Security and complexity of the McEliece cryptosystem based on QC-LDPC codes

Marco Baldi, Marco Bianchi and Franco Chiaraluce
Department of Biomedical Engineering, Electronics and Telecommunications, Università Politecnica delle Marche, Ancona, Italy;
E-mail: \{m.baldi, m.bianchi, f.chiaraluce\}@univpm.it

Abstract

In the context of public key cryptography, the McEliece cryptosystem represents a very smart solution based on the hardness of the decoding problem, that is believed to be able to resist the future advent of quantum computers. Despite this, the original McEliece cryptosystem, based on Goppa codes, has encountered limited interest in practical applications, partly because of some constraints imposed by this special class of codes. We have recently introduced a variant of the McEliece cryptosystem adopting low-density parity-check codes, that are state-of-art codes now used in many telecommunication standards and applications. In this paper, we discuss the possible usage of a bit-flipping decoder in such context, that gives a significant advantage in terms of complexity. We also provide theoretical arguments and practical tools for estimating the trade-off between security and complexity, in such a way to give a simple procedure for the system design.

I. INTRODUCTION

In recent years, a renewed interest has been devoted to the McEliece cryptosystem [1], that is one of the most attractive options for post-quantum public key cryptography. It is based on error correcting codes to obtain both the private and public keys. In general, attacking the McEliece cryptosystem is equivalent to solve the decoding problem, that is, to obtain the error vector affecting a codeword of an \((n,k)\)-linear block code. Since such problem is also equivalent to finding a minimum weight codeword in an \((n,k + 1)\)-linear block code, the
McEliece cryptosystem can be attacked by means of algorithms aimed at finding low weight codewords.

The original version of the McEliece cryptosystem, based on binary Goppa codes with irreducible generator polynomials, is faster than the widespread RSA cryptosystem. However, it has two major drawbacks: large keys and low transmission rates, the latter being coincident with the code rates. A first improvement can be obtained, by still using Goppa codes, through the variant proposed by Niederreiter [2], that has equivalent security as the McEliece cryptosystem but exploits parity-check matrices and syndrome vectors in place of generator matrices and codewords.

However, the most effective way to overcome the drawbacks of the McEliece cryptosystem would be to replace Goppa codes with other families of codes, yielding a more compact representation of their characteristic matrices, and permitting to increase the code rate. Unfortunately, although several families of codes with such characteristics exist, only in very few cases it is possible to replace Goppa codes without incurring into serious security flaws [3], [4].

Among the most recent proposals, Quasi-Cyclic (QC) [5], Quasi-Dyadic (QD) [6] and Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes [7] have been considered for possible inclusion in the McEliece cryptosystem and also in symmetric key secure channel coding schemes [8]. However, the solutions [5] and [6] have been recently attacked [9]. Despite this, it is still possible to build variants based on QD codes that are able to resist such an attack [10]. Concerning LDPC codes, they were initially thought to be unable to give significant advantages, due to the fact that the sparse nature of their matrices cannot be exploited for reducing the key length [11]. Furthermore, adopting very large codes was found to be necessary for avoiding that the intrinsic code sparsity is exploited by an attack to the dual of the public code [12]. However, it has also been shown that, by replacing the permutation matrix used for obtaining the public key with a more general transformation matrix, the code sparsity can be hidden and the attack to the dual code avoided [13]. Unfortunately, the proposal in [13] still used only sparse transformations, that exposed it to a total break attack [14]. Subsequently, however, we have presented a simple modification that allows to avoid such flaw, so obtaining a QC-LDPC codes-based cryptosystem that is immune to any known attack [15]. Such variant of the cryptosystem is able to reduce the key size with respect to the original version and to achieve
increased transmission rate. Moreover, the size of its public keys increases linearly with the code dimension; so, it scales favorably when larger keys are needed for facing the increased computing power.

In this paper, we consolidate our previous proposal, first by introducing bit-flipping decoding for the QC-LDPC codes, that yields a significant reduction in the decoding complexity, at the cost of a moderate loss in terms of error correction performance. Performance of bit-flipping decoding can be easily predicted through theoretical arguments, and this helps dimensioning the system without the need of long numerical simulations. We also consider the most effective attack procedures known up to now and estimate analytically their work factor (WF). This way, we provide tools that permit the designer to easily find the best set of system parameters in order to optimize the trade-off between security and complexity.

The paper is organized as follows: in Section II we describe the secure version of QC-LDPC codes-based cryptosystem; in Section III we propose some encryption and decryption algorithms and evaluate their complexity; in Section IV we assess the security level of the system; finally, Section V concludes the paper.

II. MCELIECE CRYPTOSYSTEM BASED ON QC-LDPC CODES

The main functions of the McEliece cryptosystem based on QC-LDPC codes are shown in Fig. 1. QC-LDPC codes with length \( n = n_0 \cdot p \), dimension \( k = (n_0 - 1) p \) and redundancy \( r = p \) are adopted, where \( n_0 \) is a small integer (e.g., \( n_0 = 3, 4 \)), while \( p \) is a large integer (on the order of some thousands). For fixed values of the parameters, the private key is formed by
the sparse parity-check matrix $H$ of one of these codes, randomly chosen, having the following form:

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

(1)

that is, a row of $n_0$ circulant blocks $H_i$, each with row (column) weight $d_v$. Without loss of generality, we can suppose that $H_{n_0-1}$ is non singular; so, a valid generator matrix for the code in systematic form is as follows:

$$G = \begin{bmatrix}
I & \left( H_{n_0-1}^{-1} \cdot H_0 \right)^T \\
\left( H_{n_0-1}^{-1} \cdot H_1 \right)^T \\
\vdots \\
\left( H_{n_0-1}^{-1} \cdot H_{n_0-2} \right)^T
\end{bmatrix},$$

(2)

where $I$ represents the $k \times k$ identity matrix and superscript $^T$ denotes transposition.

Let us denote by $h_i$, $i = 0 \ldots n_0 - 1$, the vector containing the positions of $1$ symbols in the first row of matrix $H_i$, $i = 0 \ldots n_0 - 1$. It is easy to show that, if all the $h_i$ vectors have disjoint sets of differences modulo $p$, matrix $H$ is free of length-4 cycles in its associated Tanner graph, which is an essential prerequisite for effective LDPC decoding. Based on this fact, the secret code can be easily constructed by randomly selecting $n_0$ vectors $h_i$ with such property. This permits us to obtain large families of codes with identical parameters \[13\]. Under the LDPC decoding viewpoint, all codes in a family have the same properties; so, they show comparable error correction performance when belief propagation decoding algorithms are adopted.

In the QC-LDPC codes-based cryptosystem, Bob chooses a secret QC-LDPC code by generating its parity-check matrix, $H$, and chooses other two secret matrices: a $k \times k$ non singular scrambling matrix $S$ and an $n \times n$ non singular transformation matrix $Q$ with row/column weight $m$. Then, he obtains a systematic generator matrix $G$ for the secret code, in the form (2), and produces his public key as:

$$G' = S^{-1} \cdot G \cdot Q^{-1}.$$  

(3)

The public key is a dense matrix, but, since we adopt QC-LDPC codes, the knowledge of one row of each circulant block is sufficient to describe it. We notice that, differently from the original McEliece cryptosystem, the public code is not permutation-equivalent to the private
code. In fact, the permutation matrix used in the original system \( \mathbf{P} \) has been replaced by \( \mathbf{Q} \), that is a sparse \( n \times n \) matrix, with rows and columns weight \( m > 1 \). This way, the LDPC matrix of the secret code \( (\mathbf{H}) \) is mapped into a new parity-check matrix for the public code:

\[
\mathbf{H}' = \mathbf{H} \cdot \mathbf{Q}^T
\]

and, through a suitable choice of \( m \), the density of \( \mathbf{H}' \) can be made high enough to avoid attacks to the dual code.

Alice fetches \( \mathbf{G}' \) from the public directory, divides her message into \( k \)-bit words, and applies the encryption map as follows:

\[
x = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e},
\]

where \( x \) is the ciphertext corresponding to the cleartext \( \mathbf{u} \), and \( \mathbf{e} \) is a random vector of \( t' \) intentional errors. After receiving \( x \), Bob inverts the transformation as follows:

\[
x' = x \cdot \mathbf{Q} = \mathbf{u} \cdot \mathbf{S}^{-1} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{Q},
\]

thus obtaining a codeword of the secret LDPC code affected by the error vector \( \mathbf{e} \cdot \mathbf{Q} \), with weight \( \leq t = t' \cdot m \). Bob should be able to correct all the errors through LDPC decoding and to obtain \( \mathbf{u} \cdot \mathbf{S}^{-1} \). Finally, he can recover \( \mathbf{u} \) through multiplication by \( \mathbf{S} \).

From (6), we note that the introduction of matrix \( \mathbf{Q} \) causes an error propagation effect (at most by a factor \( m \)) within each received frame. This is compensated by the high error correction capability of the QC-LDPC code, that must be able to correct up to \( t \) errors. Suitable QC-LDPC codes can be designed for such purpose.

III. ENCRYPTION, DECRYPTION AND THEIR COMPLEXITY

A. Key size and transmission rate

In the QC-LDPC codes-based cryptosystem, due to the special form \( \mathbf{H} \) of matrix \( \mathbf{H} \), the code rate is \( (n_0 - 1)/n_0 \). In the following, we will focus on two values of \( n_0 \), namely: \( n_0 = 3, 4 \), that give transmission rates equal to 2/3 and 3/4, respectively.

The public key is a binary matrix formed by \( k_0 \times n_0 \) circulant blocks, each with size \( p \times p \). Since each circulant block is completely described by a single row (or column), that is, \( p \) bits, the public key size is \( k_0 \cdot n_0 \cdot p = (n_0 - 1) \cdot n_0 \cdot p \) bits.
### TABLE I
PUBLIC KEY SIZE EXPRESSED IN BYTES.

<table>
<thead>
<tr>
<th>$p$ [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0 = 3$</td>
<td>3072</td>
<td>3840</td>
<td>4608</td>
<td>5376</td>
<td>6144</td>
<td>6912</td>
<td>7680</td>
<td>8448</td>
<td>9216</td>
<td>9984</td>
<td>10752</td>
<td>11520</td>
<td>12288</td>
</tr>
<tr>
<td>$n_0 = 4$</td>
<td>6144</td>
<td>7680</td>
<td>9216</td>
<td>10752</td>
<td>12288</td>
<td>13824</td>
<td>15360</td>
<td>16896</td>
<td>18432</td>
<td>19968</td>
<td>21504</td>
<td>23040</td>
<td>24576</td>
</tr>
</tbody>
</table>

The values of key size (expressed in bytes) are reported in Table I for $n_0 = 3, 4$ and for a set of values of $p$ that we will consider throughout the paper. As we observe from the table, all choices of the system parameters we have considered give smaller key size and higher transmission rate than those of the original McEliece cryptosystem (that has key size 67072 bytes and rate 0.51) [1] and its Niederreiter version (that has key size 32750 bytes and rate 0.57) [2].

### B. Multiplication by circulant matrices

A fundamental point for reducing complexity in the considered cryptosystem is to adopt efficient algorithms for performing multiplication of a circulant matrix by a vector.

Since circulant matrices are also Toeplitz matrices, an effective algorithm for fast computation of vector-matrix products is the Winograd convolution [16]. The Winograd algorithm is a generalization of the Karatsuba-Ofman algorithm, that has been reviewed even recently, in the perspective to allow fast VLSI implementations [17]. If we consider a $p \times p$ Toeplitz matrix $T$, with even $p$, we can decompose it as follows:

$$
\begin{bmatrix}
T_0 & T_1 \\
T_2 & T_0
\end{bmatrix}
= \begin{bmatrix}
I & 0 & I \\
0 & I & I
\end{bmatrix}
\begin{bmatrix}
T_1 - T_0 & 0 & 0 \\
0 & T_2 - T_0 & 0 \\
0 & 0 & T_0
\end{bmatrix}
\begin{bmatrix}
0 & I \\
I & 0
\end{bmatrix},
$$

(7)

where $I$ and $0$ are the $p/2 \times p/2$ identity and null matrix, respectively, and $T_0, T_1, T_2$ are $p/2 \times p/2$ Toeplitz matrices, as well as $T_1 - T_0$ and $T_2 - T_0$. It follows that the multiplication of a vector $V = [V_0 \ V_1]$ by matrix $T$ can be split into three phases:

- **Evaluation phase**: multiplication of $V$ by the first matrix translates into the addition of two $p/2$-bit vectors ($V_0$ and $V_1$); so, its cost, in terms of binary operations, is $p/2$.
- **Multiplication phase**: the vector resulting from the evaluation phase must be multiplied by the second matrix. This translates into 3 vector-matrix products by $p/2 \times p/2$ Toeplitz
matrices. If \( p/2 \) is even, the three multiplications can be computed in a recursive way, by splitting each of them into four \( p/4 \times p/4 \) blocks. If \( p/2 \) is odd (or sufficiently small to make splitting no more advantageous), vector-matrix multiplication can be performed in the traditional way and its complexity is about \((p/2)^2/2\).

- **Interpolation phase:** the result of the multiplication phase must be multiplied by the third matrix. This requires 2 additions of \( p/2 \)-bit vectors, that is, further \( p \) binary operations.

The matrix \( G' \) used in the QC-LDPC codes-based cryptosystem is formed by \( k_0 \times n_0 \) circulant blocks with size \( p \times p \). When a vector is multiplied by such matrix, we can split the vector into \( k_0 \)-bit subvectors and consider \( k_0 \cdot n_0 \) vector-matrix multiplications. However, we must take into account that the evaluation phase on the \( k_0 \)-bit subvectors must be performed only once, and that further \((k_0 - 1) \cdot n_0 \cdot p\) binary operations are needed for re-combining the result of multiplication by each column of circulants.

### C. Encryption operations and complexity

Encryption is performed by calculating the product \( u \cdot G' \) and then adding the intentional error vector \( e \). So, the encryption complexity can be estimated by considering the cost of a vector-matrix multiplication through the Winograd convolution and adding \( n \) binary operations for summing the intentional error vector.

<table>
<thead>
<tr>
<th>( p ) [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_0 = 3 )</td>
<td>726</td>
<td>823</td>
<td>919</td>
<td>1005</td>
<td>1092</td>
<td>1178</td>
<td>1236</td>
<td>1351</td>
<td>1380</td>
<td>1524</td>
<td>1510</td>
<td>1697</td>
<td>1639</td>
</tr>
<tr>
<td>( n_0 = 4 )</td>
<td>956</td>
<td>1081</td>
<td>1206</td>
<td>1321</td>
<td>1437</td>
<td>1552</td>
<td>1624</td>
<td>1783</td>
<td>1811</td>
<td>2013</td>
<td>1984</td>
<td>2244</td>
<td>2157</td>
</tr>
</tbody>
</table>

Table II reports the values of encryption complexity, expressed in terms of the number of binary operations needed for each encrypted bit, as a function of the circulant matrix size \( p \), for \( n_0 = 3 \) and \( n_0 = 4 \). The usage of the Winograd convolution is particularly efficient when \( p \) is a power of 2, since, in such cases, recursion can be exploited to the utmost.

### D. Decryption operations and complexity

Bob must perform the following three operations for decrypting the received message:
1) calculate the product $x \cdot Q$;
2) decode the secret LDPC code;
3) calculate the product $u' \cdot S$.

Matrices $Q$ and $S$ are formed, respectively, by $n_0 \times n_0$ and $k_0 \times k_0$ circulant blocks. However, while matrix $S$ is dense, matrix $Q$ is sparse (with row/column weight $m \ll n$). So, it is advantageous to use naïve multiplication (requiring $n \cdot m$ binary operations) for calculating the product $x \cdot Q$. On the contrary, the complexity of operation 3) can be reduced by resorting to the Winograd convolution for efficient multiplication of a vector by a circulant matrix. Concerning phase 2), Bob must exploit the secret LDPC matrix to implement a suitable decoding algorithm for trying to correct all intentional errors (that are $\leq t = t'm$). LDPC decoding is usually accomplished through iterative decoding algorithms on the code Tanner graph, that implement the belief propagation principle to provide very good error correction capability. Among them: the sum-product algorithm (SPA) [18] and the bit-flipping (BF) algorithm [19]. The SPA exploits real valued messages and ensures the best performance on channels with soft information. When soft information from the channel is not available, as it occurs in the present case, it may be advantageous to adopt the BF algorithm, that works on binary messages and requires very low complexity, though its performance is not as good as that of the SPA.

The principle of the BF algorithm was devised in Gallager’s seminal work for LDPC codes with a tree representation [19]. Given an LDPC parity-check matrix with column weight $d_v$, the variable nodes of its Tanner graph are initially filled with the received codeword bits. During an iteration, every check node $c_i$ sends each neighboring variable node $v_j$ the binary sum of all its neighboring variable nodes other than $v_j$. So, each variable node receives $d_v$ parity-check sums. In order to send back a message to each neighboring check node $c_j$, node $v_j$ counts the number of unsatisfied parity-check sums from check nodes other than $c_j$. If such number is $\geq b \leq d_v - 1$, then $v_j$ flips its value and sends it to $c_i$, otherwise $v_j$ sends its initial value unchanged to $c_i$.

At the next iteration, the check sums are updated with such new values, until all of them are satisfied or a maximum number of iterations is reached. Two algorithms, named A and B, were originally proposed by Gallager [19]: in algorithm A the value $b = d_v - 1$ is fixed, while in algorithm B it can vary between $\lceil d_v/2 \rceil$ and $d_v - 1$ during decoding (we denote by
Algorithm A is simpler to implement, but algorithm B ensures better performance.

An important issue of LDPC codes is that, differently from algebraic hard-decision codes, their decoding radius is generally not known. So, numerical simulations are usually exploited for estimating their performance, but such approach is time demanding and unpractical for the purpose of dimensioning the QC-LDPC codes-based cryptosystem. In the following, we show how we can estimate the performance of the BF algorithm, when applied in the considered scenario, through theoretical arguments very similar to those developed in [20].

Let us suppose that Bob, after having received the ciphertext, performs decoding through algorithm A. At each iteration of the algorithm, we denote by $p_{cc}$ the probability that a bit is not in error and a generic parity-check equation evaluates it correctly. Instead, $p_{ci}$ is the probability that a bit is not in error and a parity-check equation evaluates it incorrectly. Similarly, $p_{ic}$ and $p_{ii}$ are the probabilities that a bit is in error and a parity-check equation evaluates it correctly and incorrectly, respectively. In the considered context, it is easy to verify that the following expressions hold:

$$
\begin{align*}
    p_{cc} (q_l) &= \sum_{j=0}^{\min\{d_c-1,q_l\}} \binom{d_c-1}{j} \frac{(n-d_c)}{\binom{n-1}{j}} \\
    p_{ci} (q_l) &= \sum_{j=0}^{\min\{d_c-1,q_l\}} \binom{d_c-1}{j} \frac{n-d_c}{\binom{n-1}{j}} \\
    p_{ic} (q_l) &= \sum_{j=0}^{\min\{d_c-1,q_l\}} \binom{d_c-1}{j} \frac{n-d_c}{\binom{n-1}{j}} \\
    p_{ii} (q_l) &= \sum_{j=0}^{\min\{d_c-1,q_l\}} \binom{d_c-1}{j} \frac{n-d_c}{\binom{n-1}{j}} 
\end{align*}
$$

(8)

where $d_c = n_0 \cdot d_v$ is the row weight of matrix $H$ and $q_l$ is the average number of residual errors after iteration $l$. In the considered context, we have $q_0 \leq t = t'm$; we fix $q_0 = t = t'm$ in order to obtain worst-case estimates (maximum error propagation).

Let us suppose that, after iteration $l$, the estimate of a bit is in error. Based on (8), we can calculate the probability that, during the subsequent iteration, the message originating from its corresponding variable node is correct:

$$
    f^b (q_l) = \sum_{j=b}^{d_v-1} \binom{d_v-1}{j} \left[ p_{ic} (q_l) \right]^j \left[ p_{ii} (q_l) \right]^{d_v-1-j}.
$$

(9)
Similarly, the probability of incorrectly evaluating, in a single iteration of the algorithm, a bit that is not in error can be expressed as:

$$g^b(q_l) = \sum_{j=b}^{d_v-1} \left(\frac{d_v - 1}{j}\right) [p^{ci}(q_l)]^j [p^{ce}(q_l)]^{d_v-1-j}. \quad (10)$$

Under the ideal assumption of a cycle-free Tanner graph (that implies to consider an infinite-length code), the average number of residual bit errors at the \( l \)-th iteration, \( q_l \), can be expressed as:

$$q_l = t - t \cdot f^b(q_{l-1}) + (n - t) \cdot g^b(q_{l-1}). \quad (11)$$

Based on this recursive procedure, we can calculate a waterfall threshold by finding the maximum value \( t = t_{th} \) such that \( \lim_{l \to \infty} (q_l) = 0 \).

Actually, different values of \( t_{th} \) can be found by different choices of \( b \). So, rather than resorting only to algorithm A (in which \( b = d_v - 1 \) is fixed), we can also optimize the choice of \( b \) by looking for the minimum \( t_{th} \) for each \( b \in \{\lceil d_v/2 \rceil, \ldots, d_v - 1\} \). This way, variants of algorithm A with better choices of \( b \) can be obtained. For each set of code parameters, we will refer to the optimal choice of \( b \) in the following.

### TABLE III

<table>
<thead>
<tr>
<th>( p ) [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_0 = 3 )</td>
<td>( d_v = 13 )</td>
<td>223</td>
<td>278</td>
<td>334</td>
<td>389</td>
<td>445</td>
<td>500</td>
<td>556</td>
<td>611</td>
<td>667</td>
<td>722</td>
<td>777</td>
<td>833</td>
</tr>
<tr>
<td>( d_v = 15 )</td>
<td>190</td>
<td>240</td>
<td>290</td>
<td>339</td>
<td>389</td>
<td>438</td>
<td>488</td>
<td>538</td>
<td>587</td>
<td>637</td>
<td>686</td>
<td>736</td>
<td>785</td>
</tr>
<tr>
<td>( n_0 = 4 )</td>
<td>( d_v = 13 )</td>
<td>177</td>
<td>222</td>
<td>266</td>
<td>311</td>
<td>355</td>
<td>400</td>
<td>445</td>
<td>489</td>
<td>534</td>
<td>578</td>
<td>623</td>
<td>667</td>
</tr>
<tr>
<td>( d_v = 15 )</td>
<td>153</td>
<td>192</td>
<td>230</td>
<td>269</td>
<td>307</td>
<td>346</td>
<td>384</td>
<td>423</td>
<td>461</td>
<td>500</td>
<td>538</td>
<td>577</td>
<td>615</td>
</tr>
</tbody>
</table>

Table III reports the threshold values, so obtained, for several values of the circulant block size \( p \), code rates 2/3 \( (n_0 = 3) \) and 3/4 \( (n_0 = 4) \), and two values of column weight: \( d_v = 13, 15 \).

In more realistic scenarios, with finite code lengths and closed loops in the Tanner graphs, also adopting a finite number of decoding iterations, there is no guarantee that the error rate is arbitrarily small for \( t \leq t_{th} \). In this sense, the values in Table III should be seen as an optimistic assumption. However, we can observe that the performance achievable by BF with fixed \( b \) can be improved in a number of ways.
One of these improvements has been mentioned above, and consists in using Algorithm B (i.e., variable $b$). On the other hand, more recently, the original Gallager’s algorithms have been made more efficient through further, and more elaborated, variants [21], [22]. Such improved versions reduce the gap in performance with respect to the SPA, that, as we have verified through numerical simulations, is able to reach extremely small error rates for values of $t$ even above the BF threshold $t_{th}$ [7]. So, taking into account these aspects, we can consider the BF threshold values as reliable approximations of the decoding radius of the considered QC-LDPC codes.

As concerns complexity, we can estimate the number of binary operations needed for each iteration of the algorithm over the code Tanner graph. During an iteration, each check node receives $d_c$ binary values and EX-ORs them, for a total of $d_c - 1$ binary sums. The result is then EX-ORed again with the message coming from each variable node before sending it back to the same node, thus requiring further $d_c$ binary sums. So, the total number of operations at check nodes is $r(2d_c - 1)$. Similarly, each variable node receives $d_v$ check sum values and counts the number of them that are unsatisfied; this requires $d_v$ operations. After that, for each neighboring check node, any variable node updates the number of unsatisfied check sums by excluding the message received from that node and compares the result with the threshold $b$; this requires further $2d_v$ operations. So, the total number of operations at variable nodes is $n(3d_v)$. In conclusion, the cost of one iteration of bit flipping can be estimated as

$$C^{(1)}_{BF} = r(2d_c - 1) + n(3d_v) = 5nd_v - r$$  \hspace{2cm} (12)

Based on (12) and considering the computational effort required for calculating the $x \cdot Q$ and $u' \cdot S$ products, we can estimate the total cost, in terms of binary operations, for each decrypted bit. The values obtained are reported in Table IV where $m = 7$ has been assumed and a BF algorithm with 10 average iterations has been considered.

By using the same parameters, and considering $v = 6$ quantization bits for the decoder messages, we have estimated the decryption complexity with SPA decoding [7]; the results are reported in Table V. Performing decoding through the SPA guarantees the best error correction performance at the threshold value $t = t_{th}$. However, in comparison with Table IV the adoption of BF decoding gives a significant advantage over the SPA in terms of decryption complexity.
### TABLE IV
Binary operations needed for each decrypted bit by using BF decoding.

<table>
<thead>
<tr>
<th>$p$ [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0 = 3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>1476</td>
<td>1544</td>
<td>1611</td>
<td>1668</td>
<td>1726</td>
<td>1784</td>
<td>1827</td>
<td>1899</td>
<td>1928</td>
<td>2014</td>
<td>2101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>1626</td>
<td>1694</td>
<td>1761</td>
<td>1818</td>
<td>1876</td>
<td>1934</td>
<td>1977</td>
<td>2049</td>
<td>2078</td>
<td>2164</td>
<td>2164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_0 = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>1598</td>
<td>1694</td>
<td>1790</td>
<td>1877</td>
<td>1963</td>
<td>2050</td>
<td>2107</td>
<td>2223</td>
<td>2252</td>
<td>2396</td>
<td>2381</td>
<td>2569</td>
<td>2511</td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>1731</td>
<td>1828</td>
<td>1924</td>
<td>2010</td>
<td>2097</td>
<td>2183</td>
<td>2241</td>
<td>2356</td>
<td>2385</td>
<td>2529</td>
<td>2515</td>
<td>2702</td>
<td>2644</td>
</tr>
</tbody>
</table>

### TABLE V
Binary operations needed for each decrypted bit by using SPA decoding.

<table>
<thead>
<tr>
<th>$p$ [bits]</th>
<th>4096</th>
<th>5120</th>
<th>6144</th>
<th>7168</th>
<th>8192</th>
<th>9216</th>
<th>10240</th>
<th>11264</th>
<th>12288</th>
<th>13312</th>
<th>14336</th>
<th>15360</th>
<th>16384</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0 = 3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>9791</td>
<td>9859</td>
<td>9926</td>
<td>9983</td>
<td>10041</td>
<td>10099</td>
<td>10142</td>
<td>10214</td>
<td>10243</td>
<td>10329</td>
<td>10329</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>11261</td>
<td>11329</td>
<td>11396</td>
<td>11453</td>
<td>11511</td>
<td>11569</td>
<td>11612</td>
<td>11684</td>
<td>11712</td>
<td>11799</td>
<td>11799</td>
<td>11915</td>
<td>11886</td>
</tr>
<tr>
<td>$n_0 = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_v = 13$</td>
<td>9068</td>
<td>9164</td>
<td>9260</td>
<td>9347</td>
<td>9433</td>
<td>9520</td>
<td>9577</td>
<td>9693</td>
<td>9722</td>
<td>9866</td>
<td>9851</td>
<td>10039</td>
<td>9981</td>
</tr>
<tr>
<td>$d_v = 15$</td>
<td>10375</td>
<td>10471</td>
<td>10567</td>
<td>10653</td>
<td>10740</td>
<td>10826</td>
<td>10884</td>
<td>10999</td>
<td>11028</td>
<td>11172</td>
<td>11158</td>
<td>11345</td>
<td>11288</td>
</tr>
</tbody>
</table>

### IV. Security level

Attacks can be divided into two classes:

- attacks aimed at recovering the secret code;
- attacks aimed at decrypting the transmitted ciphertext.

As we have shown in [15], [7], the proper usage of matrices $S$ and $Q$ to disguise the secret code in the public matrix is able to prevent attacks exploiting its sparsity (even within its dual).

More precisely, the most dangerous attacks of the first type (like the attack to the dual code and OTD attacks [14]) can be prevented by choosing a dense $S$ matrix and a sparse $Q$ matrix with, for example, row and column weight $m = 7$.

On the contrary, due to the low weight ($t'$) of the intentional error vector, decoding attacks of the second type are more dangerous and, in many cases, provide the smallest WF.

#### A. Decoding attacks

Decoding attacks aim at solving the decoding problem, that is, obtaining the error vector $e$ used for encrypting a ciphertext. A way for finding $e$ is to search for the minimum weight
codewords of an extended code, generated by:

\[ G'' = \begin{bmatrix} G' \\ x \end{bmatrix}. \tag{13} \]

The WF of such attacks can be determined by referring to the Stern’s algorithm [23]. This algorithm has been further improved, reaching a speedup of \(2^3\) or \(2^4\) [24]. However, estimating the complexity of its modified versions requires resorting to Markov chains and no closed form formulas are available. So, we refer our estimates to Stern’s original formulation, and then take into account the possible speedup deriving from the most recent improvements.

In the QC-LDPC codes-based cryptosystem, a further speedup is obtained by considering that, due to the quasi-cyclic property of the codes, each block-wise cyclically shifted version of the ciphertext \(x\) is still a valid ciphertext. So, the eavesdropper can continue extending \(G''\) by adding block-wise shifted versions of \(x\), and can search for one among as many shifted versions of the error vector. So, in order to estimate the minimum WF, we must consider the optimum number of shifted ciphertexts that can be used by an attacker in the generator matrix of the extended code.

**TABLE VI**

| SECURITY LEVEL OF THE QC-LDPC CODES-BASED CRYPTOSYSTEM FOR \(m = 7\). |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \(p\) [bits] | 4096 | 5120 | 6144 | 7168 | 8192 | 9216 | 10240 | 11264 | 12288 | 13312 | 14336 | 15360 | 16384 |
| \(n_0 = 3\) | \(d_v = 13\) | \(2^{56}\) | \(2^{65}\) | \(2^{75}\) | \(2^{85}\) | \(2^{96}\) | \(2^{106}\) | \(2^{117}\) | \(2^{126}\) | \(2^{137}\) | \(2^{148}\) | \(2^{159}\) | \(2^{160}\) |
| \(d_v = 15\) | \(2^{56}\) | \(2^{66}\) | \(2^{76}\) | \(2^{86}\) | \(2^{96}\) | \(2^{106}\) | \(2^{117}\) | \(2^{128}\) | \(2^{139}\) | \(2^{148}\) | \(2^{159}\) | \(2^{169}\) | \(2^{180}\) |
| \(n_0 = 4\) | \(d_v = 13\) | \(2^{62}\) | \(2^{75}\) | \(2^{86}\) | \(2^{98}\) | \(2^{109}\) | \(2^{111}\) | \(2^{123}\) | \(2^{136}\) | \(2^{148}\) | \(2^{153}\) | \(2^{153}\) | \(2^{160}\) |
| \(d_v = 15\) | \(2^{64}\) | \(2^{77}\) | \(2^{90}\) | \(2^{101}\) | \(2^{115}\) | \(2^{128}\) | \(2^{140}\) | \(2^{153}\) | \(2^{167}\) | \(2^{176}\) | \(2^{176}\) | \(2^{176}\) | \(2^{176}\) |

For each QC-LDPC code, we have calculated the maximum number of intentional errors \(t' = \lfloor t/m \rfloor\) by considering \(m = 7\) and the estimated error correction capability \(t\) reported in Table III. The minimum values of attack WF, obtained in such conditions, are shown in Table VI.

For \(n_0 = 3\) the WF of the attack to the dual code, also based on Stern’s algorithm, is about \(2^{160}\) when \(d_v = 13\) and \(2^{184}\) when \(d_v = 15\). So, we have reported the former of such values in Table VI for those cases in which the decoding attack WF would be higher. The same has been done for \(n_0 = 4\), for which the WF of the attack to the dual code is about \(2^{153}\) and \(2^{176}\)
for $d_v = 13$ and $d_v = 15$, respectively.

In order to give an example of system design, we can consider the parameters of the Goppa code suggested in [24] for achieving 80-bit security (i.e., $WF = 2^{80}$), that are: $n = 1632$, $k = 1269$ and $t = 33$. They give a key size of 258876 bytes for the McEliece cryptosystem and 57581 bytes for the Niederreiter version. The encryption and decryption complexity, estimated through the formulas in [25], p. 27, result in 817 and 2472 operations per bit, respectively, for the McEliece cryptosystem and 48 and 7890 operations per bit for the Niederreiter version. The transmission rate is 0.78 for the McEliece cryptosystem and 0.63 for the Niederreiter version.

A similar security level can be reached by the QC-LDPC codes-based cryptosystem with $n_0 = 4$, $p = 6144$ and $d_v = 13$, even considering the possible speedup deriving from the adoption of improved versions of Stern’s algorithm. In such case, as reported in Table I, the public key size is 9216 bytes, i.e., 28 times smaller than in the McEliece cryptosystem and 6 times smaller than in the Niederreiter version. The transmission rate is 0.75, similar to that of the McEliece cryptosystem and higher than in the Niederreiter version. The encryption and decryption complexity, as reported in Tables II and IV, are determined by 1206 and 1790 operations per bit, respectively. So, complexity increases in the encryption stage, but, by exploiting the BF algorithm, the decryption complexity is reduced.

So, we can conclude that, for achieving the same security level, the QC-LDPC codes-based cryptosystem can adopt smaller keys and comparable or higher transmission rates with respect to the McEliece and Niederreiter cryptosystems. Moreover, this does not come at the expense of a significantly increased complexity.

V. CONCLUSION

We have deepened the analysis of a variant of the McEliece cryptosystem, that adopts QC-LDPC codes in place of Goppa codes. Such modification is aimed at overcoming the main drawbacks of the original system, while still allowing to reach a satisfactory security level.

We have proposed to adopt bit flipping algorithms for decoding the QC-LDPC codes, in such a way as to achieve a rather good performance while strongly reducing the decoding complexity with respect to the SPA. The adoption of bit flipping decoding has also allowed to develop simple analytical tools for estimating the error correction capability of the considered codes, thus simplifying the system design without requiring long numerical simulations. Together with
the methods we have described to evaluate complexity, these tools provide the system designer a fast procedure for optimizing the choice of the cryptosystem parameters.

REFERENCES


