decoding algorithm $\mathcal{D} : \mathbb{F}^n \to \mathcal{C}$ such that for every information vector $\mathbf{u} \in \mathbb{F}^k$ and for every error vector $\mathbf{e} \in \mathbb{F}^n$, $w(\mathbf{e}) \leq t$, the vector*

$$\mathbf{y} = \mathbf{u} \cdot G + \mathbf{e}$$

is always decoded correctly as $\mathcal{D}(\mathbf{y}) = \mathbf{u}$.

Definition 7 *Let $\mathcal{C}$ be a linear $[n, k, d_{min}]$-code over $\mathbb{F}$ whose parity-check matrix is H. Let $\mathbf{y} \in \mathbb{F}^n$. The syndrome of $\mathbf{y}$ is denoted with $\mathbf{s}$ and defined as*

$$\mathbf{s} = H \cdot \mathbf{y}^T.$$

It is clear from this definition that codewords are the vectors $\mathbf{c} \in \mathbb{F}^n$ whose syndrome is equal to the zero vector. If a vector contains non-zero elements, then these are referred to as failed parity-checks. It turns out that the error-correcting capability of a code is related to the minimum distance of the code. It can be shown that the following theorem holds [7, Proposition 1.3].

Theorem 1 *Let $\mathcal{C}$ be a linear $[n, k, d_{min}]$-code over $\mathbb{F}$. There exists a decoding algorithm $\mathcal{D} : \mathbb{F}^n \to \mathcal{C}$ that correctly decodes codewords with up to $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ errors.*

In the following chapters, the notion of the density of matrices is important. The density of a binary matrix refers to the number of ones in the matrix compared to the number of zeroes. When the number of ones grows larger, a matrix becomes more dense. If a matrix has very few entries which are ones, the matrix is called sparse.

1.2 Binary Goppa codes

To understand the original construction of the McEliece cryptosystem, binary Goppa codes are described first, which were defined by V. D. Goppa in 1970 [8]. A polynomial is called monic in which the leading coefficient is equal to one.

Definition 8 [9] *Let n, m and t be positive integers and let*

$$g(X) := \sum_{i=0}^{t} g_i X^i \in \mathbb{F}_{2^m}[X]$$

be a monic polynomial of degree t. Let $L = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{2^m}^n$ be a tuple of n distinct elements in $\mathbb{F}_{2^m}$ such that

$$g(\alpha_i) \neq 0, \quad \forall i \in \{1, \ldots, n\}.$$

The Goppa code $\mathcal{G} = \mathcal{G}(\alpha_1, \ldots, \alpha_n, g(X))$ consists of all elements $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{F}_2^n$ that satisfy

$$\sum_{i=1}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \mod g(X).$$

When the polynomial $g(X)$ is irreducible, then $g(\alpha) \neq 0$ for all elements $\alpha \in \mathbb{F}_{2^m}$. For such polynomials, the elements of L can be chosen uniformly from all elements of $\mathbb{F}_{2^m}$. Goppa codes in which the polynomial $g(X)$ is irreducible are called irreducible Goppa codes. For irreducible Goppa codes it can be shown that $k \geq n - tm$ and for cryptographic application, an equality is assumed. It can also be shown that the Hamming weight of every non-zero codeword in a Goppa code $\mathcal{G}$ is at least $2t + 1$, and therefore the minimum distance of $\mathcal{G}$ is $d_{min} \geq 2t+1$ [10]. From Theorem 1 it is known that there exists a decoder which corrects up to $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ errors, therefore there exists a decoder for irreducible Goppa codes which corrects up to t errors. Such an algorithm was published by Patterson in 1975, which corrects all t errors in polynomial time [11]. Thus, an irreducible polynomial $g(X) \in \mathbb{F}_{2^m}[X]$ of degree t and a chosen value $n \leq 2^m$ define an $[n, n - tm, 2t + 1]$ binary Goppa code which can correct up to t errors.

1.3 Original construction of the McEliece cryptosystem

The McEliece cryptosystem makes use of the error-correcting capabilities of linear codes for encrypting the messages. The underlying idea is that a sufficiently randomized version of the code is hard to distinguish from a random linear code. The cryptosystem uses this randomized version of the code as a public key whereas the secret key gives information about the structure of the particular linear code. Given an efficient decoding algorithm exists for the chosen linear code, the knowledge of the structure provides information for fast decryption, but decrypting without any knowledge of the structure is hard. In fact, it is known that decoding a general linear code is an $\mathcal{NP}$-hard problem [12].

Definition 9 *A permutation matrix P is an $n \times n$ binary matrix whose every column and every row contains a single one and all other elements are zeroes.*

Multiplying any $k \times n$ matrix A by a permutation matrix on the right results in a $k \times n$ matrix B which contains the same columns as A, but in a permuted order. In order to define the McEliece cryptosystem, it is needed to define key generation, encryption and decryption processes.

1.3.1 Key generation

The steps for the key generation in the McEliece cryptosystem are as follows:

1. Choose a linear $[n, k, 2t + 1]$-code $\mathcal{C}$ over $\mathbb{F}_2$, for which an efficient decoding algorithm $\mathcal{D}$ that can correct up to t errors exists.

2. Compute a $k \times n$ generator matrix G for $\mathcal{C}$.

3. Generate a random $k \times k$ binary invertible matrix S.

4. Generate a random $n \times n$ permutation matrix P.

5. Compute the $k \times n$ matrix $G' = S \cdot G \cdot P$.

The public key is the pair (G', t) and the private key is $(S, G, P, \mathcal{D})$. Note that the public key G' describes an equivalent code to G because P permutes the columns of G and S switches to a different basis of the same code. McEliece originally used irreducible binary Goppa codes, for which an efficient decoding algorithm was introduced by Patterson [11]. Theoretically any linear code for which an efficient decoding algorithm exists could be used instead of Goppa codes, but most of such attempts are shown insecure [13]. Problems with security and key size when using different code families are analyzed in Section 2.2.

1.3.2 Encryption and decryption

To encrypt a plaintext $\mathbf{m} \in \mathbb{F}_2^k$, generate a random vector $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight t and compute the ciphertext as

$$\mathbf{c} = \mathbf{m} \cdot G' + \mathbf{e}.$$

To decrypt a ciphertext $\mathbf{c} \in \mathbb{F}_2^n$:

1. Compute $\mathbf{c} \cdot P^{-1} = (\mathbf{m} \cdot S) \cdot G + \mathbf{e} \cdot P^{-1}$.

2. Since $(\mathbf{m} \cdot S) \cdot G$ is a codeword for the chosen linear code, and $\mathbf{e} \cdot P^{-1}$ has Hamming weight t, the decoding algorithm $\mathcal{D}$ can be applied to $\mathbf{c} \cdot P^{-1}$ to obtain $\mathbf{c}' = \mathbf{m} \cdot S$.

3. Obtain the plaintext $\mathbf{m}$ as $\mathbf{m} = \mathbf{c}' \cdot S^{-1}$.

2 Background on the underlying problem

As it is mentioned in the definition of the key generation process, the McEliece cryptosystem can use any linear $[n, k, 2t+1]$-code for which an efficient decoding algorithm exists. However, most code families are not suitable for use in the McEliece cryptosystem. In this chapter, an overview of different options and their shortcomings is given. The reasons why low density parity-check (LDPC) and moderate density parity-check (MDPC) codes and their quasi-cyclic variants are feasible candidates to use in the McEliece cryptosystem are also discussed.

2.1 Quasi-cyclic LDPC/MDPC codes

Definition 10 *A linear $[n, k, d]$-code $\mathcal{C}$ is called quasi-cyclic if there exists some integer $n_0 \in \mathbb{N}$ such that every cyclic shift of a codeword by n_0 places is a codeword in $\mathcal{C}$.*

If $n = n_0 \cdot p$, then it is possible to represent both generator and parity-check matrices of the code by $p \times p$ circulant blocks. Each circulant can be fully described by its first row which results in much smaller key sizes for the McEliece cryptosystem as only one row is needed to describe the matrix used in the scheme.

Definition 11 *An $[n, k, w]$-LDPC code is a linear code with size n and dimension k, which has a parity-check matrix with constant row weight w.*

LDPC and MDPC codes differ only by the weight of the rows in the parity-check matrix. For LDPC codes the row weight is a small constant - typically less than 10. For MDPC codes the row weight typically scales in $O(\sqrt{n \log n})$. When the codes are also quasi-cyclic, they are called QC-LDPC and QC-MDPC codes.

2.2 Limitations of the cryptosystem based on the code family

The security of the McEliece cryptosystem is based on the so-called decoding assumption, namely that the decoding of a general linear code is hard and the code family cannot be easily determined by analyzing the matrices that describe the code. The decoding assumption has been thoroughly studied and is believed to be a hard problem [12]. The original construction of McEliece is secure, but it suffers from a very large key size. McEliece originally suggested parameters $n = 1024$, $k = 524$, $t = 50$, but these parameters have been broken on modern hardware. Recent analysis suggests parameters $n = 2048$, $k = 1751$, $t = 27$ for 80-bit security using standard algebraic decoding, but this results in the public key size of 520,047 bits [14]. For security against quantum computers, the current suggested parameters are $n = 6960$, $k = 5413$, $t = 119$, which result in a public key over 8 megabytes in size and makes using the scheme implausible for real-life applications [15].

One way to reduce the problem of key size is to use quasi-cyclic codes. Using a quasi-cyclic code with strong algebraic structure results in the system being vulnerable to an algebraic attack. Combining the quasi-cyclic property with the aforementioned algebraic structure allows the attacker to set up a system of linear equations which can be solved. Using a code family which does not have algebraic structure would completely remove this threat.

LDPC and MDPC codes are good candidates for such purposes. The sparsity of the matrix is used in iterative decoding techniques to ensure sufficient error-correcting capabilities. LDPC codes have been repeatedly suggested for use in the McEliece cryptosystem [16]. However, the main problem with using LDPC codes is that the rows of the parity-check matrix can be seen as low weight codewords in the dual of the public code. This fact can be used to construct an attack against the system by finding low weight codewords in the dual code and constructing a sparse parity-check matrix [17]. MDPC codes are more resilient to such attacks due to their parity-check matrix

being less sparse. However, this results in iterative decoding techniques becoming less efficient and in an increase in decoding failure frequency for these algorithms.

2.3 Construction of a QC-MDPC code

For cryptographic purposes $[n, k, w]$-MDPC codes where $n = n_0 \cdot p$ and $r := n - k = p$ are used. This means that the parity-check matrix H can be represented in the form

$$H = [H_0 \mid H_1 \mid \cdots \mid H_{n_0-1}]$$

where H_i is a $p \times p$ circulant block, $i = \{0, 1, \ldots, n_0 - 1\}$. The first row of H is defined by picking random vectors of length p and weight w for the first rows in each circulant. The other $r - 1$ rows of H are acquired from the $r - 1$ quasi-cyclic shifts of the first row in every circulant. The total weight of the every row in the matrix is therefore $w \cdot n_0$. A generator matrix G can be easily derived from the blocks of H. Assuming that H_{n_0-1} is invertible G can be computed as

$$G = [I_k \mid Q],$$

where I_k is a $k \times k$ identity matrix and

$$Q = \begin{bmatrix} (H_{n_0-1}^{-1} H_0)^T \\ (H_{n_0-1}^{-1} H_1)^T \\ \vdots \\ (H_{n_0-1}^{-1} H_{n_0-2})^T \end{bmatrix}.$$

2.4 QC-MDPC variant of the McEliece cryptosystem

A major benefit of using QC-MDPC codes in the McEliece scheme is that the permutation matrix P and scrambling matrix S are no longer needed. This further reduces the

key size as both the public and private keys can be represented as a single row or a column of the generator matrix G and parity-check matrix H, respectively. The QC-MDPC McEliece variant works as follows:

2.4.1 Key generation

1. Generate a $(n-k) \times n$ parity-check matrix H.
2. Generate the corresponding $k \times n$ generator matrix G in its systematic form.

The public key is now the pair: column $k+1$ of G and t, which is the number of introduced errors. The private key is the first row of H.

2.4.2 Encryption

To encrypt a plaintext $\mathbf{m} \in \mathbb{F}_2^k$ into $\mathbf{x} \in \mathbb{F}_2^n$:

1. Generate a vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t.
2. Compute $\mathbf{x}$ as $\mathbf{x} = \mathbf{m} \cdot G + \mathbf{e}$.

2.4.3 Decryption

Let $\mathcal{D}$ be an efficient decoding algorithm for MDPC codes. Then, in order to decrypt $\mathbf{x}$ into $\mathbf{m}$:

1. Apply $\mathcal{D}$ to $\mathbf{x} = \mathbf{m} \cdot G + \mathbf{e}$ to obtain $\mathbf{m} \cdot G$.
2. Extract $\mathbf{m}$ from the first k positions of $\mathbf{m} \cdot G$.

It is guaranteed that the first k positions of $\mathbf{m} \cdot G$ are $\mathbf{m}$ itself because the generator matrix G is presented in a systematic form. For MDPC codes, the decoding algorithm $\mathcal{D}$ is an iterative decoder originally constructed for decoding of LDPC codes. It relies on the knowledge of H and the fact that H is a sparse matrix. The parity-check matrix

of an MDPC code is not as sparse as a matrix for an LDPC code which results in a larger decoding failure rate. Minimizing this rate is the main research objective in this direction.

3 Iterative decoding algorithms

When compared to the encryption in the McEliece cryptosystem, the decryption is a considerably more complex operation. In this chapter, an overview of iterative decoding methods for MDPC codes is given. Some known variations with improvements on the original decoding algorithm are discussed and analyzed.

3.1 Different algorithm families

This work focuses on the family of algorithms originally constructed by Gallager in 1962 [18]. This family of algorithms is called bit-flipping algorithms. Bit-flipping algorithms use the information from the parity-check matrix to make decisions whether a bit is in error or not. Usually algorithms use hard decisions, but there are variants of soft decision bit-flipping decoders as well, yet they are out of the scope of this work.

3.2 Original bit-flipping algorithm

By the definition of the parity-check matrix, each codeword can be viewed as a solution to a homogeneous system of linear equations defined by the parity-check matrix. Since during the encryption an error-vector is added to the codeword, then with overwhelming probability the encrypted ciphertext vector is not a codeword. Therefore the syndrome of the ciphertext is a non-zero vector. Each bit of the syndrome is calculated with a parity-check equation, in general the i'th bit in the syndrome is calculated by multiplying the i'th row of the parity-check matrix with the ciphertext vector. If that bit is a non-zero, then the parity-check is unsatisfied for the bits located at the positions of non-zero elements of the i'th row of the parity-check matrix. Bit-flipping algorithm uses the idea of counting the number of unsatisfied parity-checks for each ciphertext bit. This information is later used to decide whether that bit should be flipped or not. Flipping a bit is interpreted as $y = y \oplus 1$ for a bit y where $\oplus$ denotes the exclusive OR operation.

The original bit-flipping algorithm proposed by Gallager is presented in Figure 1.

Input : ciphertext $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, maximum number of iterations I_{max}.
$I = 0$, $\mathbf{s} = H \cdot \mathbf{y}^T$
while $I < I_{max}$ **and** $\mathbf{s} \neq \mathbf{0}$ **do**
    Compute number of UPCs for each bit in $\mathbf{y}$
    $b = calculateThreshold()$
    for $i = 1 : n$ **do**
        if $UPC_i > b$ **then**
            Flip y_i
        end
    end
    $\mathbf{s} = H \cdot \mathbf{y}^T$
    $I = I + 1$
end
Output: Decoded vector $\mathbf{y}$ and syndrome $\mathbf{s}$.

Figure 1. Original bit-flipping algorithm

The algorithm works as follows: first the syndrome of the ciphertext is calculated and the number of unsatisfied parity-checks (UPCs) is counted for each ciphertext bit. Then the threshold b for flipping the bits is calculated in function *calculateThreshold()* during every iteration. After that the bits with more unsatisfied parity-checks than the threshold are flipped and the syndrome is recalculated. Different methods on how to choose the threshold b have been suggested over the years. Gallager calculated the threshold based on the parameters of the code. In variations of the bit-flipping algorithm b can also be set to the maximum number of unsatisfied parity checks [19]. The option of having $b = \max\{\#_{UPC}\} - \delta$ for some small delta has also been analyzed [20]. Here $\max\{\#_{UPC}\}$ denotes the maximum number of unsatisfied parity checks for a ciphertext bit. For this work the second option of choosing $b = \max\{\#_{UPC}\}$ was chosen.

3.3 Optimizations of the original decoder

In addition to modifying the threshold for deciding whether to flip a bit, another optimization for accelerating the recalculation of the syndrome is base on the observation that the syndrome does not change arbitrarily when a bit is flipped. The new syndrome after flipping the i'th bit in the ciphertext is equal to the old syndrome to which the i'th column of the parity-check matrix has been added [21]. Furthermore, since the objective is to make the syndrome equal to the all-zero vector, the bit is flipped only if the Hamming weight of the new syndrome is smaller than that of the old syndrome. With these optimizations, the modified bit-flipping algorithm is presented in Figure 2 [21].

Input : ciphertext $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, maximum number of iterations I_{max}.
$I = 0$, $\mathbf{s} = H \cdot \mathbf{y}^T$
while $I < I_{max}$ **and** $\mathbf{s} \neq \mathbf{0}$ **do**
    Compute number of UPCs for each bit in $\mathbf{y}$
    $b = \max\{\#_{UPC}\}$
    for $i = 1 : n$ **do**
        if $UPC_i = b$ **and** $w_H(\mathbf{s}) > w_H(\mathbf{h}_i + \mathbf{s})$ **then**
            Flip y_i
            $\mathbf{s} = \mathbf{h}_i + \mathbf{s}$
        end
    end
    $I = I + 1$
end
Output: Decoded vector $\mathbf{y}$ and syndrome $\mathbf{s}$.

Figure 2. Optimized version of the algorithm in Figure 1

In the simulations for which results are presented in Chapter 4, the optimized version of the algorithm is used instead of the original algorithm introduced by Gallager.

3.4 Variations of the original decoder

To further improve the performance of the bit-flipping algorithms, numerous variations of the original algorithm have been presented. In this work, the Black-Gray variation and the CBBF variation of the bit-flipping algorithm and presented and analyzed.

3.4.1 Black-Gray variation

The main reason why iterative bit-flipping algorithms have a much larger decoding failure rate when used to decode the MDPC codes instead of the LDPC counterparts is the fact that the parity-check matrix of the LDPC code is significantly sparser than a corresponding matrix for the MDPC code. This means that each row of the matrix for the MDPC code contains a larger number of non-zero entries and therefore the number of bits included in calculating each bit in the syndrome is considerably increased. Due to this fact it is more difficult to decide whether a bit is in error or not. In the original construction presented in Figure 1 there was only one decision made per bit in one iteration and flipping a bit that is not in error leads to a higher probability of a decoding failure.

The Black-Gray variation of the bit-flipping algorithm [22] uses multiple stages in a single decoding iteration in order to reduce the probability of flipping unnecessary bits and reduces the failure rate. It uses two predefined thresholds δ and d to sort bits with high number of unsatisfied parity checks into two sets - called the black set and the gray set. The optimal choice for these parameters is discussed in Section 4.2.

Input : ciphertext $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, maximum number of iterations I_{max}, thresholds δ and d.

$I = 0$, $\mathbf{s} = H \cdot \mathbf{y}^T$

while $I < I_{max}$ **and** $\mathbf{s} \neq \mathbf{0}$ **do**

  $B = \emptyset, G = \emptyset$

  Compute the number of UPCs for each bit in $\mathbf{y}$

  for $i = 1 : n$ **do**

    if $UPC_i = \max\{\#_{UPC}\}$ **then**

      Flip y_i

      $B = B \cup \{i\}$

    else

      if $UPC_i \geq \max\{\#_{UPC}\} - \delta$ **then**

        $G = G \cup \{i\}$

      end

    end

  end

  $\mathbf{s} = H \cdot \mathbf{y}^T$

  Compute the number of UPCs for each bit in $\mathbf{y}$

  for $b \in B$ **do**

    if $UPC_b \geq \frac{d-1}{2}$ **then**

      Flip y_b

    end

  end

  $\mathbf{s} = H \cdot \mathbf{y}^T$

  Compute the number of UPCs for each bit in $\mathbf{y}$

  for $g \in G$ **do**

    if $UPC_g \geq \frac{d-1}{2}$ **then**

      Flip y_g

    end

  end

  $\mathbf{s} = H \cdot \mathbf{y}^T$

  $I = I + 1$

end

Output: Decoded vector $\mathbf{y}$ and syndrome $\mathbf{s}$.

Figure 3. The Black-Gray variation of the bit-flipping algorithm

The algorithm works as follows: the bits with a maximum number of unsatisfied parity checks are classified as black bits and flipped immediately. Bits whose count of unsatisfied checks differs from the maximum by a smaller number than the threshold δ, are classified as gray bits, but they are not flipped. After the initial loop, both the syndrome

and the number of unsatisfied checks are recalculated for each bit. Next, each bit in both the black and gray sets is analyzed again: if the number of unsatisfied checks exceeds a threshold defined by the second parameter d, the black bits which violate this threshold are flipped back to their original state and the gray bits are flipped. After each step, the syndrome and the number of unsatisfied checks are updated accordingly. Pseudocode for this variation of bit-flipping is presented in Figure 3 [22].

3.4.2 Candidate bit based bit-flipping algorithm

Another approach that can be taken to decide whether a bit should be flipped is based on the use of check weights. When there is a non-zero bit in the syndrome, it means that with high probability, some bits that participate in that check should be flipped. However, if a lot of bits are involved in the parity-check then this check is not reliable since it cannot be decided which bits in this check should be flipped. Therefore, the bits should be flipped if they participate in many unsatisfied parity checks and the number of bits participating in those checks is minimal. A candidate bit based bit-flipping algorithm [23] considers bits as candidates for flipping using the number of unsatisfied parity-checks and a parameter δ. The choice for this parameter is discussed in Section 4.2.

Input : ciphertext $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, maximum number of iterations I_{max}, threshold δ.

$I = 0$, $\mathbf{s} = H \cdot \mathbf{y}^T$, $H = (h_{ij})$

while $I < I_{max}$ **and** $\mathbf{s} \neq \mathbf{0}$ **do**

  Compute the number of UPCs for each bit in $\mathbf{y}$

  $C = \emptyset$

  for $i = 1 : n$ **do**

    if $UPC_i > \max\{\#_{UPC}\} - \delta$ **then**

      $C = C \cup \{i\}$

    end

  end

  for $i = 1 : m$ **do**

    $w_i = |\{c \in C : h_{ic} = 1\}|$

  end

  for $c \in C$ **do**

    if *Sum of* w_i *where* c *participates is minimal* **then**

      Flip y_c

    end

  end

  $\mathbf{s} = H \cdot \mathbf{y}^T$

  $I = I + 1$

end

Output: Decoded vector $\mathbf{y}$ and syndrome $\mathbf{s}$.

Figure 4. The CBBF variation of the bit-flipping algorithm

The algorithm works as follows: for each parity check, the number of candidate bits involved in that check is counted and referred to as a weight for that check. A candidate bit is only flipped when the sum of weights that the bit participates in is minimal. This variation of the algorithm tries to ensure that only bits which participate in parity-checks which are the most reliable are flipped. Pseudocode for this variation of the algorithm is presented in Figure 4 [23].

4 Analysis of algorithm implementations

In order to prevent attacks on the McEliece cryptosystem and to make the scheme feasible for practical use, the decoder must satisfy a number of criteria. The most obvious requirements are a low decoding failure rate (DFR) and a low time complexity, which would allow a ciphertext to be decoded quickly. However, a constant number of iterations or flips for every error pattern can also be required. If a pattern for which the decoder fails to decode the ciphertext results in a considerably longer run time for the decoder, then an attacker can use that information as a weakness against the cryptosystem. In this chapter, all three bit-flipping variations are analysed using both constraints to maximum number of iterations and flips to achieve the minimal DFR. The optimal choice for the parameters δ and d for the CBBF and BG algorithms respectively are discussed. Additionally, a few further optimizations in the implementation of the algorithms are presented.

4.1 Implementation optimizations

Every variation of the bit-flipping algorithm that was presented in Chapter 3 is based on counting the number of unsatisfied checks for each ciphertext bit during each iteration. This requires counting the number of ones in each column where the corresponding bit in the syndrome is non-zero. Performing this operation with a large matrix during each iteration makes the implementation slow. In Section 3.3 it was observed that the syndrome does not change arbitrarily when a ciphertext bit is flipped. The same observation can be made about the number of unsatisfied checks for each bit. The computation of the number of unsatisfied checks can be optimized by using this knowledge. The improved procedure is presented in Figure 5.

```
function failedChecks = updateFailedChecks(onePositionsInColumn, s, failed, ...
    onePositionsInRow, j)
    for k = 1:size(onePositionsInColumn,2)
        oneValue = onePositionsInColumn(j,k);
        failedColumns = onePositionsInRow(oneValue,:);
        if (s(oneValue) == 0)
            for m = 1:size(failedColumns,2)
                failed(failedColumns(m)) = failed(failedColumns(m)) - 1;
            end
        else
            for m = 1:size(failedColumns,2)
                failed(failedColumns(m)) = failed(failedColumns(m)) + 1;
            end
        end
    end
    failedChecks = failed;
end
```

Figure 5: Updating unsatisfied checks

The procedure works as follows: each time a bit is flipped, the syndrome is updated first. Then, for the column corresponding to the flipped bit in H, all non-zero entries are found. After finding a non-zero entry, all non-zero entries can be found in the row that the entry was found in, and check the corresponding position in the new syndrome. If the entry in the syndrome is zero, the number of unsatisfied checks is decremented for all positions involved in the parity check. Otherwise, the number of unsatisfied checks is incremented. Repeating this procedure each time a bit is flipped is faster than traversing the matrix during each iteration. It also allows for more dynamic updates to the threshold b at the cost of memory to store the number of unsatisfied checks for each bit. During the initialization phase, the initial count for the number of unsatisfied checks needs to be computed using the matrix traversal method described above. Every update after the initialization can be done with the improved procedure.

4.2 Optimal algorithm parameters

For the Black-Gray and CBBF variations of the bit-flipping algorithm, which use additional parameters for the decoding process, it is important that the choice of parameters is optimal. In the Black-Gray variaton there are two parameters d and δ. The value $\delta = 4$ has been previously used [22]. Thus, all Black-Gray decoding simulations in this work are done using $\delta = 4$, and it is important therefore to find the optimal values for the parameter d. For the CBBF algorithm, the optimal value for δ is also discussed. All simulations are done using the QC-MDPC codes with the parameters

$$n = 9602, \quad k = 4801, \quad w = 90, \quad n_0 = 2$$

By using these parameters, a 4801-bit plaintext block is encoded into a 9602-bit codeword, to which a varying number of errors is added. The parity-check matrix consists of two 4801×4801 circulants H_0 and H_1 and has the form

$$H = [H_0 \mid H_1],$$

4.2.1 Black-Gray algorithm

When choosing a value for the parameter d in the Black-Gray algorithm, it is important to minimize both the average number of iterations and the DFR. Furthermore, it is important to note when the conditions for flipping the gray bits and flipping the black bits back are met. When these conditions are unsatisfied, the Black-Gray algorithm becomes the original variation of the bit-flipping algorithm with threshold $b = \max\{\#_{UPC}\}$. At first all values for d from 20 to 90 in increments of 10 were tested with 100 errors. 100 different QC-MDPC codes were used, and for each code 1000 trials with different error patterns were done with maximum of 100 iterations. The results for these simulations can be seen in Table 1. The last two columns in the table represent the percentage of

total flips which were done on the gray bits and flips which flipped the black bits back to their original state respectively.

Table 1. BG simulations for the parameter d.

d	DFR	Avg. iterations	Gray flips	Back flips
20	0.763	78.4671	87%	6.31%
30	0.58	62.239	95.9%	1.2%
40	0.556	59.7999	96.9%	0.101%
50	0.378	45.2548	47.7%	5.27%
60	0.0605	33.1031	35.59%	0.0108%
70	0.0558	47.586	10.6%	0%
80	0.056	53.7878	2.01%	0%
90	0.056	55.4781	0%	0%

Results show that with 100 errors both the average number of iterations and the DFR reach their minimal values between $d = 60$ and $d = 70$. In that range, additional simulations were done with the same parameters. The results are presented in Table 2.

Table 2. BG simulations for the parameter d.

d	DFR	Avg. iterations	Gray flips	Back flips
61	0.0561	37.3154	27.2%	0.00325%
62	0.056	37.3174	25.1%	0.00296%
63	0.0551	40.8582	21%	0.000675%
64	0.0553	40.9783	19.84%	0.000652%
65	0.0554	43.642	16.3%	0.000513%
66	0.0554	43.957	16.1%	0.0000831%
67	0.0558	45.7719	13.3%	0%
68	0.0558	46.2421	11.3%	0%
69	0.0558	47.586	10.6%	0%

These measurements show that the lowest DFR occurs for $d = 63$, however the average number of iterations increases starting from $d = 60$. With a larger number of trials with each code the difference in DFR would be more noticeable and therefore $d = 63$ seems to be the optimal choice for 100 errors. It is worth noting that for a smaller number of errors smaller values for d can be used to accelerate decoding when the maximum number of iterations is small. These results are presented in Section 4.3.

4.2.2 CBBF algorithm

When analysing the CBBF algorithm, it becomes apparent that the choice $\delta > 1$ is required. Otherwise calculating the minimum check weights will have no benefit as typically the bit which has the maximum number of unsatisfied parity checks is distinct. The results for simulations of the CBBF algorithm with the same parameters as the Black-Gray algorithm are shown in Table 3.

Table 3. CBBF simulations for the parameter δ.

δ	**DFR**	**Avg. iterations**	**Errors**
2	0.0007	60.1606	95
2	0.0446	68.3411	100
2	0.388	84.5851	105
3	0.0011	61.6331	95
3	0.0554	70.4588	100
3	0.432	86.3479	105

Simulations show both DFR and the average number of iterations increasing for all error counts when increasing δ so it is optimal to choose $\delta = 2$.

4.3 Simulations with restricted number of iterations

As it is mentioned previously, an additional requirement for a decoding algorithm is that the running time for a decoding failure is not noticeably longer than for a successful result. Otherwise, an attacker can observe the time it takes to decode an erroneous codeword, and gain additional information about the cryptosystem. Therefore, it is required to restrict the maximum number of iterations so that the error patterns for which the decoding fails do not run for a considerably larger number of iterations. Simulations are done with restricting the number of iterations to either 20, 50 or 100, and by using the following algorithms:

1. Optimized version of the original bit-flipping algorithm with $b = \max\{\#_{UPC}\}$. (BF)

2. The Black-Gray algorithm with $d = 63$ and $\delta = 4$. (BG63)

3. The Black-Gray algorithm with $d = 50$ and $\delta = 4$. (BG50)

4. The CBBF algorithm with $\delta = 2$. (CBBF)

The results of the simulations with the limit of 20 iterations are presented in Figures 6-7 in the Appendix. In the legend, an abbreviation for each algorithm is given. For each algorithm, 5000 trials are done on each of the 200 randomly generated QC-MDPC codes per error count. The number of errors introduced is between 92 and 104.
When simulating the decoding with up to 20 iterations of the algorithm, only the Black-Gray algorithm with the reduced value for d was able to decode the codeword successfully. For 92 errors, it needs only 7.6 iterations on average to successfully decode a codeword with a DFR of $1.87 \cdot 10^{-3}$. Increasing the parameter d results in a very large failure rate given these restrictions. Other variations of the algorithm produced unsuccessful outcomes in all attempts.
Increasing the maximum number of iterations to 50 produced very different results. Black-Gray algorithm with $d = 63$ considerably outperformed the algorithm with the

smaller value for d in terms of the DFR. Starting from 98 errors the original bit-flipping produced better DFR, however the performance was still worse than that of the Black-Gray variation with larger d both in terms of the DFR and the average number of iterations. Choosing a smaller value for d in the Black-Gray algorithm still resulted in a smaller number of iterations on average, although the advantage became less pronounced with a larger number of introduced errors since larger DFR implies that with higher probability the upper limit of 50 iterations is achieved. It is worth noting that for the Black-Gray variation with the smaller value of d increasing the maximum number of iterations had only a 11% improvement on the DFR when nearly doubling the number of iterations on average. The CBBF algorithm with 50 iterations still produces poor results nearly in all attempts. The results of the simulations with 50 iterations are presented in Figures 8-9 in the Appendix.

By using the largest value of 100 iterations drastically improves the DFR of all algorithms except for the Black-Gray algorithm with the smaller value of d whose DFR showed practically no change when compared to the results with 50 iterations. The average number of iterations for that algorithm nearly doubles since its DFR is now considerably higher than for all other algorithms, and hence the large number of iterations is executed more often. The other three algorithms followed the same pattern for various numbers of errors both in terms of the DFR and the average number of iterations. In terms of the DFR, the CBBF algorithm outperforms all the algorithms, but it also requires the largest number of iterations on average. Since a single iteration of the CBBF algorithm is more time-consuming, this results in noticeably weaker performance than that of the other algorithms. The original bit-flipping algorithm needs a smaller number of iterations, but also has a higher DFR than both the CBBF and Black-Gray (with $d = 63$) algorithms. The Black-Gray algorithm with $d = 63$ uses a smaller number of iterations and produces a similar DFR to that of the CBBF algorithm. Nevertheless, it is still the optimal choice. If a smaller number of iterations is allowed, then the value

of the parameter d can be lowered to accelerate the decoding at the price of larger DFR. The results of the simulations with 100 iterations are presented in Figures 10-11 in the Appendix.

4.4 Simulations with restricted number of flips

In addition to restricting the maximum number of iterations, it is also possible to restrict the number of bits an algorithm can flip. Bounding the number of flips is different from restricting the number of iterations since it is possible to flip multiple bits during a single iteration, especially in the Black-Gray algorithm. It was chosen to restrict the number of flips to $1.2 \cdot t$ where t denotes the number of introduced errors. When comparing the results to the results in the other simulations, it can be seen that the original-bit flipping algorithm and the CBBF algorithm have a larger DFR while having a very similar average number of iterations compared to the algorithm with 100 iteration limit. For the Black-Gray algorithm with the larger value for the parameter d, the average number of iterations was similar to the algorithm with 50 iteration limit while having a larger DFR. However, the Black-Gray algorithm with a smaller values for the parameter d produced unsuccessful results more often while the average number of iterations did not become larger than 11. This is expected since a lower value for the parameter d results in more gray bits being flipped during each iteration, and also the number of black bits flipped back increases. The results of simulations with restricted number of flips are presented in Figures 12-13 in the Appendix.

Summary

In this thesis, a thorough overview of the original construction of the McEliece cryptosystem was presented. The QC-MDPC variant of the McEliece cryptosystem, which has smaller key sizes when compared with the original cryptosystem, was described. Various decoding algorithms used for the decoding of the QC-MDPC codes, which are used to decrypt ciphertext messages in the McEliece cryptosystem, were implemented and analyzed. The optimal values of the parameters used in different variations of the iterative decoding algorithms were found via simulations. All presented algorithms were tested in simulations with a restricted number of iterations and flips which are needed to ensure that the decoding time does not vary between different error patterns. From the simulations it can be deduced that the Black-Gray variation of the bit-flipping algorithm with parameter values $d = 63$ and $\delta = 4$ is the best choice when used with QC-MDPC codes when the number of iterations is not severely restricted. However, the value of d can be lowered to 50 to accelerate decoding at the cost of a larger DFR. It became apparent that the number of flips seems not to be a great restriction as it did not offer better performance neither in terms of the DFR nor the average number of iterations. However, experimental optimization of the algorithm parameters for other regimes could be considered for future research.

It was observed in simulations that the CBBF algorithm offers lower DFR compared to other algorithm variations, but currently each iteration takes more time to compute and also a larger number of iterations is required for successful decoding. Future work on optimizations of the CBBF algorithm could potentially make its performance similar to the other algorithm variations, and could make the CBBF algorithm a feasible alternative to the Black-Gray algorithm. Another possible future research direction is optimization of the weighted variants of the bit-flipping algorithm, which could improve the reliability of the decisions to flip the bits.

References

[1] Benioff P. The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines. *Journal of Statistical Physics*. Plenum Publishing Corporation, 1980, 563 – 564

[2] Shor P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. Published in: 1994 35th Annual Symposium on Foundations of Computer Science, 22.11.1994. DOI: 10.1137/S0097539795293172

[3] Rivest R. L, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, 1978, 2. http://people.csail.mit.edu/rivest/Rsapaper.pdf (04.04.2020)

[4] Koblitz N. Elliptic Curve Cryptosystems. *Mathematics of Computation 48*. American Mathematical Society. 1978, 203 – 209.

[5] Diffie W, Hellman M. New Directions in Cryptography, 1976, 11. https://ee.stanford.edu/ hellman/publications/24.pdf (04.04.2020)

[6] McEliece R. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *DSN Progress Report 44*. National Aeronautics and Space Administration, 1978, 114 – 116

[7] Roth R. M. Introduction to Coding Theory. Cambridge University Press. 2006.

[8] Goppa V. D. A New Class of Linear Correcting Codes. *Problems of Information Transmission 6:3*. 1970, 207 – 212.

[9] Vaidya S, Dutta S. A Study of the McEliece PKE, 2018, 3. https://www.cse.iitb.ac.in/ shaan/docs/McEliece_ProjectReport.pdf (07.05.2020)

[10] Engelbert D, Overbeck R, Schmidt A. A Summary of McEliece-Type Cryptosystems and Their Security. *IACR Cryptology ePrint Archive*. Journal of Mathematical Cryptology. 2006, 162. DOI: 10.1515/JMC.2007.009

[11] Patterson N. The Algebraic Decoding of Goppa Codes. *IEEE Transactions on Information Theory 21*. 1975, 203 – 207. DOI: 10.1109/TIT.1975.1055350

[12] Berlekamp E, McEliece R, Van Tilborg H. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory 24*. 1978, 384 – 386. DOI: 10.1109/TIT.1978.1055873

[13] Sidelnikov V. M, Shestakov S. O. On Insecurity of Cryptosystems Based on Generalized Reed-Solomon Codes. *Discrete Mathematics and Applications 2*. 1992, 439 – 444. DOI: 10.1515/dma.1992.2.4.439

[14] Bernstein D. J, Lange T, Peters C. Attacking and Defending the McEliece Cryptosystem. *International Workshop on Post-Quantum Cryptography*. 2008, 31 – 46. DOI: 10.1007/978-3-540-88403-3_3

[15] Augot D et al. Initial Recommendations of Long-Term Secure Post-Quantum Systems, 2015, 9. https://pqcrypto.eu.org/docs/initial-recommendations.pdf (13.04.2020)

[16] Baldi M, Chiaraluce F. Cryptanalysis of a New Instance of McEliece Cryptosystem Based on QC-LDPC Codes. *IEEE International Symposium*. 2007, 2591 – 2595. DOI: 10.1109/ISIT.2007.4557609

[17] Monico C, Rosenthal J, Shokrollahi A. Using Low Density Parity Check Codes in the McEliece Cryptosystem. *International Symposium on Information Theory*. 2000, 215. DOI: 10.1109/ISIT.2000.866513

[18] Gallager R. G. Low-Density Parity-Check Codes. The MIT Press, 1963.

[19] Huffman W. C, Pless V. Fundamentals of Error-Correcting Codes. Cambridge University Press, 2010.

[20] Misoczki R, Tillich J-P, Sendrier N, Barreto P. New McEliece Variants from Moderate Density Parity-Check codes. *IEEE International Symposium on Information Theory*. 2013, 2069 – 2073. DOI: 10.1109/ISIT.2013.6620590

[21] Von Maurich I, Oder T, Güneysu T. Implementing QC-MDPC McEliece Encryption, 2015, 4. https://www.pqcrypto.eu.org/docs/1-MDPCforACMTECS_preprint.pdf (25.04.2020)

[22] Drucker N, Gueron S, Kostic D. On Constant-Time QC-MDPC Decoding with Negligible Failure Rate. *IACR Cryptology ePrint Archive*. 2019, 1289.

[23] Kamabe H, Kobota S. Simple Improvements of Bit-Flipping Decoding. *The 12th International Conference on Advanced Communication Technology*. 2010, 113 – 118.

Appendix

I. Plots for simulations with at most 20 iterations

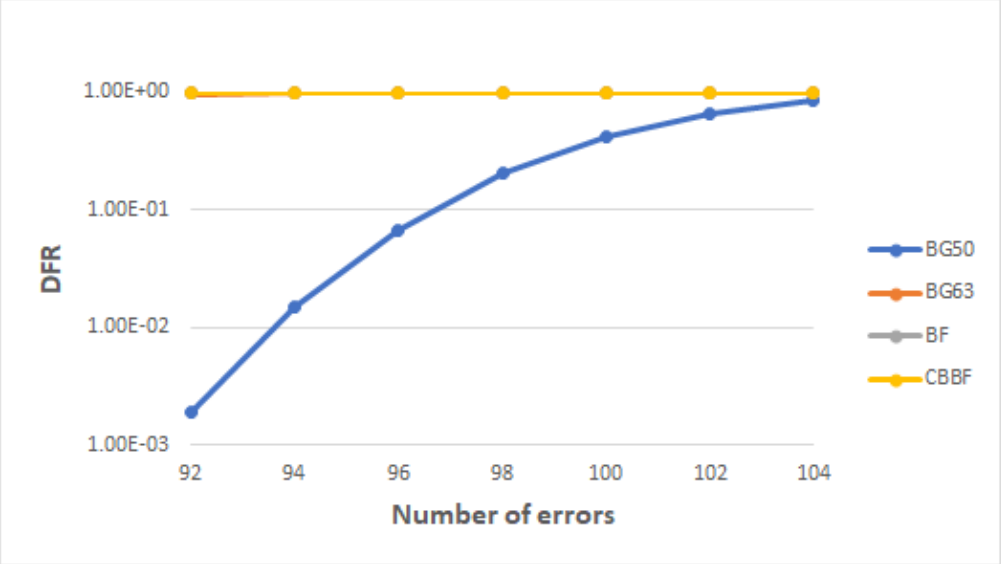

Figure 6. DFR with 20 iterations limit

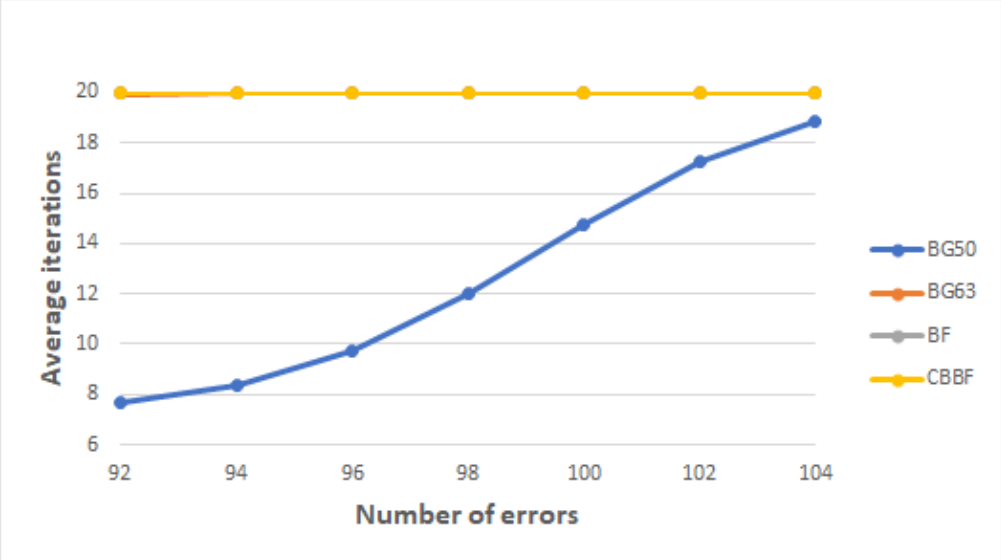

Figure 7. Average number of iterations for algorithms with 20 iterations limit

II. Plots for simulations with at most 50 iterations

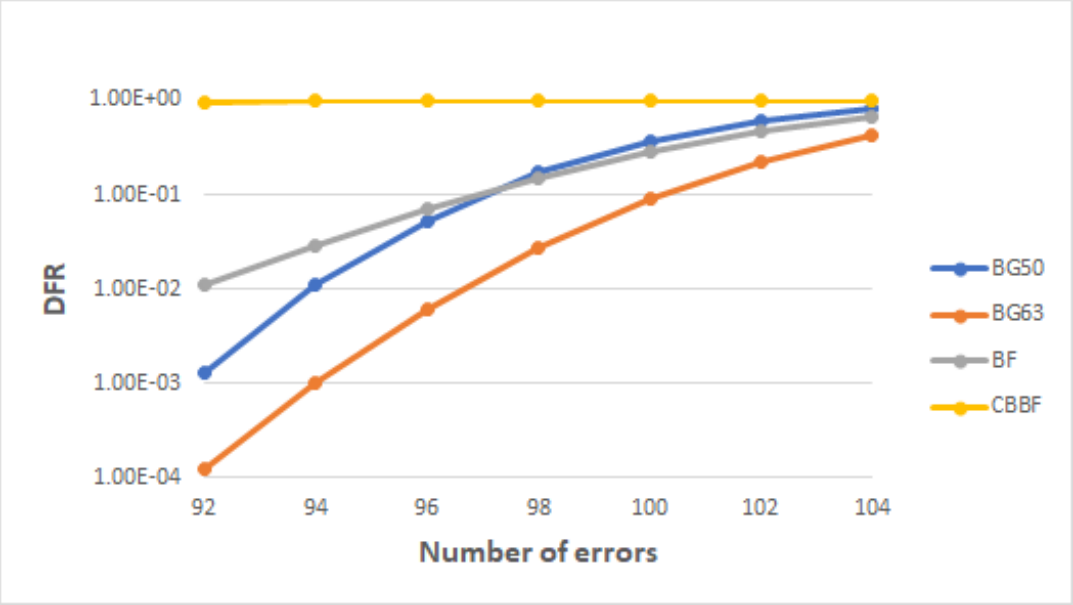

Figure 8. DFR with 50 iterations limit

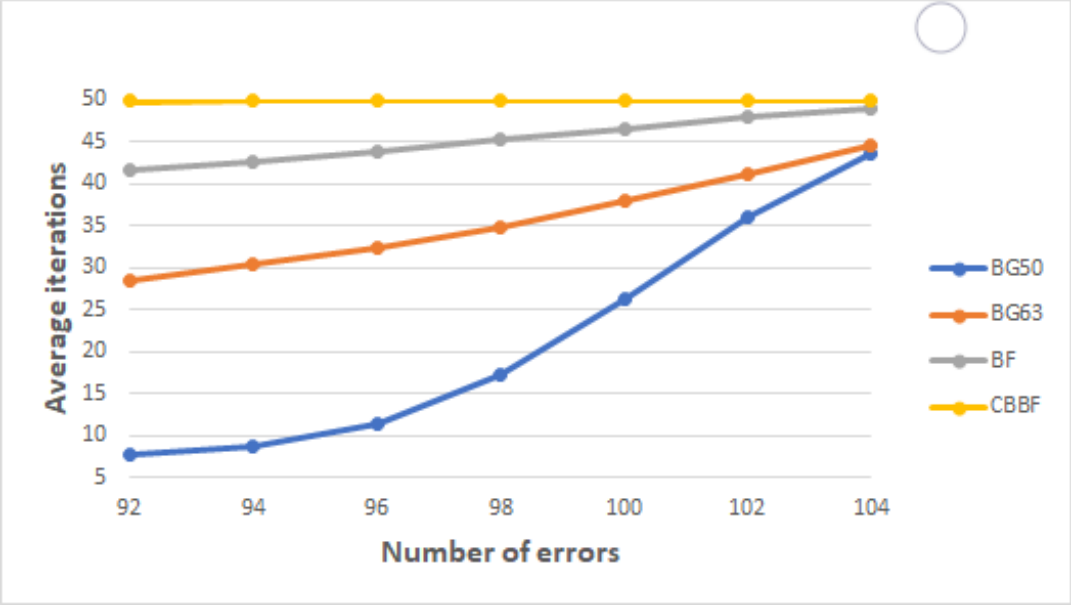

Figure 9. Average number of iterations for algorithms with 50 iterations limit

III. Plots for simulations with at most 100 iterations

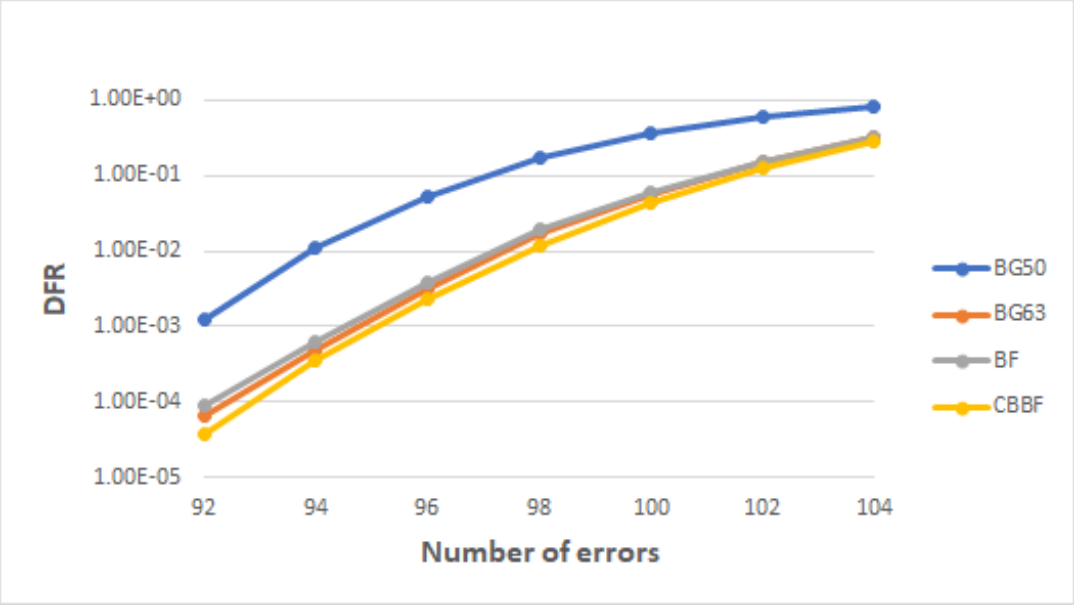

Figure 10. DFR with 100 iterations limit

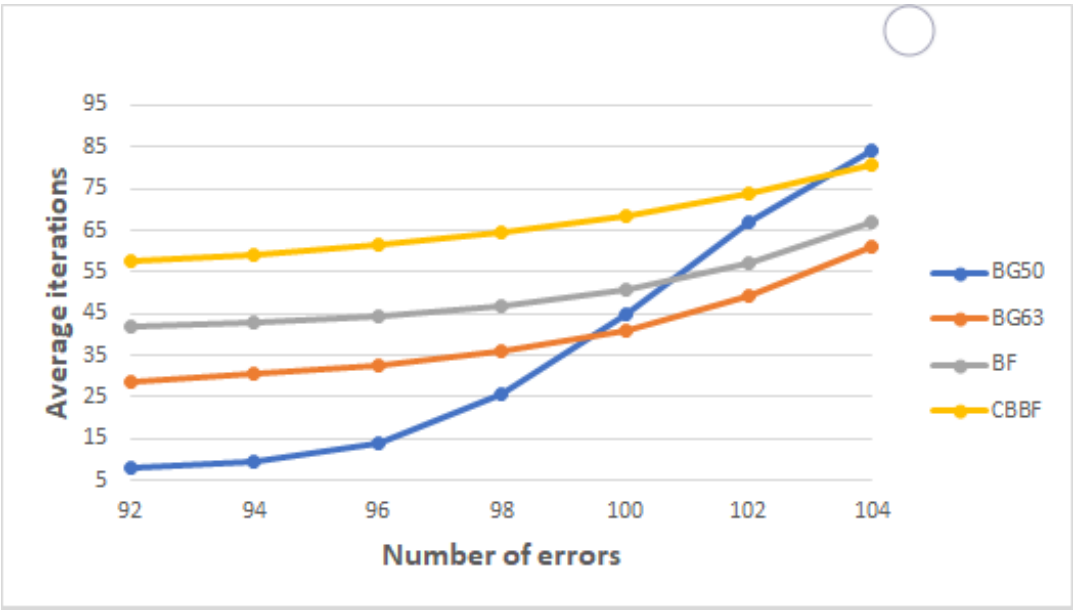

Figure 11. Average number of iterations for algorithms with 100 iterations limit

IV. Plots for simulations with number of flips

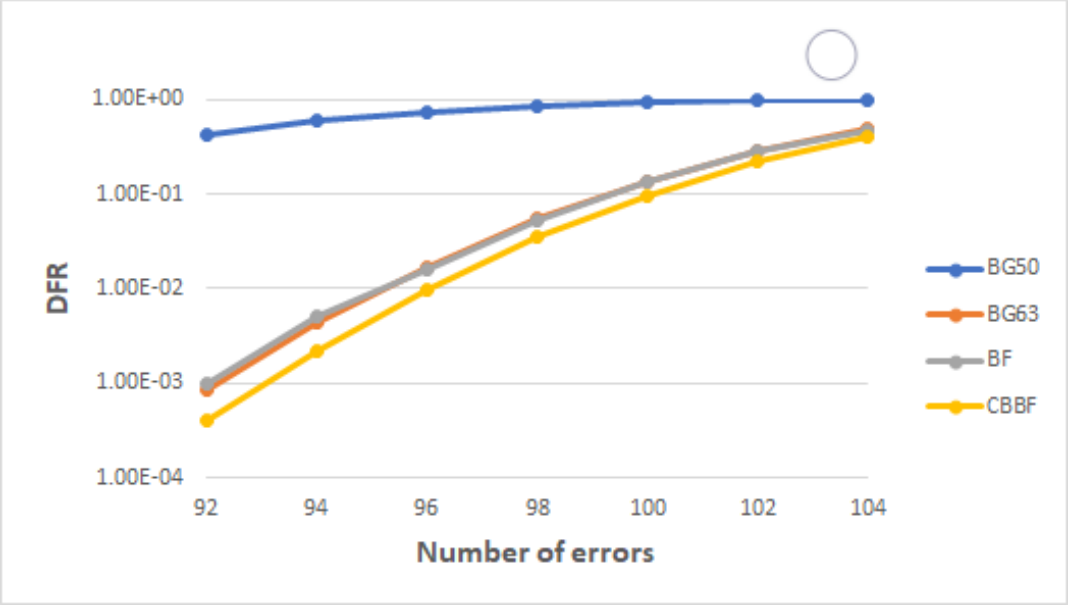

Figure 12. DFR with restricted number of flips limit

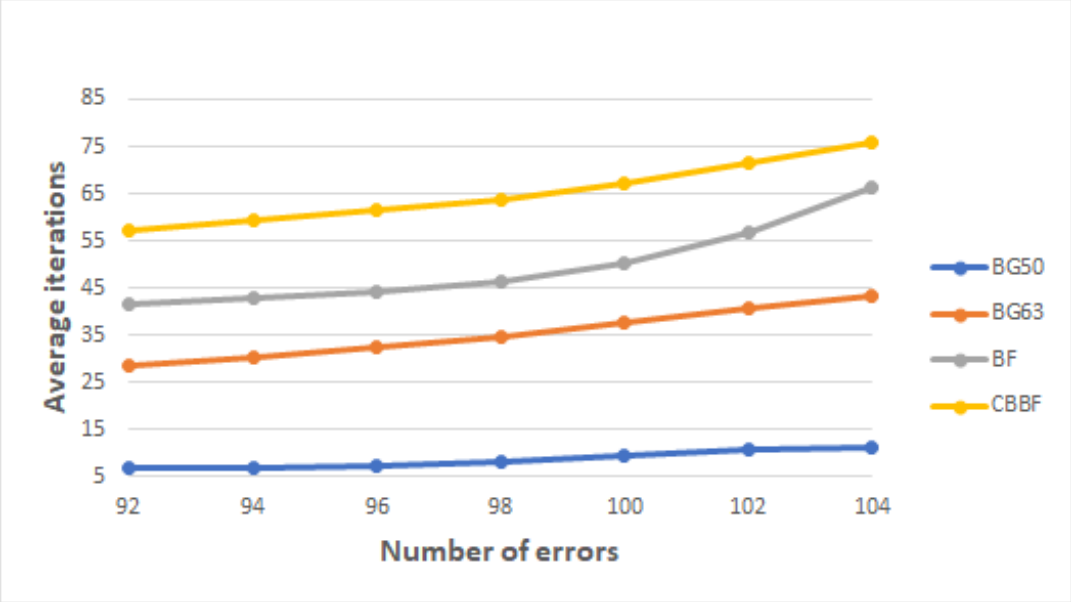

Figure 13. Average number of iterations for algorithms with number of flips limit

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Markus Punnar**,

*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, **Cryptosystem for Post-Quantum Age Based on Moderate-Density Parity-Check (MDPC) Codes**,

   *(*title of thesis)

   supervised by Vitaly Skachek and Irina Bocharova.

   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Markus Punnar

08/05/2020