UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Ergo Nigola

MDPC Code-Based Constructions and Their Decoding in Post-Quantum Cryptosystems

Masters's Thesis (30 ECTS)

Supervisor: Irina Bocharova, PhD
Supervisor: Vitaly Skachek, PhD

Tartu 2021

MDPC Code-Based Constructions and Their Decoding in Post-Quantum Cryptosystems

Abstract:

Quantum computers pose a threat to most of the popular public-key cryptosystems. This has prompted a search for good quantum-safe cryptographic protocols. Code-based cryptography is one promising approach, as its security relies on coding theory problems which are thought to be hard for both classical and quantum computers. The McEliece cryptosystem is the oldest code-based cryposystem and it is thought to be secure to this day. National Institute of Standards and Technology has started a project for standardizing quantum-resistant public-key cryptosystems. Among the candidates are three code-based cryptosystems, one of which is based on a McEliece variant which uses quasi-cyclic moderate-density parity-check (QC-MDPC) codes. We analyze a novel decoder in application to this McEliece variant and compare it to the existing decoders. Our results indicate that this decoder can be a viable alternative to the existing decoders, offering a trade-off between computational complexity and key and ciphertext lengths. We also analyze a new tail-biting unit memory convolutional QC-MDPC code construction as an alternative to the standard QC-MDPC codes used in this variant of the McEliece cryptosystem. We show that this code construction can be a better choice in settings where ephemeral keys are used.

Keywords:
Post-quantum cryptography, McEliece cryptosystem, code-based cryptography, coding theory, iterative decoding.

CERCS: P170 Computer science, numerical analysis, systems, control

MDPC koodidel põhinevad konstruktsioonid ja nende dekodeerimine post-kvant krüptosüsteemides

Lühikokkuvõte:

Kvantarvutid osutavad ohtu enamikele populaarsetele avaliku võtme krüptosüsteemidele. See on ajendanud otsinguid heade post-kvant turvaliste krüptograafiliste protokollide järele. Koodipõhine krüptograafia on üks paljulubav lähenemine, kuna selle turvalisus toetub kodeerimisteooria probleemidele, mida peetakse raskeks nii klassikalistele kui ka kvantarvutitele. McEliece krüptosüsteem on vanim koodipõhine krüptosüsteem ning seda peetakse turvaliseks tänaseni. Ameerika Ühendriikide Riiklik standardite ja tehnoloogia instituut on alustanud post-kvant turvaliste krüptosüsteemide standardiseerimise projekti. Kandidaatide seas on ka kolm koodipõhist krüptosüsteemi, millest üks põhineb McEliece krüptosüsteemi variandil mis kasutab kvaasi-tsüklilisi mõõduka tihedusega paarsuskontrolli (QC-MDPC) koode. Me uurime uudset dekooderit rakendatuna selles McEliece krüptosüsteemi variandis ning võrdleme seda olemasolevate dekooderitega. Meie tulemused näitavad, et see dekooder võib olla kasulik alternatiiv olemasolevatele dekooderitele, pakkudes kompromissi arvutusliku keerukuse ja võtmete ning salakirjade pikkuse vahel. Lisaks uurime ühte uut sabahammustavat ühikmäluga konvolutsioonilist QC-MDPC koodi konstruktsiooni alternatiivina standardsetele QC-MDPC koodidele. Me näitame, et see koodi konstruktsioon võib olla optimaalne valik olukordades, kus kasutatakse ühekordseid võtmeid.

Võtmesõnad:
Postkvant-krüptograafia, McEliece krüptosüsteem, koodipõhine krüpotgraafia, kodeerimisteooria, iteratiivne dekodeerimine.

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Quantum computers can solve the integer factorization and discrete logarithm problems in polynomial time [1]. This means that someone in possession of a sufficiently powerful quantum computer could break any cryptosystem based on the hardness of these problems, including RSA [2], DSA [3]. Code-based cryptography is thought to be secure against quantum attacks and is thus considered a potential alternative. One of the first code-based cryptosystems, the McEliece cryptosystem [4], is one such instance. The classic McEliece scheme uses Goppa codes. A problem with using Goppa codes is that the public key is very large. A McEliece variant based on quasi-cyclic moderate-density parity-check (QC-MDPC) codes has been proposed, which allows for much shorter keys and also simplifies the description of the keys and the decoding process [5]. A disadvantage of QC-MDPC codes is that decoding can fail, which also causes decryption to fail. Besides being an inconvenience, this can also pose a security risk, which we will explain later. This motivates us to study better decoding algorithms and code constructions, which would allow to decrease decoding failures.

The Post-Quantum Cryptography Standardization project of National Institute of Standards and Technology (NIST) is currently in its third round [6]. Among the four finalists in the public-key encryption and key-establishment algorithms category is the classic McEliece scheme. Among the five alternative candidates of the same category are two additional code-based cryptosystems: HQC [7] and BIKE [8]. BIKE is based on the QC-MDPC code based McEliece variant, HQC uses a different scheme. Members of the BIKE team and other researchers have analyzed the QC-MDPC code based McEliece scheme, its security and MDPC decoding techniques. We will go over the relevant results and extend upon these by analyzing a new decoder proposed by Alexander Nilsson, Irina Bocharova, Boris Kudryashov and Thomas Johansson [9], and a code construction proposed by Irina Bocharova and Boris Kudryashov [10]. Both the new decoder and the new code construction are yet unpublished at the time of writing this thesis.

We start Chapter 2 by giving the relevant coding theory definitions. After that we describe the classic McEliece cryptosystem and the QC-MDPC codes based variant. We also go over the security of the latter one. In Chapter 3 we explain the background and requirements of MDPC code decoders, we describe some well-known decoders and analyze the new decoder. We end the chapter by comparing the performance of these decoders. In Chapter 4 we present the new code construction, analyze its security, present a decoder for it and compare its performance to the standard QC-MDPC code construction. In Chapter 5 we conclude the thesis. The appendices contain a license and a reference to the source code used for simulations.

2 Background

Coding theory is a filed of study concerned with codes and their applications. A common application of codes is in data transmission, where they allow reliable communication over an unreliable medium. Due to one hard problem of coding theory, codes have also found a use in cryptography. We will give an overview of the relevant coding theory definitions before moving on to code-based cryptography.

2.1 Coding theory

For the purpose of cryptography, we are mainly interested in the sub-field of coding theory that deals with error correction, also known as channel coding. The process of channel coding is illustrated on Figure 1, based on the model of a communication system introduced by Claude E. Shannon [11].

Figure 1. Channel coding.

An information word $\mathbf{u}$ is encoded by the encoder into a codeword $\mathbf{c}$. The codeword $\mathbf{c}$ is transmitted over the channel. The channel might transform the codeword and introduce errors. As the output of the channel, we get the received word $\mathbf{y}$. The decoder then tries to decode $\mathbf{y}$ and produces a decoded codeword $\hat{\mathbf{c}}$. If $\hat{\mathbf{c}} = \mathbf{c}$ then we say that decoding was successful. To be able to recover the original information $\mathbf{u}$ from $\mathbf{c}$, the encoding should be one-to-one.

Definition (Code)**.** An (n, M) (block) code C over a finite alphabet F is a subset of F^n of cardinality $M > 0$. The parameter n is called the code length and M is the code size. Individual elements in C are called codewords.

Code dimension is defined as $k = \log_{|F|} M$ and code rate as $R = k/n$.

Definition (Linear code)**.** An (n, M) code C over a finite field $\mathbb{F}$ is called linear if C is a linear subspace of $\mathbb{F}^n$.

A linear (n, M) code C of dimension k is denoted as an $[n, k]$ code C.

Definition (Generator matrix)**.** A generator matrix of a linear $[n, k]$ code C is a $k \times n$ matrix G whose rows form a basis of the code.

A generator matrix of a code is usually not unique and it defines one possible encoding of information words to codewords. Given a generator matrix G of an $[n, k]$ code C, an information word $\mathbf{u} \in \mathbb{F}^k$ can be encoded into a codeword by the $\mathbb{F}^k \to C$ mapping $\mathbf{u} \mapsto \mathbf{c} = \mathbf{u}G$.

Definition (Parity-check matrix)**.** A parity-check matrix of a linear $[n, k]$ code C over $\mathbb{F}$ is an $r \times n$ matrix H over $\mathbb{F}$, such that for every $\mathbf{y} \in \mathbb{F}^n$:

$$\mathbf{y} \in C \iff \mathbf{y}H^T = \mathbf{0}$$

A parity-check matrix is usually chosen such that it has full rank, in which case $r = n - k$.

Note that if G is a generator matrix of a code C and H is a parity-check matrix of C, then $GH^T = \mathbf{0}$.

We say that a generator matrix G of an $[n, k]$ code is in systematic form if $G = \left[\ I_k \mid A\ \right]$, where I_k is the $k \times k$ identity matrix and A is a $k \times (n - k)$ matrix. A systematic parity-check matrix H can be constructed from a systematic $G = \left[\ I_k \mid A\ \right]$ by taking $H = \left[\ -A^T \mid I_{n-k}\ \right]$.

Here and in the remainder of the text, the vertical lines in matrices are provided only as a visual aid, to help distinguish different parts of the matrices.

Definition (Syndrome)**.** The syndrome of a word $\mathbf{y} \in \mathbb{F}^n$ with respect to a parity-check matrix H of an $[n, k]$ code over $\mathbb{F}$, is the vector

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

Thus, the codewords of a linear code are exactly those words for which $\mathbf{s} = \mathbf{0}$ (regardless of the choice of parity-check matrix for that code).

Definition (Hamming distance)**.** The Hamming distance between two vectors of the same length is the number of coordinates where they differ.

Definition (Hamming weight)**.** The Hamming weight (or simply weight) of a vector is the number of non-zero coordinates in it.

Definition (Code permutation equivalence)**.** Two linear codes are permutation equivalent if one can be obtained from the other by a permutation of the coordinates.

Definition (Dual code)**.** Let C be an $[n, k]$ linear code over $\mathbb{F}$. The dual code of the primal code C is the $[n, n - k]$ linear code $C^\perp$ defined by

$$C^\perp = \{\mathbf{x} \in \mathbb{F}^n \mid \mathbf{x} \cdot \mathbf{c} = 0 \ \ \forall \mathbf{c} \in C\}$$

where $\mathbf{x} \cdot \mathbf{c} = \sum_{i=1}^{n} x_i c_i$.

A generator matrix of the primal code is a parity-check matrix of the dual code and a generator matrix of the dual code is a parity-check matrix of the primal code.

Definition (Cyclic shift)**.** A cyclic shift of an n-tuple $(a_1, a_2, \ldots, a_n)$ is the n-tuple $(a_n, a_1, a_2, \ldots, a_{n-1})$.

Definition (Circulant matrix)**.** A circulant matrix is a square matrix in which each row, except the first, is a cyclic shift of the row above.

Definition (Quasi-cyclic code)**.** An $[n, k]$ linear code C is quasi-cyclic if there exists a positive integer $m < n$ such that for every codeword $\mathbf{c} \in C$, the word obtained by m cyclic shifts of $\mathbf{c}$ is also a codeword.

Definition (LDPC/MDPC code)**.** A low-density parity-check (LDPC) code or a moderate-density parity-check (MDPC) code is a linear code over $\mathbb{F}_2$ which has a parity-check matrix with a constant row weight w. For LDPC codes, w is a small constant. For MDPC codes, w scales in $O(\sqrt{n \log n})$.

LDPC codes were first introduced by Robert Gallager in 1962 [12] and are well studied by now. MDPC codes were derived from LDPC codes and introduced in 2009 [13].

2.2 McEliece cryptosystem

The McEliece cryptosystem is a public-key cryptosystem based on coding theory. It was developed by Robert McEliece and first published in 1978 [4]. In addition to the post-quantum security, another advantage of the McEliece cryptosystem is fast encryption and decryption [14]. The scheme is defined as follows.

- **Key generation.** Choose a generator matrix G of an $[n, k]$ code C, for which there exists an efficient decoding algorithm D that can correct up to t errors. Select a random $k \times k$ invertible matrix S and a random $n \times n$ permutation matrix P. Compute $G' = SGP$, which is the public key. The private key is (S, G, P, D);
- **Encryption.** To encrypt a plaintext $\mathbf{u} \in \mathbb{F}_2^k$, generate a random error vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t and obtain the ciphertext as $\mathbf{y} = \mathbf{u}G' + \mathbf{e}$.
- **Decryption.** To decrypt a ciphertext $\mathbf{y}$, first compute $\mathbf{y}P^{-1} = \mathbf{u}SG + \mathbf{e}P^{-1}$. Note that $(\mathbf{u}S)G$ is a codeword of C and that $\mathbf{e}P^{-1}$ still has weight t, as P^{-1} only permutes the error positions. Use the decoding algorithm D to decode $\mathbf{y}P^{-1}$ and obtain $\mathbf{u}S$. Finally, compute $\mathbf{u} = (\mathbf{u}S)S^{-1}$.

The security of the McEliece cryptosystem relies on the indistinguishability of the public code from a random linear code and on the hardness of decoding a random linear code. If it was not difficult to decode a random linear code, then the ciphertext could be decoded over the public code, with no knowledge of the private key. However, decoding a random linear code is known to be an NP-complete problem [15]. The indistinguishability of the public code from a random linear code is necessary to prevent an attacker from obtaining some efficient decoding method. The difficulty of distinguishing the public code from a random linear code depends on the underlying code family.

The original McEliece cryptosystem uses binary Goppa codes and no practical attack is known for this case. A shortcoming of the Goppa codes in this context is the large public key size. The use of various other codes has been suggested with the aim of decreasing the public key size, such as the use of quasi-cyclic codes [16] and quasi-dyadic Goppa codes [17]. But many of the variants based on algebraic codes have been broken [18].

2.3 QC-MDPC code based McEliece cryptosystem

Low-density parity-check (LDPC) codes are codes which have a parity-check matrix with low row weight [12]. This low density allows for efficient decoding using iterative decoding algorithms. LDPC codes have been suggested for use in the McEliece cryptosystem on multiple occasions [19, 20]. An advantage of LDPC codes is that they do not suffer from algebraic attacks as they do not have an algebraic structure. The main problem with the LDPC codes in McEliece cryptosystem is that the rows of the parity-check matrix can be recovered with relative ease due to their low weight.

In 2013, two McEliece cryptosystem variants based on moderate-density parity-check (MDPC) codes and quasi-cyclic MDPC (QC-MDPC) codes were proposed [5]. The row weight of MDPC codes scales in $O(\sqrt{n \log n})$ instead of being constant, which makes recovering the parity-check matrix harder. A downside of the higher row weight is the decrease in decoding performance. We will only be focusing on the quasi-cyclic variant as it is of more interest due to its very compact public keys.

For binary Goppa codes, there exist efficient decoding algorithms, such as the one by Patterson [21], which can always correct a certain amount of errors. The decoding algorithms used for MDPC codes are probabilistic and can fail at decoding. This means that decrypting a valid ciphertext can fail. Options to cope with this would be, for example, requesting a retransmission or making the probability of decoding failure negligible. Failure at decoding also gives rise to certain vulnerabilities, which will be discussed later.

The QC-MDPC codes used in the scheme have code length n and dimension k such that $n = mp$ for some integer m and $p = n - k$. With such parameters, both the generator and parity-check matrix can consist of $p \times p$ circulant blocks. The parity-check matrix will have the form

$$H = \left[\begin{array}{cccc} C_1 & C_2 & \cdots & C_m \end{array} \right]$$

where C_i, $i = 1, \ldots, m$, are $p \times p$ circulant matrices.

The corresponding generator matrix in systematic form has the structure

$$G = \left[\begin{array}{c|c} & (C_m^{-1}C_1)^T \\ & (C_m^{-1}C_2)^T \\ I_{(m-1)p} & \vdots \\ & (C_m^{-1}C_{m-1})^T \end{array} \right]$$

Denote the row weight of each matrix C_i as w_i for $i = 1, \ldots, m$, and the total row weight of H as $w = \sum_{i=1}^{m} w_i$. As C_m needs to be invertible (equivalent to having full rank), we require that w_m is odd, otherwise the columns of C_m would sum up to zero. For simplicity, we will only consider regular MDPC codes, for which $w_i = w/m$ for $i = 1, \ldots, m$. The inverse of a circulant matrix and the product of two circulant matrices are also circulant matrices, meaning that the matrices $(C_m^{-1}C_i)^T$, for $i = 1, \ldots, m-1$ are all circulant.

Strictly speaking, codes constructed in this way are not quasi-cyclic but are permutation equivalent to quasi-cyclic codes. A permutation equivalent quasi-cyclic code can be constructed by interleaving the coordinates of the blocks of length p, for example by taking the first coordinate of each block first, then the second coordinate of each block and so on. But as permutation equivalent codes are, in a lot of ways, equivalent, then these codes are also referred to as quasi-cyclic.

The QC-MDPC code base McEliece scheme is defined as follows.

- **Key generation.** Generate a parity-check matrix H of an $[n, k]$ QC-MDPC code such that H has row weight w. Construct the corresponding generator matrix $G = \left[\begin{array}{c|c} I_k & Q \end{array} \right]$. The public key is G, represented by the k-bit first column of Q. The private key is H, represented by its first row.

- **Encryption.** To encrypt a plaintext $\mathbf{u} \in \mathbb{F}_2^k$, encode it into a codeword $\mathbf{c} = \mathbf{u}G = (\mathbf{u}|\mathbf{u}Q)$. Then generate a random error vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t and obtain the ciphertext as $\mathbf{y} = \mathbf{c} + \mathbf{e}$.
- **Decryption.** To decrypt a ciphertext $\mathbf{y}$, decode it to obtain the codeword $\mathbf{c}$ and extract the message from the first k bits of $\mathbf{c}$.

Unlike the standard McEliece cryptosystem, the QC-MDPC variant does not use the scrambling matrix S and permutation matrix P, because recovering the private key H from the public key G is hard enough. Note that due to the systematic form of G, the plaintext appears in the first k bits of the ciphertext y with relatively few bits flipped. This can be solved by using an appropriate security conversion, such as the one by Kobara and Imai [22].

2.4 Security of QC-MDPC code based McEliece cryptosystem

There are two basic ways to attack the QC-MDPC code based McEliece cryptosystem. First option is to decode the ciphertext without the knowledge of the secret parity-check matrix H. Second option is to recover the secret H from the public G. Both of these problems are reduced to the problem of decoding a random linear code.

Problem (Syndrome decoding)**.**
Instance: A parity-check matrix $H \in \mathbb{F}_2^{r \times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_2^r$ and an integer $t > 0$, denoted as SD$(H, \mathbf{s}, t)$.
Problem: Find an error vector $\mathbf{e} \in \mathbb{F}_2^n$ of weight t such that $H^T = \mathbf{s}$.

For specific types of linear codes there can exists efficient decoding algorithms, but the general case of the syndrome decoding problem is known to be NP-complete [15]. The best known technique for solving the general syndrome decoding problem is information set decoding (ISD) [23]. Over the years, many improvements have been made to the basic formulation of ISD [24, 25, 26]. The workfactor of an ISD variant is usually described asymptotically for growing code length n, with fixed code rate $R = k/n$ and error rate t/n. All known variants of ISD have asymptotic complexity

$$\mathrm{WF}_{\mathrm{ISD}}(n, k, t) = 2^{ct(1+o(1))}$$

where the constant c depends on the code rate, error rate and the specific algorithm [27]. It has been shown that if we consider the case where error rate grows sublinearly with

code length, then the ISD workfactor has the same form as above and $c = \log_2 \frac{1}{1-R}$ [27]. For MDPC codes, both the row weight and inserted error count grow sublinearly with code length. Additionally, when t is small, using 2^{ct} with $c = \log_2 \frac{1}{1-R}$ gives a good estimate of the workfactor [28]. Altogether we have the following estimate:

$$\mathrm{WF}_{\mathrm{ISD}}(n, k, t) \approx 2^{t \log_2 \frac{1}{1-R}}$$

Message recovery attack. To decrypt a ciphertext $\mathbf{y}$ encoded using a public key G, first construct a parity-check matrix for the code. As the public generator matrix is in systematic form $G = [I|Q]$, the corresponding parity-check matrix in systematic form is $H_{sys} = [-Q^T|I]$. The matrix H_{sys} is not sparse and thus does not allow efficient decoding using standard MDPC decoding techniques. Calculate the syndrome $\mathbf{s} = \mathbf{y}H_{sys}^T$. Solve the general syndrome decoding problem SD($H_{sys}, \mathbf{s}, t$) to obtain the error vector $\mathbf{e}$. Extract the message from the first k bits of $\mathbf{y} - \mathbf{e}$.

Key recovery attack. To recover the private key H from the public key G of the code C, consider the dual code $C^\perp$. In the dual code $C^\perp$, G is a parity-check matrix and H is a generator matrix. Each row of H is a codeword of $C^\perp$ and has weight w. As the codewords are exactly those words for which the syndrome is zero, recover one row of H by solving SD($G, \mathbf{0}, w$). As H has quasi-cyclic structure, the whole matrix can be obtained by taking r quasi-cyclic shifts of a single row.

An aspect to take into account is that the syndrome decoding problems might be easier to solve when multiple acceptable solutions exist. For ISD, if N_s instances of the syndrome decoding problem are processed simultaneously and there are N_i solutions, then a speedup of $N_s/\sqrt{N_i}$ is gained [29].

In the message recovery attack, r instances of the decoding problem can be produced by taking the r quasi-cyclic shifts of the ciphertext. By shifting the ciphertext, the error positions are also shifted, meaning that each instance has a different solution. As we have r instances and r solutions, the workfactor of the message recovery attack is

$$\frac{\mathrm{WF}_{\mathrm{ISD}}(n, k, t)}{\sqrt{r}} \approx \frac{2^{t \log_2 \frac{1}{1-R}}}{\sqrt{r}}$$

In the key recovery attack, there is one instance, but r valid solutions (the r rows of H). Thus the workfactor of the key recovery attack is

$$\frac{\mathrm{WF}_{\mathrm{ISD}}(n, r, w)}{r} \approx \frac{2^{w \log_2 \frac{1}{R}}}{r}$$

Note that when the code rate R increase, the workfactor of message recovery attack also increases. But when the code rate of the primal code increases, the rate of the dual code decreases, and thus the workfactor of the key recovery attack decreases.

Security level of a cryptosystem is usually expressed in bits. A λ-bit security level means that it would take for an attacker 2^λ operations to break the system. For the QC-MDPC code based McEliece cryptosystem to have λ bits of classical security, the workfactor of the key recovery and message recovery attacks needs to be at least 2^λ.

The probability of a decoder failing to decode a received word is called decoding failure rate (DFR). The average DFR of QC-MDPC codes tends to increase as the row weight w or error count t increases. For a fixed code and decoder, whether decoding succeeds or fails depends on the relationship between the structure of the parity-check matrix and the error pattern. This dependence can be exploited to recover the private key. If an attacker obtains sufficient information about which error patterns cause a decoding failure, the private parity-check matrix can be reconstructed [30]. This attack is known as key recovery reaction attack. It is also known that after obtaining a single error pattern which causes a decoding failure, subsequent failing error patterns are much easier to obtain [31]. Because of this, we require the DFR to be at most $2^{-\lambda}$ for security level λ.

If we fix the code rate R, then the inserted error count t and the row weight w will be the main factors in determining message security and key security respectively. The block size p (which is equal to the parity-check matrix height r) plays a relatively small role. This means that we can choose p large enough to reach the required DFR.

2.5 Post-quantum security

Using Shor's algorithm, the integer factorization and discrete logarithm problem can be solved in polynomial time [1]. Cryptosystems based on the difficulty of these problems, such as RSA, would thus be broken by a quantum computer. The security of symmetric encryption schemes, such as AES [32], is also known to decrease with the use of quantum computers [33]. For example, Grover's algorithm can provide a quadratic speedup in breaking AES [34]. However, a quadratic speedup can be mitigated by doubling the key size. Using Grover's algorithm, a quadratic speedup can be gained for information set-decoding as well [35].

As a part of its post-quantum standardization project, NIST proposed to base the security against quantum attacks on the security of known block ciphers against quantum attacks [36]. For example, category 1 security requires that breaking the cryptosystem must be at least as hard as finding the 128-bit key of AES-128. Similarly, category 3 is based on AES-192 and category 5 on AES-256. The author of BIKE argue that a classical security level of 128, 192 and 256 bits should correspond to NIST security categories 1, 3 and 5 respectively.

3 MDPC decoding

Decoding of MDPC codes is done using techniques similar to those used for LDPC codes. Maximum likelihood decoding of LDPC codes is computationally too complex. Instead, iterative decoding algorithms are used. Due to the higher density of the parity-check matrices, these algorithms are less efficient on MDPC codes. The cryptographic setting imposes some additional requirements on decoders. Improving decoding allows to use shorter code lengths. In the McEliece cryptosystem, shorter code lengths will in turn mean shorter ciphertext and key lengths. At the same time, we want decoding to be fast so as to keep decryption fast.

3.1 MDPC decoding for cryptography

A relatively simple family of iterative decoders derive from the concept of bit-flipping. Bit-flipping decoders find at each iteration a set of positions in the received word that are most likely to be incorrect. The bits on those positions are then flipped and the next iteration proceeds with the updated word. Decoding proceeds until the word is decoded or a maximum number of iterations is reached. Different algorithms vary in the method of choosing the set of positions to be flipped and can include extra steps.

More powerful iterative decoders rely on the concept of belief propagation and its approximations, such as the sum-product and min-sum decoders [37]. Similarly to bit-flipping algorithms, they proceed in iterations, but instead of flipping bits, they update the likelihood of a bit having the value 0 or 1. They require floating-point operations and are computationally more complex.

Iterative decoding algorithms used for LDPC codes perform well due to the low density of the parity-check matrices. Increasing the density leads to decreased decoding performance. However, as the density of the parity-check matrices of MDPC codes is still relatively low, iterative decoding methods remain viable. Additionally, in the cryptographic setting we do not need to correct as many errors as might be required in communication systems. We only need to correct enough errors to guarantee a required security level.

As stated in Section 2.4, for λ-bit security level, we require DFR to be at most $2^{-\lambda}$, as information about the parity-check matrix can be inferred from decoding failures. Similar information can be inferred from the time it takes to decode a certain error pattern [38]. To prevent this and other potential side-channel attacks, we require the decoding to be constant-time. In particular, the number of iterations that a decoder performs

must be fixed. Increasing the iteration count increases decoding time, while decreasing the iteration count increases DFR. Other aspects of the algorithms, such as branching and memory access patterns, need to be considered for constant-time implementation as well. But accounting for these should not alter algorithms decoding performance, only running time.

Due to the probabilistic nature of LDPC decoding methods, performance of decoders is analyzed by running simulations of decoding. Simulating low DFR levels such as 2^{-128} is infeasible in practice. Instead, extrapolation methods are applied to results achieved at higher DFR levels.

3.2 Gallager's bit-flipping

Gallager originally introduced LDPC codes as having regular parity-check matrices, for which the row weight and column weight are constant. One could also consider irregular LDPC/MDPC codes, but we will only be looking at regular ones. With regular parity-check matrices, each bit participates in the same number of parity-checks, which simplifies the description of decoders.

Each row of a parity-check matrix can be considered a single parity-check. The non-zero positions of a row indicate which bits of the received word participate in that parity-check. The non-zero positions of a parity-check matrix column indicate which parity-checks a bit participates in. The syndrome of a received word indicates which parity-checks are unsatisfied.

Together with the introduction of LDPC codes, Gallager described an iterative hard-decision decoder for these codes [12]. This decoding algorithm is usually referred to as Gallager's bit-flipping or simply bit-flipping algorithm. The algorithm works by calculating the number of unsatisfied parity-checks for each bit of the received word, and then flipping those bits that have this value above a certain threshold. This process is iterated until the word is decoded. Algorithm 1 describes a method for calculating the number of unsatisfied parity-checks and Algorithm 2 describes the whole decoding process. Some computations in the algorithms presented here can be combined to get functionally equivalent but more efficient algorithms. We present only the less efficient versions for the sake of simplicity. Lines 4-6 in Algorithm 2 give an early stopping criterion for when decoding succeeds with less than the maximum number of iterations. For constant-time implementations, this would be removed.

In addition to the limit on the number of iterations, the performance of bit-flipping algorithms depends on the threshold selection rule. One option would be to choose the

Algorithm 1: `ComputeUPC`

Input: $H \in \mathbb{F}_2^{r \times n}$ (parity-check matrix), $\mathbf{s} \in \mathbb{F}_2^r$ (syndrome)
Output: $\mathbf{upc} \in \mathbb{N}^n$ (unsatisfied parity-check counts)

$\mathbf{upc} = 0^n$
for $i = 1 \dots n$ **do**
  for $j = 1 \dots r$ **do**
    if $H[i, j] = 1$ **and** $\mathbf{s}[j] = 1$ **then**
      $\mathbf{upc}[i] = \mathbf{upc}[i] + 1$
    end
  end
end
return $\mathbf{upc}$

Algorithm 2: Bit-Flip

Input: $H \in \mathbb{F}_2^{r \times n}$ (parity-check matrix), $\mathbf{y} \in \mathbb{F}_2^n$ (received word), $MaxIter \in \mathbb{N}$, `threshold` (function)
Output: $\mathbf{c} \in \mathbb{F}_2^n$

$\mathbf{c} = \mathbf{y}$
for $iter = 1 \dots MaxIter$ **do**
  $\mathbf{s} = H\mathbf{c}^T$
  if $\mathbf{s} = 0^r$ **then**
    break
  end
  $\mathbf{upc} = \texttt{ComputeUPC}(H, \mathbf{s})$
  $th = \texttt{threshold}(context)$
  for $i = 1 \dots n$ **do**
    if $\mathbf{upc}[i] \geq th$ **then**
      $\mathbf{c}[i] = 1 - \mathbf{c}[i]$
    end
  end
end
return $\mathbf{c}$

threshold to be the maximum number of unsatisfied parity-checks $\max(\mathbf{upc})$. With this threshold, only a small number of bits would be flipped at each iteration and thus a large number of iterations would be required. An alternative would be to choose $\max(\mathbf{upc}) - \delta$ for some constant $\delta > 0$, which would cause more bits to be flipped at each iterations. This can increase algorithms convergence speed, but can also cause more correct bits to be flipped, possibly leading to decoding failure. More advanced threshold selection rules can, for example, use syndrome weight, estimated remaining error count and parity-check matrix column weight [39]. In Algorithm 2, threshold selection rule is given by the argument function `threshold`. The *context* argument of `threshold` indicates that the threshold selection rule can depend on any information calculated at that point (that is, any information in the context of the algorithm).

3.3 Black-Gray-Flip

The authors of the BIKE suite have defined multiple variants of the basic bit-flipping algorithm suitable for use in the MDPC code based McEliece cryptosystem. One of them is Backflip [40, 41]. Backflip assigns each flipped bit a time-to-live value. After the time-to-live expires, the bit is flipped back. Bits with higher unsatisfied parity-check count are given higher time-to-live values to indicate that they have higher reliability. This method allows to record information about the reliability of bits, similar to soft-decision decoding methods.

BIKE round 1 submission's Additional Implementation [42] contains a decoder called Black-Gray, originally proposed in an early submission of CAKE (a predecessor of BIKE) by Nicolas Sandrier and Rafael Misoczki. It works similarly to Gallager's bit-flipping algorithm, but during each iteration it additionally creates two sets: a *black* set containing bits that were flipped and a $gray$ set containing bits that were slightly below threshold. After the initial flips, bits in the black set, for which over half the parity-checks are unsatisfied, are flipped back. After that, bits in the gray set, for which over half the parity-checks are unsatisfied, are flipped. Black-Gray decoder is given in Algorithm 3.

The Backflip algorithm achieves low DFR levels, but was found to be not well suited for constant-time implementation [43]. The Black-Gray algorithm is better suited for constant-time implementation and can achieve similarly low DFR levels with less computation [43].

The black and gray sets in Black-Gray decoder help to correct falsely flipped bits and give confidence in correctness of performed flips. The probability of falsely flipping a

Algorithm 3: Black-Gray

Input: $H \in \mathbb{F}_2^{r \times n}$ (parity-check matrix), $\mathbf{y} \in \mathbb{F}_2^n$ (received word), $\tau \in \mathbb{N}$, $MaxIter \in \mathbb{N}$, `threshold` (function)

Output: $\mathbf{c} \in \mathbb{F}_2^n$

$\mathbf{c} = \mathbf{y}$
$d = \texttt{ColumnWeight}(H)$
$maskTh = (d+1)/2 + 1$
for $iter = 1, \ldots, MaxIter$ **do**
  $\mathbf{s} = H\mathbf{c}^T$
  $\mathbf{upc} = \texttt{ComputeUPC}(H, \mathbf{s})$
  $th = \texttt{threshold}(context)$
  $\mathbf{black} = 0^n$
  $\mathbf{gray} = 0^n$
  for $i = 1 \ldots n$ **do**
    if $\mathbf{upc}[i] \geq th$ **then**
      $\mathbf{c}[i] = 1 - \mathbf{c}[i]$
      $\mathbf{black}[i] = 1$
    else if $\mathbf{upc}[i] \geq th - \tau$ **then**
      $\mathbf{gray}[i] = 1$
  end
  $\mathbf{c} = \texttt{MaskedBitFlip}(\mathbf{c}, H, \mathbf{black}, maskTh)$
  $\mathbf{c} = \texttt{MaskedBitFlip}(\mathbf{c}, H, \mathbf{gray}, maskTh)$
end
return $\mathbf{c}$

function $\texttt{MaskedBitFlip}(\mathbf{c}, H, \mathbf{mask}, th)$
  $\mathbf{s} = H\mathbf{c}^T$
  $\mathbf{upc} = \texttt{ComputeUPC}(H, s)$
  for $i = 1 \ldots n$ **do**
    if $mask[i] = 1$ **and** $\mathbf{upc}[i] \geq th$ **then**
      $\mathbf{c}[i] = 1 - \mathbf{c}[i]$
    end
  end
  return $\mathbf{c}$

correct bit decreases as the number of remaining errors decreases. Thus, after correcting some initial errors, the use of black and gray sets is not needed anymore. This reasoning lead to the creation of the Black-Gray-Flip (BGF) decoder [44]. BGF performs one iteration of Black-Gray decoding and then proceeds with basic bit-flipping. The algorithm for BGF is as Algorithm 3, except that operations on lines 20 and 21 are performed only during the first iteration. BGF is the decoding algorithm used by BIKE as of its round 3 submission [28].

3.4 Weighted bit-flipping

Another class of decoders that try to improve the decoding performance of bit-flipping are weighted bit-flipping (WBF) decoders [45, 46, 47]. These decoders assign weights, alternatively referred to as reliabilities, to each syndrome component. The weights are usually calculated from the soft information provided by the channel. In the setting of McEliece cryptosystem, this information does not exist. A variant of bit-flipping, called candidate bit based bit-flipping, has been proposed, which also assigns reliability values to syndrome components, but does not require soft information from the channel or real-valued computations [48]. A weighted bit-flipping variant based on similar ideas has been proposed by Alexander Nilsson, Irina Bocharova, Boris Kudryashov and Thomas Johansson and has been accepted for publication at IEEE International Symposium on Information Theory 2021 [9]. This weighted bit-flipping variant is designed for use in the MDPC code based McEliece cryptosystem and we will describe this decoder next.

As described in Section 3.1, bit-flipping works by identifying at each iteration a set of bits that are most likely to be incorrect. The main property used to select those bits is the amount of unsatisfied parity-checks that they participate in. The goal of weighted bit-flipping is to improve bit-flipping decoding performance by additionally assigning weights to those parity-checks. If some bit is determined to be unreliable and should possibly be flipped, then the parity-checks that it participates in are also somewhat unreliable. Thus, parity-checks containing a lot of unreliable bits should be assigned a lower weight, as they give less reliable information about whether a given bit in them should be flipped.

The newly proposed WBF algorithm starts by finding the number of unsatisfied parity-checks for each bit. Each bit that has over half of its parity-checks unsatisfied is considered as a vote for flipping. Next, the number of votes in each parity-check is counted and this count is used to determine the weight of that parity-check.

A satisfied parity-check must contain an even number of errors (possibly none) and

an unsatisfied parity-check must contain an odd number of errors. The authors of the algorithm have determined through simulations the following likelihoods:

$$A_0(\theta_c) = \log \frac{\Pr(e_c = 0 | s_c = 0, \theta_c)}{\Pr(e_c > 0 | s_c = 0, \theta_c)}$$
$$A_1(\theta_c) = \log \frac{\Pr(e_c = 1 | s_c = 1, \theta_c)}{\Pr(e_c > 1 | s_c = 1, \theta_c)}$$

where θ_c is the number of votes in the c-th parity-check, s_c is the value of the c-th syndrome component and e_c is the number of errors in the bits of the c-th parity-check. These function were simulated for a fixed set of parameters and might not be optimal for other parameters. The functions A_0 and A_1 can be used to assign a weight to a parity-check, given the respective syndrome value and vote count. Based on these weights, a new reliability value is calculated for all bits and the most unreliable ones are flipped. These steps are repeated in iterations until the received word is decoded or a maximum number of iterations is reached.

Instead of using this theoretical decoder, the authors opted to use a simplification of it. The simplified version is presented in Algorithm 4. Instead of using the functions A_0 and A_1, the weights of the parity-checks with the least amount of votes (lines 19 and 21) are amplified by 3 (lines 20 and 22). This is based on the observation that A_0 and A_1 assign parity-checks with a small amount votes about 3 to 4 times higher reliabilities compared to parity-checks with many votes. The authors claim that the performance of the simplified WBF algorithm is hardly distinguishable from the one using A_0 and A_1.

The originally proposed `selectFlipMask` rule (line 28) chooses a single bit at each iteration. It starts by finding all the bits which have maximal $\mathbf{metric}$ value. For each such bit, the total amount of votes from all the parity-checks that it participates in is calculated. Finally, the bit with the smallest amount of total votes is chosen. The authors note the possibility of other selection rules which choose more than one bit at each iteration, giving faster convergence at the cost of increased DFR.

Weighted bit-flipping includes some steps, such as calculating votes and weights, which are not required for Black-Gray-Flip. This makes WBF computationally more complex. The reasoning that lead to the creation of BGF was that more sophisticated bit selection rules are only required to correct some initial errors, after which simpler methods can be used. Based on the same idea, the authors of the described WBF algorithm propose a hybrid decoder, which performs two iterations of WBF followed by three iterations of BGF. This hybrid decoder provides a compromise between computational complexity and decoding performance.

Algorithm 4: Simplified Weighted Bit-Flipping

Input: $H \in \mathbb{F}_2^{r \times n}$ (parity-check matrix), $\mathbf{y} \in \mathbb{F}_2^n$ (received word), $MaxIter \in \mathbb{N}$, `selectFlipMask` (function)

Output: $\mathbf{c} \in \mathbb{F}_2^n$

$\mathbf{c} = \mathbf{y}$
$d = \texttt{ColumnWeight}(H)$
for $iter = 1 \dots MaxIter$ **do**
  $\mathbf{s} = H\mathbf{c}^T$
  $\mathbf{upc} = \texttt{ComputeUPC}(H, \mathbf{s})$
  $\mathbf{votes} = 0^r$
  for $i = 1 \dots r$ **do**
    for $j = 1 \dots n$ **do**
      if $H[j,i] = 1$ **and** $\mathbf{upc}[j] > d/2$ **then**
        $\mathbf{votes}[i] = \mathbf{votes}[i] + 1$
      end
    end
  end
  $minVotes0 = \min_{i=1\dots r;\mathbf{s}[i]=0} \mathbf{votes}[i]$
  $minVotes1 = \min_{i=1\dots r;\mathbf{s}[i]=1} \mathbf{votes}[i]$
  $\mathbf{rel} = 0^r$
  for $i = 1 \dots r$ **do**
    $\mathbf{rel}[i] = 2 \cdot \mathbf{s}[i] - 1$
    if $\mathbf{s}[i] = 0$ **and** $\mathbf{votes}[i] \leq minVotes0 + 3$ **then**
      $\mathbf{rel}[i] = 3 \cdot \mathbf{rel}[i]$
    else if $\mathbf{s}[i] = 1$ **and** $\mathbf{votes}[i] \leq minVotes1 + 3$ **then**
      $\mathbf{rel}[i] = 3 \cdot \mathbf{rel}[i]$
  end
  $\mathbf{metric} = 0^n$
  for $i = 1 \dots n$ **do**
    $\mathbf{metric}[i] = \sum_{j=1\dots r, H[j,i]=1} \mathbf{rel}[j]$
  end
  $\mathbf{flipMask} = \texttt{selectFlipMask}(context)$
  for $i = 1 \dots n$ **do**
    if $\mathbf{flipMask}[i] = 1$ **then**
      $\mathbf{c}[i] = 1 - \mathbf{c}[i]$
    end
  end
end
return $\mathbf{c}$

3.5 Performance comparison

We will proceed to compare the performance of BGF, WBF and the hybrid solution through simulation. The parameters used for the simulation are same as defined by BIKE security level 1 corresponding to 128-bit classical security level [28]. BIKE level 1 uses inserted error count $t = 134$, parity-check matrix row weight $w = 142$ and parity-check matrices consisting of $m = 2$ circulant blocks. BIKE level 1 uses BGF decoder with the parameters $\tau = 3$, $MaxIter = 5$ and

$$\texttt{threshold}(S) = \max(\lfloor 0.0069722 \cdot S + 13.530 \rfloor, 36)$$

where S is the syndrome weight. These same parameters have been used for BGF in this simulation.

The authors of the WBF algorithm performed a similar simulation comparing these three decoders. The `selectFlipMask` rule used for WBF is not specified, but their simulation results indicate that it gives good decoding performance while requiring many iterations. For the simulations presented here, WBF used a `selectFlipMask` rule, devised by the author of this thesis, which chooses all the bits that have the $\mathbf{metric}$ value at or above the threshold

$$\max(\mathbf{metric}) - (\max(\mathbf{metric}) - \min(\mathbf{metric})) \cdot (0.55 - \texttt{Weight}(\mathbf{s})/r).$$

The WBF decoder was limited to running 7 iterations.

The hybrid decoder uses the same parameters as the BGF and WBF decoders for its respective WBF and BGF iterations. As mentioned, the hybrid decoder runs 2 iterations of WBF and 3 iterations of BGF.

One iteration of Black-Gray flipping performs three sets of flips: the initial flips, the black set flips and the gray set flips. So five iterations of BGF perform in total 7 sets of flips. Similarly, using the iteration counts specified, the hybrid decoder and WBF decoder perform in total 7 sets of flips. However, due to the extra calculations, WBF is computationally most complex, followed by the hybrid decoder. The BGF decoder requires the least amount of computations.

The simulation results are presented if Figure 2. Each data point was simulated with 30 random codes and 50 random ciphertexts per code. The algorithms were implemented and simulations were run in MATLAB, a reference to source code can be found in Appendix I.

It can be seen that that the WBF and hybrid decoder perform better than BGF at the lower DFR levels. However, it can also be seen that their performance gain from higher

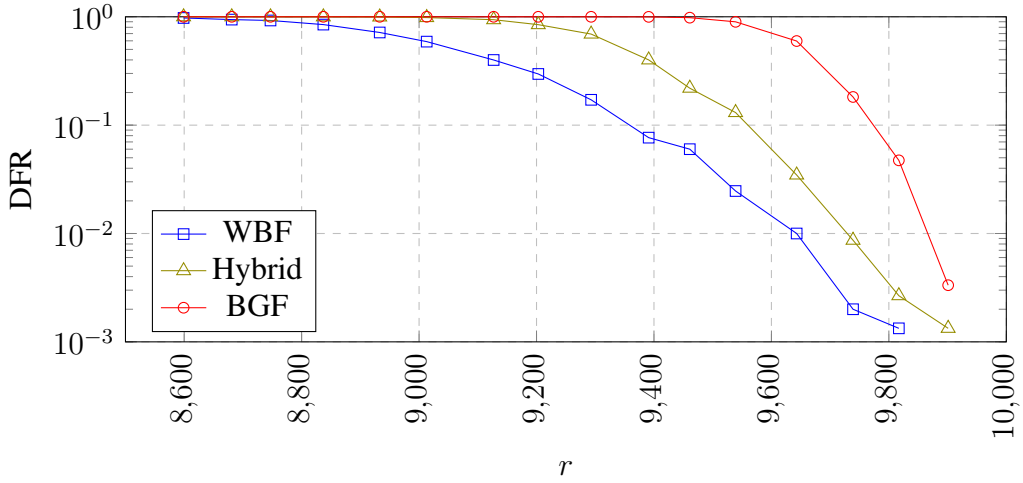

Figure 2. Comparison of decoders DFR in relation to parity-check matrix height r.

r values is not as big as for the BGF decoder, which would most likely surpass them at lower DFR levels. The WBF and hybrid decoders seem to be more sensitive to changes in r values. These shortcomings can be attributed to the specific threshold rule used. The simulations run by the authors of the WBF algorithm show that using different threshold rules, WBF can achieve more stable and better performance even at lower DFR levels, although requiring more iterations. However, the results presented here provide evidence that the WBF decoder can give good decoding performance while maintaining a low iteration count, at least under certain conditions.

4 Tail-biting convolutional code based McEliece

Over the year, various codes have been suggested for use in the McEliece cryptosystem [18]. Among them have been suggestions of convolutional codes [49, 50, 51], at least one of which has been broken [52]. We will present and analyze a tail-biting unit memory convolutional QC-MDPC code constructions proposed by Irina Bocharova and Boris Kudryashov in an unpublished paper [10]. This construction allows using a better decoding technique, which might allow improved code parameters. An advantage of this code constructions is that its security can be partially reduced to the security of the QC-MDPC code based McEliece scheme.

4.1 LDPC convolutional codes

Unlike block codes, convolutional codes can encode information sequences consisting of an arbitrary amount of b-bit blocks. A convolutional encoder encodes b-bit information blocks into c-bit code blocks and is characterized by the code rate $R = b/c$. LDPC convolutional codes, also referred to as spatially-coupled LDPC codes, are a sub-type of convolutional codes [53]. LDPC convolutional codes can be represented by a semi-infinite parity-check matrix of the form

$$H_{\text{conv}} = \begin{bmatrix} H_0 & & & \\ H_1 & H_0 & & \\ \vdots & H_1 & H_0 & \\ H_M & \vdots & H_1 & \ddots \\ & H_M & \vdots & \ddots \\ & & H_M & \ddots \\ & & & \ddots \end{bmatrix}$$

where H_i, for $i = 0, \ldots, M$, are $(c - b) \times c$ binary matrices and the empty parts are filled with zeros. The parameter M is referred to as syndrome former memory. If M is equal to one, then the code is said to have *unit memory*. In general, the matrices H_i, for $i = 0, \ldots, M$, could depend on the number of the row that they are in. But we are only interested in *time-invariant* convolutional codes, for which the matrices H_i depend only on the index i.

To get a finite parity-check matrix of a block code, we need to terminate the convo-

lutional code. One option would be *zero-tail* termination, which would give as the parity-check matrix

$$H_{\text{zt}} = \begin{bmatrix} H_0 & & & \\ H_1 & H_0 & & \\ \vdots & H_1 & \ddots & \\ H_M & \vdots & \ddots & H_0 \\ & H_M & \ddots & H_1 \\ & & \ddots & \vdots \\ & & & H_M \end{bmatrix}.$$

If H_{zt} was obtained by terminating H_{conv} after L levels then H_{zt} is an $(L+M)(c-b) \times Lc$ matrix. One problem with zero-termination is that it causes code rate loss. Where as the convolutional code has the rate b/c, the block code represented by H_{zt} has rate

$$\frac{b}{c} - \frac{M(c-b)}{Lc}.$$

Additionally, the parity-check matrix H_{zt} is unsuitable for the McEliece cryptosystem, because some rows have much lower weight than others. These low weight rows are low weight codewords in the dual code, and recovering them is much easier, which would in turn make recovering the whole parity-check matrix much easier. To avoid these problem, we use *tail-biting* termination. With tail-biting we get a parity-check matrix of the form

$$H'_{\text{tb}} = \begin{bmatrix} H_0 & & & H_M & H_{M-1} & \dots & H_1 \\ H_1 & H_0 & & & H_M & \dots & H_2 \\ \vdots & H_1 & \ddots & & & \ddots & \vdots \\ H_M & \vdots & \ddots & H_0 & & & H_M \\ & H_M & \ddots & H_1 & H_0 & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ & & & H_{M-1} & H_{M-2} & \dots & H_0 \end{bmatrix}.$$

If a tail-biting level of L is used, then H_{tb} is an $L(c-b) \times Lc$ parity-check matrix of an $[Lc, Lb]$ block code with rate b/c. We can permute the parity-checks (the rows of the

parity-check matrix) and relabel the matrices H_i for $i = 0, \ldots, M$ to get a parity-check matrix of the form

$$H''_{\text{tb}} = \begin{bmatrix} H_0 & H_1 & \cdots & H_M & & & \\ & H_0 & H_1 & \cdots & H_M & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ H_M & & & H_0 & H_1 & \cdots & H_{M-1} \\ H_{M-1} & H_M & & & H_0 & \cdots & H_{M-2} \\ \vdots & \vdots & \ddots & & & \ddots & \vdots \\ H_1 & H_2 & \cdots & H_M & & & H_0 \end{bmatrix},$$

which is the form used by the proposed tail-biting unit memory convolutional QC-MDPC codes.

4.2 New code construction

Standard QC-MDPC codes are already tail-biting convolutional codes, with no limit on the memory M. The new code construction adds another level of convolutional structure, where we limit the memory to one. The codes suggested by Irina Bocharova and Boris Kudryashov are a type of tail-biting unit memory convolutional QC-MDPC codes. Thus, the parity-check matrices of these codes are of the form

$$H_{\text{tb}} = \begin{bmatrix} H_0 & H_1 & & & \\ & H_0 & H_1 & & \\ & & \ddots & \ddots & \\ & & & H_0 & H_1 \\ H_1 & & & & H_0 \end{bmatrix}.$$

The submatrices H_0 and H_1 are defined as $p \times mp$ binary matrices

$$H_0 = \begin{bmatrix} C_1 & C_2 & \cdots & C_m \end{bmatrix}$$

$$H_1 = \begin{bmatrix} C_{m+1} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

where C_i, for $i = 1, \ldots, m+1$, are $p \times p$ circulant matrices with row weight $w/(m+1)$ and $\mathbf{0}$ is a $p \times p$ all-zero matrix. If a tail-biting level L is used, then H_{tb} is a $Lp \times Lmp$ matrix. Note that H_0 is parity-check matrix of an $[mp, (m-1)p]$ QC-MDPC code. By expanding H_0 and H_1 in H_{tb}, we see that

$$H_{\text{tb}} = \begin{bmatrix} C_1 & \cdots & C_m & C_{m+1} & & & & \\ & & & C_1 & \cdots & C_m & C_{m+1} & \\ & & & & \ddots & \ddots & \ddots & \\ C_{m+1} & & & & & C_1 & \cdots & C_m \end{bmatrix}.$$

By permuting the block columns of H_{tb} that contain C_m to the end, we get a parity-check matrix H of a permutation equivalent code:

$$H_{L;m} = \left[\; A \;\middle|\; B \; \right]$$

where A is an $Lp \times L(m-1)p$ matrix

$$A = \begin{bmatrix} C_1 & \cdots & C_{m-1} & C_{m+1} & & & & \\ & & & C_1 & \cdots & C_{m-1} & C_{m+1} & \\ & & & & \ddots & \ddots & \ddots & \\ C_{m+1} & & & & & C_1 & \cdots & C_{m-1} \end{bmatrix}$$

and B is an $Lp \times Lp$ matrix

$$B = \begin{bmatrix} C_m & & & \\ & C_m & & \\ & & \ddots & \\ & & & C_m \end{bmatrix}.$$

A corresponding generator matrix in systematic form is

$$G_{L;m} = \left[\; I_{L(m-1)p} \;\middle|\; Q \; \right],$$

where Q is an $L(m-1)p \times Lp$ matrix consisting of L quasi-cyclic downwards shifts by $(m-1)p$ positions of the matrix column

$$\begin{bmatrix} (C_m^{-1}C_1)^T \\ (C_m^{-1}C_2)^T \\ \vdots \\ (C_m^{-1}C_{m-1})^T \\ (C_m^{-1}C_{m+1})^T \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}.$$

For example, with tail-biting level $L = 2$ and $m = 2$ we obtain the parity-check matrix

$$H_{2;2} = \left[\begin{array}{cc|cc} C_1 & C_3 & C_2 & 0 \\ C_3 & C_1 & 0 & C_2 \end{array}\right]$$

and the generator matrix

$$G_{2;2} = \left[\begin{array}{cc|cc} I & 0 & (C_2^{-1}C_1)^T & (C_2^{-1}C_3)^T \\ 0 & I & (C_2^{-1}C_3)^T & (C_2^{-1}C_1)^T \end{array}\right],$$

and with $L = 3$ and $m = 3$ we obtain

$$H_{3;3} = \left[\begin{array}{cccccc|ccc} C_1 & C_2 & C_4 & 0 & 0 & 0 & C_3 & 0 & 0 \\ 0 & 0 & C_1 & C_2 & C_4 & 0 & 0 & C_3 & 0 \\ C_4 & 0 & 0 & 0 & C_1 & C_2 & 0 & 0 & C_3 \end{array}\right],$$

$$G_{3;3} = \left[\begin{array}{cccccc|ccc} I & 0 & 0 & 0 & 0 & 0 & (C_3^{-1}C_1)^T & 0 & (C_3^{-1}C_4)^T \\ 0 & I & 0 & 0 & 0 & 0 & (C_3^{-1}C_2)^T & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & (C_3^{-1}C_4)^T & (C_3^{-1}C_1)^T & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 & (C_3^{-1}C_2)^T & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 & (C_3^{-1}C_4)^T & (C_3^{-1}C_1)^T \\ 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & (C_3^{-1}C_2)^T \end{array}\right].$$

The standard QC-MDPC codes in the McEliece variant described in Section 2.3 can be replaced with these unit memory QC-MDPC codes, without having to alter the scheme in any significant way. The convolutional and quasi-cyclic structure give very compact representations of these matrices. The generator matrix, which is the public key, can be

represented by one column of the non-trivial part of matrix Q, which has length mp. The parity-check matrix, which is the private key, can be represented by the first rows of the C_i, for $i = 1, \ldots, m+1$, having total length $(m+1)p$. Note that increasing the tail-biting level L does not increase the key lengths.

Given the values L, m and p, other code parameters are:

- code (and ciphertext) length $n = Lmp$;
- code dimension $k = L(m-1)p$;
- code rate $R = (m-1)/m$;
- parity-check matrix height $r = Lp$.

The standard QC-MDPC codes described in Section 2.3 can be considered a special case of this construction with tail-biting level $L = 1$. For the special case of $L = 1$, only difference in parameters is that the private key length is $(m-1)p$ and the public key length is mp, due to the generator and parity-check matrices not containing the circulant matrix C_{m+1}.

4.3 Security

We will extend the security analysis provided in Section 2.4 to the tail-biting unit memory convolutional QC-MDPC code construction. We start with message security.

Recall that for QC-MDPC codes with matrix height $r = p$, we could produce p syndrome decoding instances from a single ciphertext, due to the quasi-cyclic structure. This gave an ISD speedup of $\sqrt{p}$. For the unit memory QC-MDPC codes we can still construct p syndrome decoding instances. But as some of the block columns of the parity-check matrices $H_{L;m}$ contain both C_1 and C_{m+1}, there is no obvious way to produce more than p instances.

One option for performing a message recovery attack would be to simply apply the same attack as described in the Section 2.4 to recover the whole matrix $H_{L;m}$. But due to the structure of the generator and parity-check matrices, an alternative approach is possible. For example, consider a generator matrix of a code with $L = 3$ and $m = 2$ given in Figure 3.

Denote the submatrix corresponding to the highlighted part of $G_{3;2}$ in Figure 3 as G'_{sub}, which is a valid generator matrix of a standard QC-MDPC code with $m = 3$. Assume

$$G_{3;2} = \begin{bmatrix} I & 0 & 0 & (C_2^{-1}C_1)^T & 0 & (C_2^{-1}C_3)^T \\ 0 & I & 0 & (C_2^{-1}C_3)^T & (C_2^{-1}C_1)^T & 0 \\ 0 & 0 & I & 0 & (C_2^{-1}C_3)^T & (C_2^{-1}C_1)^T \end{bmatrix}$$

Figure 3. Generator matrix of an $L = 3, m = 2$ unit memory QC-MDPC code, with the highlighted part corresponding to a generator matrix of a standard QC-MDPC code with $m = 3$.

that a message $\mathbf{u}$ of length $3p$ was encoded with $G_{3;2}$ into a codeword $\mathbf{c}$ of length $6p$. As $G_{3;2}$ contains only zeros below the highlighted part, the first, second and fourth p-bit blocks of $\mathbf{c}$ are the same as if the first 2 p-bit blocks of $\mathbf{u}$ had been encoded with G'_{sub}. Because of this, it is possible to recover a part of the message by only considering G'_{sub}. Remaining parts of the message could be recovered by constructing a matrix similar to G'_{sub}, except by using the fifth or sixth block column of $G_{3;2}$.

The workfactor of the key recovery attack over a whole generator matrix $G_{L;m}$ is

$$\frac{2^{t \log_2 \frac{1}{1-R}}}{\sqrt{p}} = \frac{2^{t \log_2 m}}{\sqrt{p}}. \tag{1}$$

Generator matrix $G_{L:m}$ represents a code of length $n = Lmp$ and rate $R = (m-1)/m$. A submatrix G_{sub} constructed from $G_{L:m}$ in the manner described above, is a generator matrix of a code of length $(m+1)p$ and rate $m/(m+1)$. Thus, the parts of the ciphertext corresponding to G_{sub} contain on average

$$\frac{m+1}{Lm}t$$

errors, where t is the total amount of errors inserted into the ciphertext. The workfactor of the key recover attack over G_{sub} and the corresponding ciphertext parts is

$$\frac{2^{\frac{m+1}{Lm}t \log_2 \frac{1}{1-\frac{m}{m+1}}}}{\sqrt{p}} = \frac{2^{t \frac{m+1}{Lm} \log_2(m+1)}}{\sqrt{p}}. \tag{2}$$

For us, the relevant ranges of L and m are $2 \leq L \leq 5$ and $2 \leq m \leq 5$ respectively. For $L = 2$ and $m = 2$, the quantity (1) is smaller than the quantity (2). For $L > 2$ or $m > 2$ in the relevant range, the quantity (2) is smaller than the quantity (1), which

can be verified by calculating the quantities (1) and (2) for each m and L combination. When choosing security parameters, this discrepancy needs to be taken into account.

Message security of the unit memory convolutional QC-MDPC code construction is relatively simple to analyze. Consider again a generator matrix $G_{L;m}$ and a generator matrix G_{sub} constructed from it as before. The parity-check matrix corresponding to G_{sub} is

$$H_{\text{sub}} = \begin{bmatrix} C_1 & C_2 & \cdots & C_{m-1} & C_{m+1} & C_m \end{bmatrix}$$

which is, again, a valid parity-check of a standard QC-MDPC code. H_{sub} contains the all the circulant matrices C_i, for $i = 1, \ldots, m+1$, that $H_{L;m}$ does. This means that $H_{L;m}$ can be constructed from H_{sub}, and H_{sub} can be constructed from $H_{L;m}$. As the row weights of $H_{L;m}$ and H_{sub} are the same, recovering one is equivalent to recovering the other. Recall that the rate of the code corresponding to G_{sub} and H_{sub} is $m/(m+1)$. Thus, based on the message recovery attack workfactor given in Section 2.4, the workfactor for the tail-biting unit memory convolutional QC-MDPC codes can be expressed as

$$\frac{2^{w \log_2 \frac{1}{\frac{m}{m+1}}}}{p} = \frac{2^{w \log_2 \frac{m+1}{m}}}{p}.$$

As previously stated, to obtain a λ-bit security level, the workfactors of the message and key recovery attacks need to be at least 2^λ.

4.4 Sliding window decoding

Sliding-window decoding [54, 55] is one possible method of decoding LDPC convolutional codes. Sliding-window decoders consider only a part of the received word and parity-check matrix at one time. That part is referred to as decoding window. After one part is decoded, the window *slides* forward to the next (possibly overlapping) part.

The authors of the unit memory QC-MDPC code construction propose a sliding-window decoding variant for decoding these codes. The circulant blocks C_i, for $i = 1, \ldots, m+1$, in each row of a parity-check matrix constitute a single decoding window. The number of such windows is thus equal to the tail-biting level L. For example, for an $L = 3$,

$m = 2$ code, the decoding windows are as indicated on Figure 4. The part highlighted in green is the first window, in blue is the second window and in red is the third window.

$$H_{3;2} = \begin{bmatrix} C_1 & C_3 & 0 & C_2 & 0 & 0 \\ 0 & C_1 & C_3 & 0 & C_2 & 0 \\ C_3 & 0 & C_1 & 0 & 0 & C_2 \end{bmatrix}$$

Figure 4. Decoding windows for an $L = 3$, $m = 2$ code.

The sliding-window decoder starts by considering the first window of a parity-check matrix and the corresponding part of a received word. Then some standard MDPC code decoder is applied to it, for example one of the bit-flipping decoders described in Chapter 3. The part of the received word corresponding to the current window is replaced by the word returned by the window decoder. After that, the algorithm moves to the next window and the same steps repeat. Depending on configuration, sliding-window decoder can stop after decoding each window once or it can return to the first window and keep going.

The sliding-window decoder is expressed in Algorithm 5. Extracting the part of the parity-check matrix H or the part of the word $\mathbf{c}$ which corresponds to the w-th window is done by the function `extractWindow`. The function `decodeWindow`, which is a parameter of the algorithm, perform decoding on the individual windows. The function `replaceWindow` replaces the word $\mathbf{c}_w$ into the whole word $\mathbf{c}$ on the positions corresponding to w-th window. The parameter $GlobalIter$ specifies how many global iterations of decoding are performed. That is, how many window decoding attempts are performed in total. The global iteration count should be at least as big as the number of different decoding windows.

Algorithm 5: Sliding-window decoder

Input: $H \in \mathbb{F}_2^{r \times n}$ (parity-check matrix), $L \in \mathbb{N}$ (window count/tail-biting level), $\mathbf{y} \in \mathbb{F}_2^n$ (received word), $GlobalIter \in \mathbb{N}$ (global iterations), `decodeWindow` (function)

Output: $\mathbf{c} \in \mathbb{F}_2^n$

$\mathbf{c} = \mathbf{y}$
for $gIter = 1 \ldots GlobalIter$ **do**
  $w = (gIter \bmod L) + 1$
  $H_w = \texttt{extractWindow}(H, w)$
  $\mathbf{c}_w = \texttt{extractWindow}(\mathbf{c}, w)$
  $\mathbf{c}'_w = \texttt{decodeWindow}(H_w, \mathbf{c}_w)$
  $\mathbf{c} = \texttt{replaceWindow}(\mathbf{c}, w, \mathbf{c}_w)$
end
return $\mathbf{c}$

If decoding of one window is successful, then the preceding and the following windows become easier to decode. This is because each decoding window overlaps with the preceding window on the column containing C_1, and with the following window on the column containing C_{m+1}. If a window was not completely decoded, then it might be worth returning to it after having decoded some of the overlapping windows, because the corrected errors in the overlapping parts might allow to finish decoding the original window.

To determine the optimal amount of global iterations, we performed the following experiment. We simulated the performance of the sliding window decoder on an $L = 3, m = 2$ code, with global iteration counts 3, 4, 5 and 6, which can be thought of as L, $L + 1$, $2L - 1$ and $2L$ iterations respectively. We used Gallager's bit-flipping algorithm on each window with the threshold being the maximum number of unsatisfied parity-checks. The row weight w of the parity-check matrix was set to135 and the inserted error count t was set to 89, which correspond to a security level of 64 bits. The results are given in Figure 5, where each data point was simulated with 25 random codes and 100 ciphertexts per code.

As it can be seen from Figure 5, returning to the first decoding window after having performed decoding on each window once, does decrease the overall DFR. This can be explained by the first window being the hardest to decode during the first attempt, which is because there is no advantage from the overlapping windows having decreased error counts, as they have not been decoded yet. We can also see from the results that increasing the global iteration count past $L + 1$ decreases the DFR only by an insignificant amount, which is not worth the additional complexity. Thus, $L + 1$ global iterations seems to be the optimal choice.

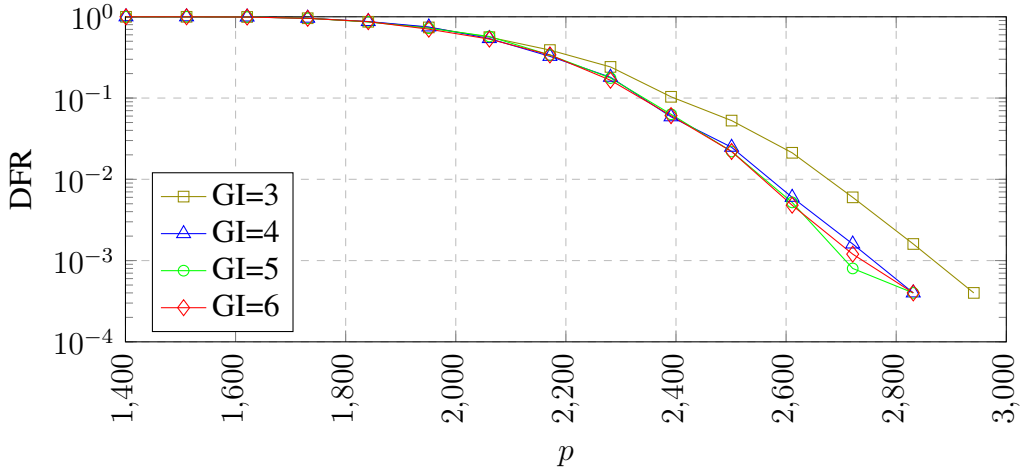

Figure 5. Performance of different global iteration counts for the sliding-window decoder at different block sizes p.

4.5 Comparison of code constructions

To compare different code constructions in the setting of the McEliece cryptosystem, we compare them based on what properties they have at the same security level. This allows to choose the most suitable code type for the target setting. We start by comparing different combinations of the parameters L and m for the unit memory QC-MDPC codes. In the end, we compare the unit memory convolutional codes to standard QC-MDPC codes.

4.5.1 Changing circulant block count

We start our analysis by comparing codes with different values of m. The value m indicates the ratio of parity-check matrix width (which is also the code length) to parity-check matrix height. We performed simulations for codes with tail-biting level $L = 3$ and m values 2, 3, 4 and 5. The parity-check matrix row weights and inserted error counts were chosen such that they give a security level of 64 bits. Decoding was performed with the sliding-window decoder using Gallager's bit-flipping as a window decoder. The simulations results are presented in Figure 6, as a comparison between the DFR and code length n. Each data point was simulated with 25 random codes and 100 ciphertexts per code.

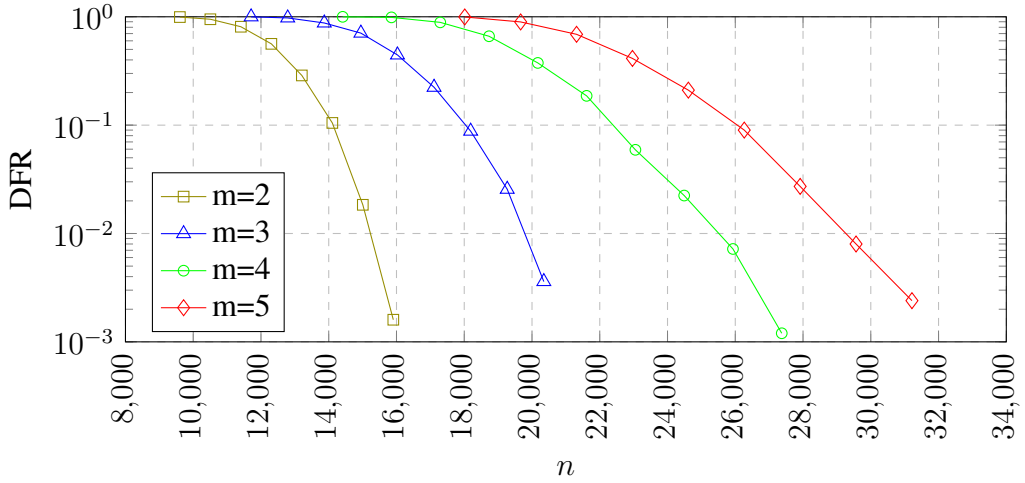

Figure 6. Comparison of the DFR at different code lengths for codes with different m values.

To get the block size p corresponding to a security level of 64 bits, the DFR would have to be extrapolated to 2^{-64}. For any extrapolation technique to give a reliable estimate, data points at lower DFR levels would have to be simulated and musch more extensive simulations would have to be run. Assuming that the relative difference between codes with different m values would be similar at high and low DFR levels, we compare them only at DFR = 10^{-2} level. The comparison is given in Table 1.

Table 1. Parameters for the compared codes at DFR $= 10^{-2}$.

L	m	w	t	R	p	n	public key length
3	2	135	89	0.5	2539	15234	5078
3	3	188	79	0.67	2199	19791	6597
3	4	245	73	0.75	2127	25524	8508
3	5	294	68	0.8	1947	29205	9735

We see from Table 1 that increasing m leads to the disadvantage of increased ciphertext lengths and key sizes. An advantage of higher m values is the increased code rate, which means that the ciphertext has a smaller ratio of redundant information.

4.5.2 Changing tail-biting level

Next we compare tail-biting convolutional QC-MDPC codes with different tail-biting levels L. We compare codes with tail-biting levels of 2, 3, 4 and 5, with m equal to 2 for all of them. Again, inserted error counts and parity-check matrix row weights were chosen to correspond to 64-bit security. Simulation results are given in Figure 7, where each data point was simulated with 50 codes and 100 ciphertexts per code.

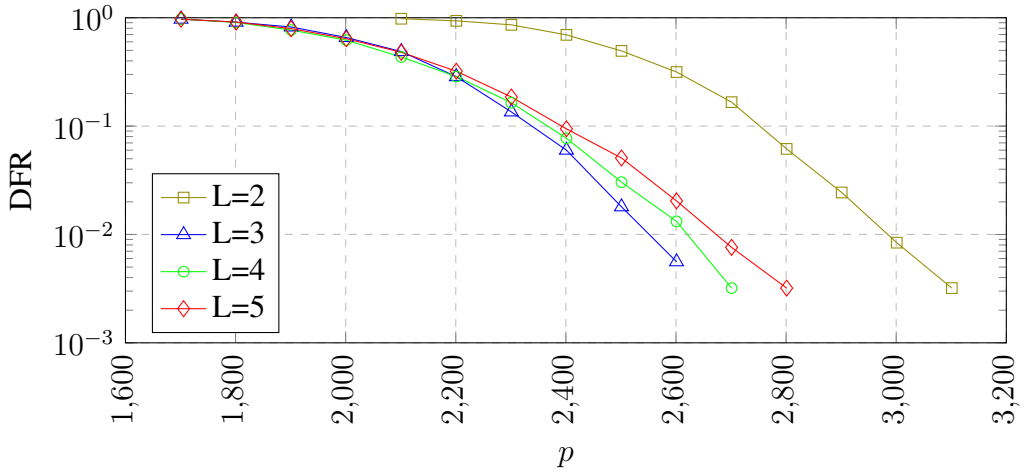

Figure 7. Comparison of the DFR at different code lengths for codes with different L values.

The reason that the codes with $L = 2$ and $m = 2$ require higher block sizes is due to the relatively larger inserted error count, which is due to the security considerations explained in Section 4.3. An overview of the parameters for codes with different tail-biting levels is given in Table 2. As before, we have taken the values based on the DFR level of 10^{-2}.

Table 2. Parameters for the compared codes at DFR $= 10^{-2}$.

L	m	w	t	R	p	n	public key length
2	2	135	71	0.5	2981	11924	5962
3	2	135	89	0.5	2553	15318	5103
4	2	135	118	0.5	2621	20968	5242
5	2	135	148	0.5	2667	26670	5334

Data in Table 2 indicates that the combination of $L = 2$ and $m = 2$ is not optimal for decreased key lengths, but in general, increasing the tail-biting level leads to an

increased in key lengths. Higher tail-biting levels also mean larger ciphertext lengths.

4.5.3 Comparison with standard QC-MDPC codes

Finally, we compare the new tail-biting unit memory convolutional QC-MDPC codes with the standard QC-MDPC codes. The standard QC-MDPC codes were decoded using Gallager's bit-flipping algorithm with the threshold being the maximum number of unsatisfied parity-checks. The same algorithm was used as the window decoder for tail-biting convolutional codes. Inserted error count and parity-check matrix row weight were chosen to correspond to security level of 64-bits. The results are presented in Table 3, where $L = 1$ indicates standard QC-MDPC code. The block length p, code length n and public key length correspond to a DFR level of 10^{-3}. This was determined by simulating data points with 200 codes and 100 ciphertexts per code.

Table 3. Parameters for the convolutional and standard QC-MDPC construction at DFR $= 10^{-3}$.

L	m	w	t	R	p	n	public key length	key to n ratio
1	2	78	71	0.5	3291	6582	3291	0.5
1	3	135	45	0.67	2463	7389	4926	0.67
2	2	135	71	0.5	3113	12452	6226	0.5
3	2	135	89	0.5	2705	16230	5410	0.33
2	3	188	53	0.67	2311	13866	6933	0.5

We can see from the results in Table 3, that adding the convolutional structure can decrease the required block size p for codes of the same rate R. However, this does not lead to a decrease in public key lengths. The convolutional construction also requires longer ciphertext lengths, although it also mean longer message lengths, as adding the convolutional structure does not decrease the code rate. An advantage of higher L values is that the public key length is relatively smaller to the ciphertext length, as can be seen from the construction with $L = 3$ in the table. Increasing the tail-biting level to higher values would reduce this ratio further, as the key size does not increase with tail-biting level. This could be useful in setting where ephemeral keys (single use keys) are used to transmit large amounts of data, as the key would take up a smaller part of the amount of transmitted data. The standard QC-MDPC construction would still be better in cases where short messages need to be encrypted, for example when encrypting symmetric cryptography keys in a hybrid cryptosystem.

5 Conclusion

The goal of the thesis was to study methods to improve the QC-MDPC code based McEliece cryptosystem. Thesis started with an overview of the relevant coding theory background. The McEliece cryptosystem and its variant based on QC-MDPC codes were defined, and both classical and post-quantum security of the latter was explained.

Challenges and methods for decoding of MDPC codes in the cryptographic setting were explained. Three existing decoders were described: Gallager's original bit-flipping decoder, a state of the art decoder Black-Gray-Flip and a novel weighted bit-flipping decoder. The performance of the weighted bit-flipping decoder and the Black-Gray-Flip decoder was compared. The result demonstrated that for some code parameters the weighted bit-flipping decoder can perform better than Black-Gray-Flip decoder, while maintaining a reasonable computational complexity. These results suggest that the weighted bit-flipping decoder is worth further study.

Additionally, a new tail-biting convolution QC-MDPC code construction, in application to the QC-MDPC code based McEliece variant, was desribed and its security was analyzed. A sliding-window decoder, which is suitable for decoding these convolutional codes, was presented. Various code parameter combinations were considered, and the new convolutional codes were compared to the standard QC-MDPC codes. It was shown that new codes can be the better choice in settings where ephemeral keys are used. In the future, more extensive experiments could be performed to confirm and extend the results presented in this thesis.

References

[1] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. USA: IEEE Computer Society, 1994, pp. 124–134.

[2] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* 21.2, 1978, pp. 120–126.

[3] D. Kravitz. Digital signature algorithm. US patent 5231668. 1991.

[4] R. McEliece. A public key cryptosystem based on algebraic coding theory. *The Deep Space Network Progress Report*. Vol. 44. 1978, pp. 114–116.

[5] R. Misoczki, J. Tillich, N. Sendrier, and P. S. L. M. Barreto. MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes. *IEEE International Symposium on Information Theory*. 2013, pp. 2069–2073.

[6] Post-Quantum Cryptography, Round 3 Submissions. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions (accessed 2021-05-12).

[7] HQC project homepage. URL: http://pqc-hqc.org/ (accessed 2021-05-12).

[8] BIKE project homepage. URL: https://bikesuite.org/ (accessed 2021-05-12).

[9] A. Nilsson, I. Bocharova, B. Kudryashov, and T. Johansson. A Weighted Bit Flipping decoder for QC-MDPC-based Cryptosystems. unpublished.

[10] I. Bocharova and B. Kudryashov. Tail-biting 2D-convolutional MDPC codes with improved decoding in the McEliece public key cryptosystem. unpublished.

[11] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal* 27.3, 1948, pp. 379–423.

[12] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory* 8.1, 1962, pp. 21–28.

[13] S. Ouzan and Y. Be'ery. Moderate-Density Parity-Check Codes. *CoRR* abs/0911.3262, 2009. arXiv: 0911.3262.

[14] D. J. Bernstein, T. Chou, and P. Schwabe. McBits: Fast Constant-Time Code-Based Cryptography. *Cryptographic Hardware and Embedded Systems - CHES*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 250–272.

[15] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.) *IEEE Transactions on Information Theory* 24.3, 1978, pp. 384–386.

[16] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing Key Length of the McEliece Cryptosystem. *Progress in Cryptology – AFRICACRYPT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 77–97.

[17] R. Misoczki and P. S. L. M. Barreto. Compact McEliece Keys from Goppa Codes. *Selected Areas in Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 376–392.

[18] J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys. *Advances in Cryptology – EUROCRYPT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 279–298.

[19] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. *IEEE International Symposium on Information Theory*. 2000, p. 215.

[20] M. Baldi, F. Chiaraluce, and R. Garello. On the Usage of Quasi-Cyclic Low-Density Parity-Check Codes in the McEliece Cryptosystem. *2006 First International Conference on Communications and Electronics*. 2006, pp. 305–310.

[21] N. Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory* 21.2, 1975, pp. 203–207.

[22] K. Kobara and H. Imai. Semantically Secure McEliece Public-Key Cryptosystems -Conversions for McEliece PKC -. *Public Key Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 19–35.

[23] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory* 8.5, 1962, pp. 5–9.

[24] J. Stern. A method for finding codewords of small weight. *Coding Theory and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 106–113.

[25] A. May, A. Meurer, and E. Thomae. Decoding Random Linear Codes in $O(2^{0.054n})$. *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security*. Seoul, South Korea: Springer-Verlag, 2011, pp. 107–124.

[26] A. Becker, A. Joux, A. May, and A. Meurer. Decoding Random Binary Linear Codes in $2^{n/20}$: How 1 + 1 = 0 Improves Information Set Decoding. *Advances in Cryptology – EUROCRYPT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 520–536.

[27] R. Canto Torres and N. Sendrier. Analysis of Information Set Decoding for a Sublinear Error Weight. *Post-Quantum Cryptography - PQCrypto*. Fukuoka, Japan, 2016.

[28] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Ghosh, S. Gueron, T. Güneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zémor. BIKE: Bit Flipping Key Encapsulation, Round 3 Submission. 4.1. 2020.

[29] N. Sendrier. Decoding One Out of Many. *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 51–67.

[30] Q. Guo, T. Johansson, and P. Stankovski. A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors. *Advances in Cryptology – ASIACRYPT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 789–815.

[31] A. Nilsson, T. Johansson, and P. Wagner. Error Amplification in Code-based Cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018, pp. 238–258.

[32] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray. Advanced Encryption Standard (AES). Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[33] X. Bogomolec, J. G. Underhill, and S. A. Kovac. Towards Post-Quantum Secure Symmetric Cryptography: A Mathematical Perspective. *IACR Cryptol. ePrint Arch.* 2019, 2019, p. 1208.

[34] L. K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219.

[35] D. J. Bernstein. Grover vs. McEliece. *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 73–80.

[36] Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. National Institute of Standards and Technology. 2016. URL: csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf (accessed 2021-05-12).

[37] N. Wiberg. Codes and Decoding on General Graphs. PhD thesis. Linköping University, 1996.

[38] E. Eaton, M. Lequesne, A. Parent, and N. Sendrier. QC-MDPC: A Timing Attack and a CCA2 KEM. *Post-Quantum Cryptography*. Cham: Springer International Publishing, 2018, pp. 47–76.

[39] N. Sendrier and V. Vasseur. On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders. *Post-Quantum Cryptography*. Cham: Springer International Publishing, 2019, pp. 404–416.

[40] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Güneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zémor. BIKE: Bit Flipping Key Encapsulation, Round 2 Submission. 3.0. 2019.

[41] N. Sendrier and V. Vasseur. About Low DFR for QC-MDPC Decoding. *Post-Quantum Cryptography*. Cham: Springer International Publishing, 2020, pp. 20–34.

[42] N. Ducker and S. Gueron. Additional implementation of "BIKE: Bit Flipping Key Encapsulation". https://bikesuite.org/additional.html. 2017.

[43] N. Drucker, S. Gueron, and D. Kostic. On Constant-Time QC-MDPC Decoders with Negligible Failure Rate. *Code-Based Cryptography*. Cham: Springer International Publishing, 2020, pp. 50–79.

[44] N. Drucker, S. Gueron, and D. Kostic. QC-MDPC Decoders with Several Shades of Gray. *Post-Quantum Cryptography*. Cham: Springer International Publishing, 2020, pp. 35–50.

[45] Y. Kou, S. Lin, and M. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Transactions on Information Theory* 47.7, 2001, pp. 2711–2736.

[46] J. Zhang and M. Fossorier. A Modified Weighted Bit-Flipping Decoding of Low-Density Parity-Check Codes. *Communications Letters, IEEE* 8, Apr. 2004, pp. 165–167.

[47] M. Jiang, C. Zhao, Z. Shi, and Y. Chen. An improvement on the modified weighted bit flipping decoding algorithm for LDPC codes. *IEEE Communications Letters* 9.9, 2005, pp. 814–816.

[48] G. Dong, Y. Li, N. Xie, T. Zhang, and H. Liu. Candidate bit based bit-flipping decoding algorithm for LDPC codes. *IEEE International Symposium on Information Theory*. 2009, pp. 2166–2168.

[49] C. Löndahl and T. Johansson. A New Version of McEliece PKC Based on Convolutional Codes. *Information and Communications Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 461–470.

[50] P. Almeida, M. Beltrá, D. Napp, and C. Sebastião. Smaller Keys for Code-Based Cryptography: McEliece Cryptosystems with Convolutional Encoders. 2021. arXiv: 2104.06809 [cs.IT].

[51] J. Bolkema, H. Gluesing-Luerssen, C. A. Kelley, K. E. Lauter, B. Malmskog, and J. Rosenthal. Variations of the McEliece Cryptosystem. *Algebraic Geometry for Coding Theory and Cryptography*. Cham: Springer International Publishing, 2017, pp. 129–150.

[52] G. Landais and J.-P. Tillich. An Efficient Attack of a McEliece Cryptosystem Variant Based on Convolutional Codes. *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 102–117.

[53] A. Jimenez Felstrom and K. Zigangirov. Time-varying periodic convolutional codes with low-density parity-check matrix. *IEEE Transactions on Information Theory* 45.6, 1999, pp. 2181–2191.

[54] M. Papaleo, A. R. Iyengar, P. H. Siegel, J. K. Wolf, and G. E. Corazza. Windowed erasure decoding of LDPC Convolutional Codes. *IEEE Information Theory Workshop on Information Theory*. 2010, pp. 1–5.

[55] I. E. Bocharova, B. D. Kudryashov, E. Rosnes, V. Skachek, and Ø. Ytrehus. Wrap-around sliding-window near-ML decoding of binary LDPC codes over the BEC. *9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*. 2016, pp. 16–20.

Appendix

I. Source code

The source code used for simulations was implemented in MATLAB and can be found in a Git repository at the address https://github.com/ENigola/masters-thesis

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Ergo Nigola**,
*(*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,

   MDPC Code-Based Constructions and Their Decoding in Post-Quantum Cryptosystems,
   *(*title of thesis)

   supervised by Irina Bocharova and Vitaly Skachek.
   *(*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.

Ergo Nigola
14/05/2021