MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes

Rafael Misoczki[1] and Jean-Pierre Tillich[1] and
Nicolas Sendrier[1] and Paulo S. L. M. Barreto[2]

[1] Project SECRET, INRIA-Rocquencourt, France
[2] Escola Politécnica, Universidade de São Paulo, Brazil

Abstract. In this work, we propose two McEliece cryptosystem variants: one from Moderate Density Parity-Check (MDPC) codes and another from quasi-cyclic MDPC codes. MDPC codes are LDPC codes of higher density than what is usually adopted for telecommunication applications. In general, this leads to a worse error-correction capability. However, in code-based cryptography we are not necessarily interested in correcting many errors, but only a number which ensures an adequate security level, a condition satisfied by MDPC codes. The benefits of their employment are many. Under a reasonable assumption, MDPC codes reduce the key-distinguishing McEliece problem to the problem of decoding linear codes. Since the message-attacks against the McEliece scheme also reduce to this problem, the security of our scheme has the benefit of relying on a single, well studied coding-theory problem. Furthermore, adding a quasi-cyclic structure, our proposal provides extremely compact-keys: for 80-bits of security, the public-key has only 4801 bits.

Keywords: post-quantum cryptography, code-based cryptography, coding-theory, LDPC codes.

1 Introduction

All cryptosystems based on the hardness of factoring or discrete logarithm can be attacked [39] in polynomial time with a quantum computer (see [10] for an extensive report). This threatens most if not all public-key cryptosystems deployed in practice, such as RSA [36] or DSA [24]. Code-based cryptography is believed to be quantum resistant and is therefore considered as a viable replacement for those schemes in future applications. Yet, independently of their so-called "post-quantum" nature, code-based cryptosystems offer other benefits even for present-day applications due to their excellent algorithmic efficiency, which is up to several orders of complexity better than traditional schemes.

The McEliece cryptosystem [28] is the first code-based cryptosystem, originally proposed using Goppa codes. Its security is based on two assumptions, the indistinguishability of the code family and the hardness of decoding a generic linear code [13]. The decoding problem is a well studied NP-complete problem [8], believed to be hard after decades of research. On the other hand, the indistinguishability problem is usually the weakest one, strongly depending on

the choice of the code family. As an example of this fragility, a distinguisher for high rate Goppa codes (like those originally suggested for CFS signature [13] and for some realistic secure parameters of McEliece cryptosystems) is presented in [15]. Although this does not represent a practical attack, it suggests that Goppa codes do not seem to be the optimal choice for code-based cryptography.

Although efficient, this cryptosystem suffers from an extremely large key-size. There is a way to reduce considerably the key-size which consists in choosing codes with a large automorphism group, such as quasi-cyclic codes [19]. It has been followed by several other proposals such as [29, 7]. However, an structural algebraic attack [16] succeeds in breaking many of them (except the binary case of [29]). The effectiveness of this attack is due to the strong algebraic structure of the suggested code-families (they are subfamilies of alternant codes), which allows the adversary to set up an algebraic equations system and solve it with Gröbner bases techniques. This algebraic system has several features that make this computation feasible: the system is bihomogeneous and bilinear and, most importantly, the quasi-cyclic or the quasi-dyadic structure of these schemes allows a drastic reduction of the number of unknowns in the system. This kind of attack is exponential in nature and can be easily prevented by choosing more conservative parameters. Note however that codes which does not have an algebraic structure would completely prevent this threat.

Related work. Low-Density Parity Check (LDPC) codes [20] are good candidates for this purpose. They are codes with no algebraic structure which meet a very simple combinatorial property: they admit a sparse parity-check matrix. This sparsity is used by decoding algorithms for efficient error-correction. These codes have been repeatedly suggested for the McEliece scheme [30, 4, 5, 3, 2]. However, the main problem of using LDPC codes in this context is that their low weight parity-check rows can be seen as low weight codewords in the dual of the public code. Thus a straightforward attack against an LDPC-McEliece variant amounts to find dual low weight codewords and use them to build a sparse parity-check matrix. This is the conclusion of [30], where the LDPC-McEliece variant is analyzed: the private-key is a sparse parity-check matrix H of constant row weight w of a code $\mathcal{C}$ and the public-key is a dense generator matrix $G' = S \cdot G \cdot P$ of a code $\mathcal{C}'$, where S is a scrambling matrix, G is a generator matrix for $\mathcal{C}$ and P is a permutation matrix. Indeed, for usual LDPC parameters, finding low weight codewords in the dual of $\mathcal{C}'$ is feasible. In [3], a proposal to fix this problem is suggested. It consists in replacing the permutation matrix P by an invertible matrix Q of some small constant row weight m and in choosing S sparse. For properly chosen w and m, the task of finding codewords of weight wm in $\mathcal{C}'$ becomes unfeasible. Nevertheless, the unfortunate choices for the structure of these matrices allowed to successfully cryptanalyze the scheme [32]. In [2], an improved variant suggests a dense matrix S and a more general construction for Q, and it seems to be immune to the attack of [32]. The authors also propose a quasi-cyclic variant with compact keys of 48384 bits[3], for 80-bits of security.

[3] Note that the authors did not consider the use of CCA-2 security conversions, which would allow public-keys in systematic form reducing the key-size to 12096 bits.

Our contribution. Our first observation is that none auxiliary matrix of constant row weight (e.g. the matrix Q of [2]) is needed to instantiate the McEliece scheme with LDPC codes. Simply increasing moderately the length and the row weight of the secret sparse parity-check matrix is enough to avoid all known message attacks (based on standard decoding algorithms) and key recovery attacks (aiming at finding low weight codewords in the dual of the public code). We call these codes Moderate Parity Check (MDPC) codes[4] to insist on the fact that they admit a parity-check which is only moderately sparse. Although this leads to a significantly degraded error correction performance (when compared to standard LDPC codes), it is still sufficiently good to prevent the effectiveness of standard decoding algorithms. Note that our proposal is scalable for any security level and code rate.

We also give a quite satisfactory security reduction towards the well studied syndrome decoding problem. To achieve this goal, we make a single, natural assumption: distinguishing an MDPC code from a random linear code amounts to being able to ascertain the existence of low weight codewords in its dual code. This provides a strong argument in favor of the security of our scheme. Furthermore, adding a quasi-cyclic structure, our proposal provides extremely compact keys: for 80-bits of security, the public-key has only 4801 bits.

2 Preliminaries

We gather here a few basic definitions which are used in this paper.

Definition 1 (Hamming distance and weight). *The Hamming weight (or simply weight) of a vector $x \in \mathbb{F}_2^n$ is the number $\mathsf{wt}(x)$ of its nonzero components.*

Definition 2 (Linear codes). *A binary (n, r)-linear code $\mathcal{C}$ of length n, dimension $n-r$ and codimension r, is a $(n-r)$-dimensional vector subspace of $\mathbb{F}_2^n$. It is spanned by the rows of a matrix $G \in \mathbb{F}_2^{(n-r)\times n}$, called a* generator matrix *of $\mathcal{C}$. Equivalently, it is the kernel of a matrix $H \in \mathbb{F}_2^{r\times n}$, called a* parity-check matrix *of $\mathcal{C}$. The* codeword *$c \in \mathcal{C}$ of a vector $m \in \mathbb{F}_2^{(n-r)}$ is $c = mG$. The* syndrome *$s \in \mathbb{F}_2^r$ of a vector $e \in \mathbb{F}_2^n$ is $s = He^T$. The* dual *$\mathcal{C}^\perp$ of $\mathcal{C}$ is the linear code spanned by the rows of any parity-check matrix of $\mathcal{C}$.*

Definition 3 (Quasi-cyclic code). *An (n, r)-linear code is quasi-cyclic (QC) if there is some integer n_0 such that every cyclic shift of a codeword by n_0 places is again a codeword.*

When $n = n_0 p$, for some integer p, it is possible and convenient to have both generator and parity check matrices composed by $p \times p$ circulant blocks. Note

[4] This terminology has already been proposed before in the communications theory literature for the very same concept [33]. The authors showed that certain quasi-cyclic MDPC codes may perform well at moderate lengths for correcting a rather large number of errors by using a variation of the standard belief propagation taking advantage of the quasi-cyclic structure.

that a circulant block is completely described by its first row (or column) and the algebra of $p \times p$ binary circulant matrices is isomorphic to the algebra of polynomials modulo $x^p - 1$ over $\mathbb{F}_2$, allowing efficient computations.

Definition 4 (LDPC/MDPC codes). *An (n, r, w)-LDPC or MDPC code is a linear code of length n, codimension r which admits a parity-check matrix of constant row weight w.*

LDPC and MDPC codes only differ in the magnitude of the row weight w. While LDPC codes have small constant row weights (usually less than 10), we assume for MDPC codes row weights which scale in $O(\sqrt{n \log n})$. When these codes are also quasi-cyclic, we call them (n, r, w)-QC-LDPC or QC-MDPC codes.

3 Moderate Density Parity-Check McEliece variants

In this section, we present the construction of MDPC and QC-MDPC codes, then the description of our McEliece variant (which can be instantiated either with an MDPC or a QC-MDPC code).

3.1 (n, r, w)-MDPC code construction.

A random (n, r, w)-MDPC code is easily generated by picking a random parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ of row weight w. With overwhelming probability this matrix is of full rank and the rightmost $r \times r$ block is always invertible after possibly swapping a few columns.

3.2 (n, r, w)-QC-MDPC code construction.

We are specially interested in (n, r, w)-QC-MDPC codes where $n = n_0 p$ and $r = p$. This means that the parity-check matrix has the form

$$H = [H_0 | H_1 | \dots | H_{n_0 - 1}],$$

where H_i is a $p \times p$ circulant block.

We define the first row of H picking a random vector of length $n = n_0 p$ and weight w. The other $r - 1$ rows are obtained from the $r - 1$ quasi-cyclic shifts of this first row. Each block H_i will have a row weight w_i, such that $w = \sum_{i=0}^{n_0 - 1} w_i$. In general, a smooth distribution is expected for the sequence of w_i's.

A generator matrix G in row reduced echelon form can be easily derived from the H_i's blocks. Assuming the rightmost block $H_{n_0 - 1}$ is non-singular (which particularly implies $w_{n_0 - 1}$ odd, otherwise the rows of $H_{n_0 - 1}$ would sum up to 0), we construct a generator-matrix as follows.

$$G = \left[\begin{array}{c|c} & (H_{n_0-1}^{-1} \cdot H_0)^T \\ & (H_{n_0-1}^{-1} \cdot H_1)^T \\ \mathrm{I} & \vdots \\ & (H_{n_0-1}^{-1} \cdot H_{n_0-2})^T \end{array}\right]$$

3.3 MDPC/QC-MDPC McEliece variant

1. *Key-Generation.*
   (a) Generate a parity-check matrix $H \in \mathbb{F}_2^{r \times n}$ of a t-error-correcting (n, r, w)-MDPC or (n, r, w)-QC-MDPC code.
   (b) Generate its corresponding generator matrix $G \in \mathbb{F}_2^{(n-r) \times n}$ in row reduced echelon form.

   The public-key is G and the private-key is H.

2. *Encryption.* To encrypt a plaintext $m \in \mathbb{F}_2^{(n-r)}$ into $x \in \mathbb{F}_2^n$:
   (a) Generate $e \in \mathbb{F}_2^n$ of $\mathsf{wt}(e) \leq t$ at random.
   (b) Compute $x \leftarrow mG + e$.

3. *Decryption.* Let Ψ_H be a t-error correcting LDPC decoding algorithm equipped with the knowledge of H. To decrypt $x \in \mathbb{F}_2^n$ into $m \in \mathbb{F}_2^{(n-r)}$:
   (a) Compute $mG \leftarrow \Psi_H(mG + e)$.
   (b) Extract the plaintext m from the first $(n - r)$ positions of mG.

Note that this description gets rid[5] of the usual scrambling matrix S and permutation matrix P. Note also that the use of a CCA2-secure conversion, e.g. [23], allows for G in systematic-form without leading to security-flaws. Thus the QC-MDPC variant has a public-key of size $(n - r)$ and the MDPC variant of size $r(n - r)$. In practice, the MDPC variant obtains huge keys whilst the QC-MDPC allows for extremely compact keys. Regarding the quasi-cyclic variant, note that the state of the art indicates that a quasi-cyclic structure, by itself, does not imply a significant improvement for adversaries. All previous attacks on compact-keys McEliece variants are based on the combination of a quasi-cyclic/dyadic structure with some *algebraic* code information.

4 Decoding MDPC codes

Our MDPC codes will be decoded with a variant of the Gallager's bit flipping algorithm [20]. This iterative decoding algorithm provides an error-correction capability which increases linearly with the code-length and decreases more or less linearly with the weight of the parity-checks. Thus, when moving from LDPC to MDPC codes, a degradation in the error-correcting capability is expected. However in cryptography we are not necessarily interested in correcting a large number of errors, but only a number which ensures an adequate security level.

Gallager's bit flipping algorithm works as follows. At each iteration, the number of unsatisfied parity-check equations associated to each bit of the message is computed. Each bit associated to more than b unsatisfied equations is flipped and the syndrome is recomputed. This process is repeated until either the syndrome

[5] A folklore reasoning assigns security functions to those matrices. However it is enough that the public-key does not reveal any useful information for decoding, a condition satisfied by the dense public matrix.

becomes zero or after a maximum number of iteration. It is easy to see that this algorithm has complexity $O(nwI)$, where I stands for the average number of iterations. Due to the increased row weight (and the existence of short-cycles in the corresponding Tanner graph), MDPC codes may lead to an increased number of iterations. To minimize this problem, we suggest a modification for choosing b. Below a few possibilities for this choice and our approach:

I. Precomputing a sequence of b's (see Inequality 4.16, pg. 46, of [20]).
II. In [22], at each iteration, b is chosen as the maximum number of unsatisfied parity-check equations, here denoted by Max_{upc}.
III. Our approach is: $b = \text{Max}_{\text{upc}} - \delta$, for a variable small integer δ.

The main feature of each approach is: Approach I uses an estimation for b and therefore avoids its computation at each iteration. Approach II is more general than I, leading to a better error-correcting capability at the price of an increased number of iterations. Finally, Approach III combines the benefits from I and II:

- It reduces the overall number of iterations obtained by Approach II because much more bits are flipped at each iteration.
- In the case of a decoding failure, we suggest to decrease the value of δ by 1 and to restart the process. Obviously, when $\delta = 0$, we are back to Approach II ensuring at least its error-correcting capability.

The optimal initial value for δ is determined empirically. For the parameters suggested in Section 6, a good choice is $\delta \approx 5$, reducing the number of iterations from ~ 65 to less than 10.

A final remark on this decoding algorithm: note that the value of Max_{upc} tends to decrease at each iteration. Another bit flipping variant might use this information to estimate the sequence of Max_{upc}'s, avoiding its computation at each iteration. However, since it is an estimation, this may increase the average number of iterations.

4.1 Error-correction capability estimation

To estimate the error correction capability of Gallager's bit-flipping algorithm for MDPC codes we begin with the Gallager's analysis presented in [20], which gives a threshold for the number of errors that an (n, r, w)-LDPC code may correct. In Appendix A, we describe this technique. Although this analysis is not quite precise for MDPC codes (due to the existence of short cycles in the associated Tanner graph), it provides an upper bound for its error correction capability. Alternatively, it is possible to estimate the quality of an MDPC code (in correcting a given number of errors) in terms of its *decoding failure rate* (DFR), which is the fraction of decoding failures in a given number of decoding tests. Thus a valid strategy for choosing parameters is to start with the theoretical upper-bound and decrease it until reaching an adequate DFR. Using this approach, we validate that the parameters of Section 6 reach a DFR below 10^{-7}.

4.2 Dealing with decoding failures

As discussed above, MDPC codes (like any other code that use probabilistic decoding techniques) admit a non-zero decoding failure probability. In cryptography, this must be treated. Next we present three approaches to deal with it.

A. A straightforward approach consists in conservatively choosing the number of errors so that the decoding failure rate becomes negligible. For example, a common approach in error-correcting systems consists in using codes whose DFR is smaller than the machine failure rate where the system is deployed.
B. A second approach deals with these unlikely events on the fly. In the case of a decoding failure, more sophisticated decoding algorithms with better error correction capability can be used, e.g. [21]. Note however that this comes at the price of a significantly increased decoding complexity.
C. When the application allows, a third approach consists in using a CCA-2 security conversion, e.g. [23]. In short, a CCA2-security conversion uses hash functions and random sequences to ensure the indistinguishability of the encrypted messages. Thus, in the case of a decoding failure, new encryptions can be requested. Since the encrypted messages behave like random sequences, the adversary cannot extract information from this redundancy.

5 Security Assessment

This section is divided into security reduction and practical security assessment.

5.1 Security reduction

By security reduction, we mean a proof that an adversary able to attack the scheme is able to solve some (presumably hard) computational problem with a similar effort. We start by giving the generic security reduction presented in [37] for the Niederreiter cryptosystem [31]. This scheme is equivalent in terms of security to the McEliece cryptosystem [25]. It is easy to see that this security reduction also holds for the McEliece scheme, at the price of more involved probability space and statements. After the generic security reduction, we provide the reduction regarding our proposal.

Notation:

- $\mathcal{F}_{n,r,w}$: a t-error correcting code family which can be either (n, r, w)-MDPC or (n, r, w)-QC-MDPC. We assume the public-key is a parity check matrix of some code in $\mathcal{F}_{n,r,w}$.
- $\mathcal{K}_{n,r,w}$: the key space of $\mathcal{F}_{n,r,w}$.
- $\mathcal{H}_{n,r} \supset \mathcal{K}_{n,r,w}$: the apparent key space of $\mathcal{F}_{n,r,w}$.
  - MDPC case: $\mathcal{H}_{n,r}$ is the set of all full rank matrices in $\mathbb{F}_2^{r\times n}$.
  - QC-MDPC case: $\mathcal{H}_{n,r}$ is the set of all full rank matrices in $\mathbb{F}_2^{r\times n}$, restricted to block circulant matrices.

Generic Reduction. Let $\mathcal{S}_n(0,t)$ denote the sphere centered in zero of radius t in the Hamming space $\mathbb{F}_2^n$ and let Ω denote the probability space consisting of the sample space $\mathcal{H}_{n,r} \times \mathcal{S}_n(0,t)$ equipped with a uniform distribution. We define:

Distinguisher. A program $\mathcal{D} : \mathcal{H}_{n,r} \longrightarrow \{0,1\}$ is a (T,ϵ)-distinguisher for $\mathcal{K}_{n,r,w}$ (*vs.* $\mathcal{H}_{n,r}$) if it runs in time at most T and the *advantage of* $\mathcal{D}$ *for* $\mathcal{K}_{n,r,w}$

$$Adv(\mathcal{D}, \mathcal{K}_{n,r,w}) = |\Pr_{\Omega}[\mathcal{D}(H) = 1 \mid H \in \mathcal{K}_{n,r,w}] - \Pr_{\Omega}[\mathcal{D}(H) = 1]|$$

is greater than ϵ.

Decoder. A program $\phi : \mathcal{H}_{n,r} \times \mathbb{F}_2^r \longrightarrow \mathcal{S}_n(0,t)$ is a (T,ϵ)-decoder for $(\mathcal{H}_{n,r}, t)$ if it runs in time at most T and its *success probability*

$$Succ(\phi) = \Pr_{\Omega}[\phi(H, eH^{\mathrm{T}}) = e]$$

is greater than ϵ.

Adversary. A program $\mathcal{A} : \mathcal{H}_{n,r} \times \mathbb{F}_2^n \longrightarrow \mathcal{S}_n(0,t)$ is a (T,ϵ)-adversary against $\mathcal{K}_{n,r,w}$-Niederreiter if it runs in time at most T its *success probability*

$$Succ(\mathcal{A}, \mathcal{K}_{n,r,w}) = \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e \mid H \in \mathcal{K}_{n,r,w}]$$

is greater than ϵ.

An adversary against $\mathcal{K}_{n,r,w}$-McEliece could be defined as a program $\mathcal{H}_{n,r} \times \mathbb{F}_2^n \to \mathbb{F}_2^{(n-r)} \times \mathcal{S}_n(0,t)$ of probability space Ω and sample set $\mathcal{H}_{n,r} \times \mathbb{F}_2^k \times \mathcal{S}_n(0,t)$. As stated before, this setup would only make all the statements and proofs more cumbersome. Next, the proposition which supports the security reduction.

Proposition 1 ([37]). *Given the security parameters (n,r,w) and t, if there exists a (T,ϵ)-adversary against $\mathcal{K}_{n,r,w}$-Niederreiter, then there exists either a $(T,\epsilon/2)$-decoder for $(\mathcal{H}_{n,r}, t)$ or a $(T + O(n^2), \epsilon/2)$-distinguisher for $\mathcal{K}_{n,r,w}$ vs. $\mathcal{H}_{n,r}$.*

Proof. Let $\mathcal{A} : \mathcal{H}_{n,r} \times \mathbb{F}_2^r \to \mathcal{S}_n(0,t)$ be a (T,ϵ)-adversary against $\mathcal{K}_{n,r,w}$-Niederreiter. We define the following distinguisher:

$\mathcal{D}$: input $H \in \mathcal{H}_{n,r}$.
  $e \leftarrow \mathcal{S}_n(0,t)$ *//pick randomly and uniformly*
  if $(\mathcal{A}(H, eH^{\mathrm{T}}) = e)$ **then return** 1 **else return** 0.

which implies:

$$\begin{aligned}
\Pr_{\Omega}[\mathcal{D}(H) = 1] &= \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e] \\
&= \mathrm{Succ}(\mathcal{A}) \\
\Pr_{\Omega}[\mathcal{D}(H) = 1 \mid H \in \mathcal{K}_{n,r,w}] &= \Pr_{\Omega}[\mathcal{A}(H, eH^{\mathrm{T}}) = e \mid H \in \mathcal{K}_{n,r,w}] \\
&= \mathrm{Succ}(\mathcal{A}, \mathcal{K}_{n,r,w})
\end{aligned}$$

thus $Adv(\mathcal{D}, \mathcal{K}_{n,r,w}) = |Succ(\mathcal{A}, \mathcal{K}_{n,r,w}) - Succ(\mathcal{A})|$ and particularly:

$$Adv(\mathcal{D}, \mathcal{K}_{n,r,w}) + Succ(\mathcal{A}, \mathcal{K}_{n,r,w}) \geq Succ(\mathcal{A})$$

Since $Succ(\mathcal{A}, \mathcal{K}_{n,r,w}) \geq \epsilon$, we either have $Adv(\mathcal{C}, \mathcal{K}_{n,r,w})$ or $Succ(\mathcal{A})$ greater or equal to $\epsilon/2$ (recall that both are positive). The running time of $\mathcal{D}$ is equal to the running time of $\mathcal{A}$ increased by the cost for picking e and computing the product eH^{T}, which cannot exceed $O(n^2)$. So either $\mathcal{A}$ is a (T, ϵ)-decoder for $(\mathcal{H}_{n,r}, t)$ or $\mathcal{D}$ is a $(T + O(n^2), \epsilon/2)$-distinguisher for $\mathcal{K}_{n,r,w}$. □

A distinguisher for $\mathcal{K}_{n,r,w}$ *vs.* $\mathcal{H}_{n,r}$ and a decoder for $(\mathcal{H}_{n,r}, t)$ provide a solution respectively to the two following problems:

Problem 1 (Code distinguishing problem).
Parameters: $\mathcal{K}_{n,r,w}$, $\mathcal{H}_{n,r}$.
Instance: a matrix $H \in \mathcal{H}_{n,r}$.
Question: is $H \in \mathcal{K}_{n,r,w}$?

Problem 2 (Computational syndrome decoding problem).
Parameters: $\mathcal{H}_{n,r}$, an integer $t > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,r}$ and a vector $s \in \mathbb{F}_2^r$.
Problem: find a vector $e \in \mathcal{S}_n(0, t)$ such that $eH^{\mathrm{T}} = s$.

Thus it is enough to assume that none of those problems can be solved efficiently to ensure that no efficient adversary against the scheme exists.

The MDPC and the QC-MDPC cases. All the statements in this section are valid in both (MDPC and QC-MDPC) cases. We introduce an additional problem which consists in deciding the existence of words of given weight in a given linear code. Note that the code we consider below has a *generator matrix* $H \in \mathcal{H}_{n,r}$, it is thus the dual of a code in $\mathcal{F}_{n,r,w}$.

Problem 3 (Codeword existence problem).
Parameters: $\mathcal{H}_{n,r}$, an integer $w > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,r}$.
Question: is there a codeword of weight at most w in the code of generator matrix H?

Ideally, we would like to replace Problem 1 by Problem 3 in Proposition 1. Unfortunately, one would need to replace the distinguisher advantage by the quantity:

$$Adv(\mathcal{E}, \mathcal{K}_{n,r,w}) = |\Pr_{\Omega}[\mathcal{E}(H) = 1 \mid H \in \mathcal{K}_{n,r,w}] - \Pr_{\Omega}[\mathcal{E}(H) = 1]|$$

where $\mathcal{E}$ denotes a program deciding the existence of a word of weight w in a given code. However this quantity is not directly related to the hardness of Problem 3 and therefore cannot be considered. Nevertheless we reach our purpose if we assume the following assumption.

Assumption 1 *Solving Problem 1 for parameters* $(\mathcal{H}_{n,r}, \mathcal{K}_{n,r,w})$ *is not easier than solving Problem 3 for the parameters* $(\mathcal{H}_{n,r}, w)$.

Within this assumption we could modify the reduction to a claim that the $\mathcal{K}_{n,r,w}$-McEliece scheme is at least as hard as either Problem 2 and Problem 3. However we can do much better. Consider the computational problem associated to Problem 3 as follows.

Problem 4 (Codeword finding problem).
Parameters: $\mathcal{H}_{n,r}$, an integer $w > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,r}$.
Problem: find a codeword of weight at most w in the code of generator-matrix H.

This problem is polynomially equivalent to Problem 3. Furthermore, note that Problem 4 is polynomially equivalent to Problem 2.

Lemma 1. *Problem 3 is polynomially equivalent to Problem 4.*

Proof. Let $\mathcal{G}_{n,k}$ denote a subset of $\mathbb{F}_2^{k\times n}$ composed by full rank matrices. A matrix $G \in \mathcal{G}_{n,k}$ is the generator matrix of some binary linear code $\mathcal{C}$ of length n and dimension k. For any $1 \leq i \leq n$, we denote $\mathcal{C}_i$ the code shortened at i, that is

$$\mathcal{C}_i = \{c = (c_1, \ldots, c_n) \in \mathcal{C} \mid c_i = 0\}.$$

We will denote by G_i a generator matrix of $\mathcal{C}_i$. We assume we have a solution to Problem 3, that is a program $\mathcal{E} : \mathcal{G}_{n,k} \to \{0,1\}$ such that $\mathcal{E}(G) = 1$ if and only if there exists a word of weight w in the code spanned by G. The following program called on input G such that $\mathcal{E}(G) = 1$

$\mathcal{A}$: input $G \in \mathcal{G}_{n,k}$
  `for` i `from` 1 `to` n `while` G has a rank > 1
    `if` $\mathcal{E}(G_i) = 1$ `then` $G \leftarrow G_i$ // *false at most* w *times*
  `return` the first row of G of weight at most w

will return a word of weight at most w in the code spanned by G. It calls the program $\mathcal{E}$ at most n times. Conversely a solution to Problem 4 obviously provides a solution to Problem 3.

□

Lemma 2. *Problem 4 is polynomially equivalent to Problem 2.*

Proof. For a matter of simplicity, we rewrite Problem 4 (codeword finding) to receive as input the parity-check matrix of the code, instead of its generator-matrix. Obviously, both descriptions are polynomially equivalent since one matrix can be obtained from the other in polynomial time. Let $\mathcal{H}_{n,r}$ denote a subset of $\mathbb{F}_2^{r\times n}$ composed by full rank matrices. A matrix $H \in \mathcal{H}_{n,r}$ is the parity check matrix of some binary linear code $\mathcal{C}$ of length n and dimension $k = n - r$.

Problem 4 (Codeword finding problem).
Parameters: $\mathcal{H}_{n,r}$, an integer $w > 0$.
Instance: a matrix $H \in \mathcal{H}_{n,r}$.
Problem: find a codeword of weight w in the code of parity check matrix H.

1. Let us assume that we have a program $\mathcal{B}$ which solves the Problem 4 for parameters $(\mathcal{H}_{n+1,r}, w+1)$, we define the following program

   $\mathcal{A}$: input $H \in \mathcal{H}_{n,r}$, $s \in \mathbb{F}_2^r$
   $H' \leftarrow (H \mid s^T)$ // *s serves as $(n+1)$-th column of H'*
   $e \leftarrow \mathcal{B}(H')$ // $e = (e_1, \ldots, e_n, e_{n+1})$
   `if` $e_{n+1} = 1$ `then return` $(e_1, \ldots, e_n)$ `else` FAIL

   If $w+1$ is smaller than the minimum distance of the code of parity check matrix H, the call $\mathcal{A}(H)$ will never fail. This provides a solution to Problem 2 with parameters $(\mathcal{H}_{n,r}, w)$.
2. Conversely, let us assume that we have a program $\mathcal{A}$ which solves the Problem 2 for parameters $(\mathcal{H}_{n,r+1}, w)$

   $\mathcal{B}$: input $H \in \mathcal{H}_{n,r}$
   $(g_1, \ldots, g_k) \leftarrow$ a basis of $\mathcal{C}$ // *where $\mathcal{C}$ is the code of parity check matrix H*
   `for` j `from` 1 `to` n
   $H' \leftarrow$ parity check matrix of $\bigoplus_{i \neq j} \langle g_i \rangle$ // *subcode of $\mathcal{C}$ without g_j*
   `if` $\mathcal{A}(H', g_j H'^{\mathrm{T}}) \neq$ FAIL then
   $z \leftarrow \mathcal{A}(H', g_j H'^{\mathrm{T}})$
   `return` $z + g_j$
   FAIL // *$\mathcal{A}$ fails to decode for all j*

   If there exists a codeword of weight w, the decoder $\mathcal{A}$ will succeed for at least one value of j. The above program provide a solution to Problem 4 for parameters $(\mathcal{H}_{n,r}, w)$.

□

Within Assumption 1, Lemma 1 and Lemma 2, we are able to produce strong security statements.

Proposition 2. *Given Assumption 1:*

- *Breaking the MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random code.*
- *Breaking the QC-MDPC variant of McEliece or Niederreiter is not easier than solving the syndrome decoding problem for a random quasi-cyclic linear code.*

Proof. This follows directly from Proposition 1 and the polynomial equivalence of problems 3–4 (Lemma 1) and 4–2 (Lemma 2).

□

5.2 Practical security

In this section, we analyze the practical attacks against the proposed scheme. Key attacks aim either at recovering the secret decoder or simply distinguishing the public-key from a random matrix (what invalidates the security reduction). Message attacks try to decode a noisy codeword that contains a message.

Consider the system as an instantiation of the McEliece (or Niederreiter) scheme with an (n, r, w)-MDPC code, possibly quasy-cyclic, correcting t errors. We denote $\mathcal{C}$ the hidden MDPC code defined by the public-key (a generator matrix of $\mathcal{C}$ for McEliece or a parity-check matrix of $\mathcal{C}$ for Niederreiter). We claim that the best attacks for each scenario are:

- *Key distinguishing attack:* exhibit one codeword of $\mathcal{C}^\perp$ of weight w.
- *Key recovery attack:* exhibit r codewords of $\mathcal{C}^\perp$ of weight w.
- *Decoding attack:* decode t errors in a $(n, n-r)$-linear code.

For all those attacks we have to solve either the codeword finding problem or the computational syndrome decoding problem. For both problems (and for the considered parameters) the best technique currently known is information set decoding (ISD) [34]. In today's state-of-the-art the best variants derive from Stern's collision decoding algorithm [40]. There have been numerous contributions and improvements [14, 12, 11, 17, 9] until the recent asymptotic improvements [27, 6]. For selecting our parameters, we have analyzed all of them and an unpublished non-asymptotic analysis of [6] gives slightly lower workfactors (closed formulas[6] in Appendix B). ISD workfactors are commonly used to estimate the practical security of code-based schemes. However there is a novelty related to the practical security of our proposal. The problem of finding a single low weight codeword in an MDPC code may admit *many* solutions.

We denote by $\mathrm{WF}_{\mathrm{isd}}(n, r, t)$ the cost for decoding t errors (or finding a codeword of weight t) in an (n, r)-binary linear code when there is a single solution of the problem. We start by giving a basic description of the ISD algorithms. These algorithms assume a pattern for the sought error vector and it proceeds analyzing a certain set of candidates until a solution is found. This set of candidates is usually stored in lists of a certain size $\mathcal{L}$ and each candidate has a probability P to produce the solution. When the algorithm parameters are *optimal*, the workfactor $\mathrm{WF}_{\mathrm{isd}}(n, r, t)$ matches the ratio L/P, up to a small factor.

In [38], also mentioned by *Decoding One Out of Many* setting (DOOM), the gains when the decoding problem have multiple solutions and the adversary is satisfied with a single solution are analyzed. In short, when the problem has N_s solutions, the probability of success P increases by a factor N_s (as long as $N_s P \ll 1$) and when N_i instances are treated simultaneously the list size L increases at most by a factor $\sqrt{N_i}$. Therefore the DOOM technique [38] provides a gain[7] of $N_s/\sqrt{N_i}$. This gain impacts on the practical security of our MDPC

[6] This is part of an unpublished work in progress.

[7] In general, the real gain is in fact slightly smaller because these algorithms depend on optimal parameters which are not the same for multiple instances.

and QC-MDPC McEliece variants. Below we discuss these gains regarding each kind of attack against our scheme.

Key Distinguishing Attack. We assume that producing one word of weight w in the dual code $\mathcal{C}^\perp$ is enough to distinguish a public-key from a random matrix. In this scenario, an adversary applying ISD to the all-zero syndrome will face a problem with r solutions (the r rows of the sparse parity-check matrix). Then $N_s = r$ and $N_i = 1$ and the distinguishing attack cost drops by a factor of r:

$$\mathrm{WF}_{\mathrm{dist}}(n, r, w) = \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r}.$$

In the quasi-cyclic case, there is no obvious speedup and the distinguishing attack has the same cost as above.

Key Recovery Attack. To recover an equivalent private-key, it is enough to recover all (or almost all) low weight parity-check equations. All ISD variants are randomized and thus we can make r independent calls to a codeword finding algorithm. Each call costs on average $\frac{\mathrm{WF}_{\mathrm{isd}}(n,n-r,w)}{r}$ because there are r codewords of weight w. Therefore on average, recovering all equations will cost:

$$\mathrm{WF}_{\mathrm{reco}}(n, r, w) = r \cdot \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r} = \mathrm{WF}_{\mathrm{isd}}(n, n-r, w).$$

In the quasi-cyclic case, any word of low weight will provide the sparse matrix and thus the key recovery attack is no more expensive than the key distinguishing attack.

$$\mathrm{WF}^{\mathrm{QC}}_{\mathrm{reco}}(n, r, w) = \mathrm{WF}^{\mathrm{QC}}_{\mathrm{dist}}(n, r, w) = \frac{\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)}{r}.$$

Decoding Attack. In the MDPC case, the message security is related to the hardness of decoding t errors in a seemingly random binary linear code of length n and codimension r:

$$\mathrm{WF}_{\mathrm{dec}}(n, r, t) = \mathrm{WF}_{\mathrm{isd}}(n, r, t).$$

In the quasi-cyclic case, any cyclic shift of the target syndrome $s \in \mathbb{F}_2^r$ provides a new instance whose solution is equal to the one of the original syndrome, up to a block-wise cyclic shift. The number of instances and the number of solutions are thus $N_i = N_s = r$. Therefore a factor $\sqrt{r}$ is gained:

$$\mathrm{WF}^{\mathrm{QC}}_{\mathrm{dec}}(n, r, t) \geq \frac{\mathrm{WF}_{\mathrm{isd}}(n, r, t)}{\sqrt{r}}.$$

In summary, to compute the cost of each attack, we considered the non-asymptotic analysis of [6] *decreased* by the possible gains obtained by the DOOM technique described above. Note that the complex structure of the ISD variant [6] (an increased number of initial lists, pairs of non-disjoint lists and the probability of overlapped positions) might prejudice the maximal gain claimed for DOOM.

	MDPC	QC-MDPC
Key distinguishing	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$
Key recovery	$\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$	$\frac{1}{r}\mathrm{WF}_{\mathrm{isd}}(n, n-r, w)$
Decoding	$\mathrm{WF}_{\mathrm{isd}}(n, r, t)$	$\frac{1}{\sqrt{r}}\mathrm{WF}_{\mathrm{isd}}(n, r, t)$

Table 1. Best attacks for code-based encryption schemes using t-error correcting (n, r, w)-MDPC (or QC-MDPC) codes

However, since the difference of the work-factor obtained by the ISD variant [6] to the work-factor of less complex variants (which may achieve the DOOM maximal gain) is marginal, it is reasonable to use it as a secure lower bound.

Example. Let $n_0 = 2$, $n = 9602$, $r = 4801$, $w = 90$, $t = 84$. The non-asymptotic analysis of [6] gives a cost of $2^{92.70}$ for key-recovery and $2^{87.16}$ for decoding attacks. Decreasing it by the gains of the DOOM setting (a factor of 4801 and $\sqrt{4801}$), the final workfactors are $2^{80.47}$ and $2^{81.04}$.

A final remark on practical security: we choose r as a prime number to avoid attacks exploiting non-prime quasi-cyclicity [18, 26].

6 Practical Application

In Table 2, we suggest parameters for our quasi-cyclic variant, the most relevant for practical applications. For each security level, we propose three parameter sets ($n_0 = 2$, $n_0 = 3$ and $n_0 = 4$), leading to different code rates (1/2, 2/3, 3/4, respectively). The column r also gives the syndrome size in bits.

As stated before, the security assessment is based on the workfactor of the ISD variant [6] decreased by the possible gains obtained by the DOOM setting [38]. These QC-MDPC codes attain decoding failure rates below 10^{-7}, using our bit-flipping variant. Note that an MDPC code of same parameters might present a worse DFR due to the non-regularity of the column weights, but significant improvements can be obtained with slightly increased code-lengths.

The MDPC variant has a huge public-key of $r(n-r)$ bits, whilst the QC-MDPC allows for an extremely compact public-key of $(n-r)$ bits. Table 3 provides a key-size comparison of our QC-MDPC proposal, the potential[8] key-size of the QC-LDPC variant [2], the key-size of the Quasi-Dyadic Goppa McEliece variant [29] and the original McEliece scheme using updated parameters [11].

Regarding the complexity efficiency of our proposal, the key-generation step depends only on the generation of random word(s) and on (quasi-cyclic) block products. The encryption reduces to a matrix-vector product and a vector addition. For decryption, a non-optimized C++ implementation running at an Intel

[8] In [2], the use of a CCA-2 secure conversion is not considered, which would allow public-keys in systematic form. To have a fair comparison, we recompute their key-sizes assuming matrices in systematic form.

Xeon CPU @3.20GHz decrypts in less than 3 milliseconds for parameters of 80-bits of security. We prefer to omit these timings since serious optimizations may lead to much better results.

Table 2. Suggested parameters. Syndrome and key-size given in bits.

Level security	n_0	n	r	w	t	QC-MDPC key-size
80	2	9602	4801	90	84	4801
80	3	10779	3593	153	53	7186
80	4	12316	3079	220	42	9237
128	2	19714	9857	142	134	9857
128	3	22299	7433	243	85	14866
128	4	27212	6803	340	68	20409
256	2	65542	32771	274	264	32771
256	3	67593	22531	465	167	45062
256	4	81932	20483	644	137	61449

Table 3. Key-size comparison. Key-sizes given in bits.

Level security	QC-MDPC	QC-LDPC [2]	QD-Goppa [29]	Goppa [11]
80	4801	12096	20480	460 647
128	9857	–	32768	1 537 536
256	32771	–	65536	7 667 855

Note that our system can be scaled to meet arbitrarily large security requirements. It is rather straightforward to prove that the number of errors which can be corrected by the bit flipping algorithm is of order $\frac{n(1+o(1))\ln(w(1-R))}{4w}$, where n is the code-length, w the density of the parity-check matrix, R is the rate of the code. Message recovery attacks and key recovery attacks are of the same order of complexity when w is chosen of the form $(1+o(1))\sqrt{\frac{n\ln n\ln(1-R)}{\ln R}}$. Thus choosing an $(n,(1-R)n,w)$-code with w of this form allows to reach arbitrarily large security, when n goes to infinity.

7 Conclusion

MDPC codes seem to be very convenient for cryptographic purposes. Under the reasonable assumption that distinguishing a (quasi-cyclic) MDPC code from a (quasi-cyclic) random linear code amounts to being able to ascertain the existence of low weight codewords in its dual code, we show that these codes reduce the McEliece key-distinguishing problem to the problem of decoding random (quasi-cyclic) linear codes. Thus the security of our McEliece variant relies only

on a single, well studied coding-theory problem. This provides a strong argument in favor of our scheme and must be compared to the scenario for Goppa codes at the moment. Distinguishing Goppa codes is not necessarily a hard problem [15]. Although this does not necessarily lead to a practical attack, it shows that algebraic codes do not seem to be the optimal choice for cryptography.

Besides, adding a quasi-cyclic structure, our variant provides extremely compact keys: 4801 bits for 80-bits of security. Note that the state of the art indicates that a quasi-cyclic structure, by itself, does not imply a significant improvement for an adversary. All previous attacks on compact-keys McEliece variants are based on the combination of a quasi-cyclic/dyadic structure with some *algebraic* code information. Considering the way we generate our codes, this last ingredient simply does not exist. Furthermore, our variant reduces all processes (key-generation, encryption and decryption) to very low-complexity operations.

References

1. V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradzev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics—Doklady*, 11(5):1209 – 1210, 1970.
2. M. Baldi, M. Bodrato, and F. Chiaraluce. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Proceedings of the 6th international conference on Security and Cryptography for Networks*, SCN '08, pages 246–262, Berlin, Heidelberg, 2008. Springer-Verlag.
3. M. Baldi and F. Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 2591 –2595, june 2007.
4. M. Baldi, F. Chiaraluce, and R. Garello. On the usage of quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *Proceedings of the First International Conference on Communication and Electronics (ICEE'06)*, pages 305–310, October 2006.
5. M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni. Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 951 –956, june 2007.
6. 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. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 520–536. Springer, 2012.
7. T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In B. Preneel, editor, *Progress in Cryptology – Africacrypt'2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97. Springer, 2009.
8. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384 – 386, may 1978.
9. D. Bernstein, T. Lange, and C. Peters. Smaller decoding exponents: Ball-collision decoding. In P. Rogaway, editor, *Advances in Cryptology CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-22792-942.

10. D. J. Bernstein, J. Buchmann, and E. Dahmen, editors. *Post-Quantum Cryptography*. Springer-Verlag, 2009.
11. D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Proceedings of the 2nd International Workshop on Post-Quantum Cryptography*, PQCrypto '08, pages 31–46, Berlin, Heidelberg, 2008. Springer-Verlag.
12. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *Information Theory, IEEE Transactions on*, 44(1):367 –378, Jan. 1998.
13. N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology – Asiacrypt'2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174, Gold Coast, Australia, 2001. Springer.
14. I. Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.
15. J.-C. Faugère, V. Gauthier, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high rate McEliece cryptosystems. In *ITW 2011*, pages 282–286, Paraty, Brazil, Oct. 2011.
16. J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In H. Gilbert, editor, *Advances in Cryptology – Eurocrypt'2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010.
17. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology – Asiacrypt 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2009.
18. P.-A. Fouque and G. Leurent. Cryptanalysis of a hash function based on quasi-cyclic codes. In T. Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 19–35. Springer, 2008.
19. P. Gaborit. Shorter keys for code based cryptography. In *International Workshop on Coding and Cryptography – WCC'2005*, pages 81–91, Bergen, Norway, 2005. ACM Press.
20. R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, 1963.
21. J. Hagenauer, E. Offer, and L. Papke. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 42(2):429 – 445, march 1996.
22. W. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.
23. K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems -conversions for mceliece pkc -. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44586-2-2.
24. D. Kravitz. Digital signature algorithm. US patent 5231668, July 1991.
25. Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of mceliece's and niederreiter's public-key cryptosystems. *Information Theory, IEEE Transactions on*, 40(1):271 –273, jan 1994.
26. P. Loidreau. personal communication.
27. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In D. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

28. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, Jan. 1978.
29. R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, pages 376–392, 2009.
30. C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *IEEE International Symposium on Information Theory – ISIT'2000*, page 215, Sorrento, Italy, 2000. IEEE.
31. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.
32. A. Otmani, J. Tillich, and L. Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Special Issues of Mathematics in Computer Science*, 3(2):129–140, Jan. 2010.
33. S. Ouzan and Y. Be'ery. Moderate-density parity-check codes. *CoRR*, abs/0911.3262, 2009.
34. E. Prange. The use of information sets in decoding cyclic codes. *Information Theory, IRE Transactions on*, 8(5):5–9, september 1962.
35. T. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, 2008.
36. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
37. N. Sendrier. On the use of structured codes in code based cryptography. In S. Nikova, B. Preneel, and L. Storme, editors, *Coding Theory and Cryptography III*, Contactforum, pages 59–68. Koninklijke Vlaamse Academie van België voor Wetenschaeppen en Kunsten, 2009.
38. N. Sendrier. Decoding one out of many. In B.-Y. Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 51–67. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-25405-5-4.
39. P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
40. J. Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1989.

A Computing the threshold for the Bit-Flipping algorithm

A way for estimating the threshold of the bit-flipping algorithm consists in estimating the probability of a bit to be in error after a given number of algorithm iterations. When such probability converges to zero, reliable error correction can be achieved. Below we discuss a weak bound for this probability [20].

We denote by P_i the probability of a bit to be in error after i iterations of the decoding algorithm. When the code-length is supposed to be infinite and when there are no cycles of length less or equal to $2i$ in the Tanner graph associated to the parity-check matrix, this probability does not depend on a particular position [35]. These conditions can be relaxed and a finite analysis of the decoding process can be obtained, but this is beyond the scope of this work. Furthermore, practical parameters can be refined until reaching an adequate decoding failure rate.

We denote by H the parity-check matrix of an (n, r, w)-MDPC code. Suppose we are verifying the convergence of P_i, when messages containing t errors are received (thus $P_0 = \frac{t}{n}$). To describe how p_i evolves, we have to introduce some additional notation. Let m be the total number of entries equal to 1 in H. Let m_i be the total number of entries equal to 1 of H which appear in a column of weight i and let $\lambda_i \stackrel{\text{def}}{=} \frac{m_i}{m}$. Notice that m_i is also equal to i times the number of columns of weight i in H. In the quasi-cyclic case, note that $m = rw$ and $m_i = \sum_{j=0}^{n_0-1} w_j^2 \mathbf{1}_{w_j=i}$, where $\mathbf{1}_{w_j=i}$ stands for the indicator of the event $w_j = i$ (i.e. it is equal to 1 if $w_j = i$ and 0 otherwise). With this notation we have

$$p_{i+1} = p_0 - p_0 \sum_d \lambda_d \sum_{l=b_d}^{d-1} \binom{d-1}{l} \left[\frac{1+(1-2p_i)^{w-1}}{2}\right]^l \left[\frac{1-(1-2p_i)^{w-1}}{2}\right]^{d-l-1}$$
$$+(1-p_0) \sum_d \lambda_d \sum_{l=b_d}^{d-1} \binom{d-1}{l} \left[\frac{1-(1-2p_i)^{w-1}}{2}\right]^l \left[\frac{1+(1-2p_i)^{w-1}}{2}\right]^{d-l-1}$$

In [20], the integer b_d is chosen as an integer between $d-1$ and $d/2$ which aims at minimizing the function p_{i+1}.

$$\frac{1-p_0}{p_0} \leq \left[\frac{1+(1-2p_i)^{w-1}}{1-(1-2p_i)^{w-1}}\right]^{2b_d-d+1}$$

The *threshold* of an (n, r, w)-MDPC code for the original bit-flipping algorithm is obtained as the maximal integer t such that $p_0 = t/n$ and p_i converges to 0.

B Computing the work-factor of the ISD variant [6].

Consider $H \in \mathbb{F}_2^{r \times n}$, $s \in \mathbb{F}_2^r$ and $k = n - r$. We are interested in finding a vector $e \in \mathbb{F}_2^n$ of weight w such that $He^T = s$. Equivalently, we want to find a linear combination of w columns of H which when added to s gives a 0-vector. Below we briefly describe the algorithm proposed in [6] for solving this problem. The algorithm is divided in two steps: the setup and the search step. The former consists in randomly permute the columns of H and then it proceeds with a partial Gaussian elimination on the rows of H. More precisely, let l be an optimal algorithm parameter, we compute the following matrix $H' \in \mathbb{F}_2^{r \times n}$ from H:

$$H' = \left[\begin{array}{c|c} I^{(r-l)\times(r-l)} & \\ 0^{l\times(r-l)} & Q^{r\times(k+l)} \end{array}\right]$$

where I stands to an identity block and 0 to a zero block. The second step depends on the algorithm parameter $p < w$. The value of p defines the weight distribution in the sought error vector. More precisely, we will looking for vectors of weight $w-p$ in the first $r-l$ positions and of weight p in the last $k+l$ positions. A valid strategy for finding those vectors consists in computing all possible linear

combinations of p columns in Q and then select those one which sums up to a vector coinciding to the last l positions of the syndrome. We find a solution when the sum of such combination plus the syndrome gives a vector of weight exactly $w-p$. Note that the sum of each combination plus the syndrome gives a vector of weight 0 in the last l positions. Thus the weight of each combination plus the syndrome will be concentrated in the first $r-l$ positions. When this part has weight exactly $w-p$, we can add the $w-p$ columns from the identity part of H' which erase these positions. In summary, we have selected $w-p$ columns from the first $r-l$ columns of H' plus p columns from the last $k+l$ columns of H' which sum up to a vector of weight w.

An improvement in this strategy is achieved using a meet-in-the-middle strategy. It is better to compute two lists $\mathcal{L}_1$, $\mathcal{L}_2$ of all possible linear combinations of $p/2$ columns in Q, instead of computing all possible linear combinations of p columns in Q. This approach takes advantage from the birthday-paradox. Then we select the sums $\{\ a+b \mid a \in \mathcal{L}_1, b \in \mathcal{L}_2\ \}$ that have weight exactly p. Note that the fact of $\mathcal{L}_1$ and $\mathcal{L}_2$ be not disjoint might lead to multiple representations of the same solution. The attack presented in [6] uses this approach with a new advantage: they allow elements in $\mathcal{L}_1$ and $\mathcal{L}_2$ of weight $p/2+\epsilon$, for some small integer ϵ. Basically they are also considering the case when ϵ positions of a are erased by ϵ positions of b (i.e. $1+1=0$ for binary codes), which still gives a sum of weight p. The authors propose to apply this strategy not only once, but a few times, initially constructing intermediate solutions in the hope that the final solution will be the combination of these intermediate ones. This leads to an algorithm which can be divided in 4 layers, we label it from 3 (the initial) to 0 (the final layer). The third layer has 4 pairs of two disjoint lists each one. The second layer has two pairs of lists. The first layer has one pair and the layer 0 has the final list. Next we describe the cost for each step and then our estimation for the work-factor of [6].

Let p, l, p_1, p_2, ϵ_1, ϵ_2, r_1, r_2 be optimal algorithm parameters such that: $p_1 = p/2+\epsilon_1$, $p_2 = p_1/2+\epsilon_2$ and $l > r_1 > r_2$. In the initial layer, we produce 4 pairs of 2 disjoint lists each one. Each list has the linear combination of $p_2/2$ columns of Q. Thus the size of each list is: $S_3 = \binom{(k+l)/2}{p_2/2}$. We develop the discussion for a pair of lists $\mathcal{L}_{3,1}$ and $\mathcal{L}_{3,2}$, but the same apply for the other three pairs.

For the next layer, we select all sums $\{a+b \mid a \in \mathcal{L}_{3,1},\ b \in \mathcal{L}_{3,2}\}$ of weight $p_2 = p_1/2+\epsilon_2$ and which coincide with the syndrome in the last r_2 positions. Thus the size of each list is: $S_2 = \frac{(S_3)^2}{2^{r_2}}$. Let the result be $\mathcal{L}_{2,1}$ and let $\mathcal{L}_{2,2}$ be the merge from another pair in the previous layer.

For the next layer, we select all sums $\{a+b | a \in \mathcal{L}_{2,1}, b \in \mathcal{L}_{2,2}\}$ of weight $p_1 = p/2+\epsilon_1$ and which coincide with the syndrome in the last r_1 positions. Since all elements already coincide in the last r_2 positions, and $r_1 > r_2$, we have to discard only $2^{r_1-r_2}$ from all possibilities obtained from $\mathcal{L}_{2,1} \times \mathcal{L}_{2,2}$. Thus the cost of merging these lists is $C_2 = \frac{(S_2)^2}{2^{r_1-r_2}}$. Since $\mathcal{L}_{2,1}$ and $\mathcal{L}_{2,2}$ are not disjoint, we can obtain multiple representations of the same partial solution. We must proceed

with a single representation of each solution. The rate of distinct solutions is:

$$\mu_2 = \frac{\binom{k+l}{\epsilon_2}\binom{k+l-\epsilon_2}{p_2-\epsilon_2}\binom{k+l-p_2}{p_2-\epsilon_2}}{\binom{k+l}{p_2}^2}$$

The maximal size of this list is $S_1^{max} = \frac{\binom{k+l}{p_1}}{2^{r_1}}$. Thus the size of the list of distinct solutions is $S_1 = \min(\mu_2 C_2, S_1^{max})$. Let the result be $\mathcal{L}_{1,1}$ and consider $\mathcal{L}_{1,2}$ be the result from the other pair in the second layer. Finally, we select all sums $\{a+b|a \in \mathcal{L}_{1,1}, b \in \mathcal{L}_{1,2}\}$ of weight p and which coincide with the syndrome in the last l positions. Since all elements already coincide in the last r_1 positions, and $l > r_1$, we have to discard only 2^{l-r_1} from all possibilities obtained from $\mathcal{L}_{1,1} \times \mathcal{L}_{1,2}$. Thus the cost of merging these lists is $C_1 = \frac{(S_1)^2}{2^{l-r_1}}$. Again, since $\mathcal{L}_{1,1}$ and $\mathcal{L}_{1,2}$ are not disjoint, we can obtain multiple representations of the same solution, but we must consider a single representation of each solution. The rate of distinct solutions is:

$$\mu_1 = \frac{\binom{k+l}{\epsilon_1}\binom{k+l-\epsilon_1}{p_1-\epsilon_1}\binom{k+l-p_1}{p_1-\epsilon_1}}{\binom{k+l}{p_1}^2}$$

The maximal size of the final list is $S_0^{max} = \frac{\binom{k+l}{p}}{2^l}$. Thus the size of the final list of distinct solutions is $S_0 = \min(\mu_1 C_1, S_0^{max})$. Considering the cost for the Gaussian elimination as $K_0 = \frac{(n+1)(n-k)}{\log_2(n+1)}$ [1] and the cost of merging two lists being twice the cost of building a list (we use coefficients $K_1 = 1$ and $K_2 = 2$ to make this adjustment), the cost of each iteration (an attempt of the algorithm in finding a solution) is:

$$WF^{\text{iteration}}(n, r, w, p, l, r_1, r_2, \epsilon_1, \epsilon_2, p_1, p_2) = K_0 + 8S_3K_1 + 4C_3K_2 + 2C_2K_2 + C_1K_2$$

The number of iterations that the algorithm must perform until find a solution depends on the probability of finding an error vector with the sought error pattern: vectors of weight $w-p$ in the first $r-l$ positions and p in the last $k+l$ positions. This probability is

$$P(n, r, w, p, l, r_1, r_2, \epsilon_1, \epsilon_2, p_1, p_2) = \frac{\binom{n-k-l}{w-p}\binom{k+l}{p}\frac{S_0}{S_0^{max}}}{\binom{n}{w}} = \frac{\binom{n-k-l}{w-p}S_0 2^l}{\binom{n}{w}}$$

Thus we estimate the work-factor of [6], given l, p, r_1, r_2, ϵ_1, ϵ_2, p_1, p_2, as:

$$\begin{aligned} WF(n, r, w, p, l, r_1, r_2, \epsilon_1, \epsilon_2, p_1, p_2) &= P^{-1} \cdot WF^{\text{iteration}}(n, k, w, p, l, r_1, r_2, \epsilon_1, \epsilon_2, p_1, p_2) \\ &= P^{-1}(K_0 + 8S_3K_1 + 4C_3K_2 + 2C_2K_2 + C_1K_2) \end{aligned}$$

There are several ways for choosing the parameters l, p, r_1, r_2, ϵ_1, ϵ_2, p_1, p_2. With some heuristic, we succeeded to find parameters good enough to result in slightly smaller work-factors when compared to other ISD variants.