A new cryptosystem based on the McEliece

Pedro de Melo Branco
Instituto Superior Técnico, Lisboa, Portugal

October 2017

Abstract

An efficient McEliece-based cryptosystem is proposed to handle large messages and that can be easily implemented in hardware. The main idea is to incorporate LDPC codes after several parallel runs of the original McEliece cryptosystem, achieving a low circuit-depth complexity while profiting from the capability of LDPC codes to deal with large blocks of messages. The proposed cryptosystem is, at least, as hard as the original McEliece cryptosystem, and therefore it is believed to be robust to quantum attacks. Moreover, attacks to McEliece cryptosystems based on LDPC codes do not hold for our proposal. The key size of the cryptosystem is shown to be roughly ten times smaller than the original McEliece to achieve similar levels of security. Finally, an IND-CCA2 variant for the new cryptosystem is presented.
Keywords: cryptography, coding theory, public-key encryption, McEliece cryptosystem, LDPC codes

1. Introduction

One of the most important features an asymmetric cryptosystem must fullfil is to be hardware implementable. Both RSA and elliptic curve cryptosystems (ECC) are highly demanding from the processing time point of view, making them very slow to use, even in dedicated hardware implementations [**?**]. Moreover, both are not quantum resistant [**?**], motivating the community to look for post-quantum cryptosystems [**?**]. One of the most promising candidates is the McEliece cryptosystem [**?**] that is based on error-correcting codes, which are usually well behaved in hardware implementations. This cryptosystem is the oldest one that is yet to be broken, by both classical and quantum algorithms.

Unfortunately, although much more efficient than RSA and ECC [**?**], the McEliece cryptosystem has still some downsides: the key size is much larger than both RSA and ECC, which is a problem for implementations (especially hardware implementations, where the memory available to store data is very limited); and the decryption algorithm adquires a bottleneck effect as the key size grows since the decoding algorithm of Goppa codes (the codes used in the original McEliece) takes to long as the key size increases [**?**]. On the other hand, we know that LDPC codes [**?**] scale very well in harware implementations since their decoding algorithms are much simpler. But using LDPC codes in the McEliece may not be as secure as using Goppa codes [**?**].

The McEliece cryptosystem has yet another downside: it is not indistinguishable against adaptive chosen-ciphertext attacks (IND-CCA2). IND-CCA2 security is a crucial property for cryptosystems nowadays as this notion reveals how much information an adversary can get on the message from the ciphertext.

In this paper, we propose a cryptosystem to solve these problems. The cryptosystem, given the message encrypted, divides it into several parts and encrypts each part individually using a Goppa code and McEliece cryptosystem (such that it can be implemented in parallel) and then encrypts the result using an LDPC code. With this proposal, we aim to achieve the security provided by traditional McEliece and the scalability provided by LDPC codes.

We proved that our cryptosystem is, at least, as hard as the original McEliece and argue that the known attacks do not work on it. Thus, our proposal is suitable for a post-quantum world. We were able to reduce roughly ten times the key size comparing to the original McEliece by noting that, to attack our proposal, an adversary has to attack several times several McEliece based cryptosystems. We also analyze the circuit-depth complexity of our cryptosystem and we conclude that, as long as we use small Goppa codes, it is much faster than the original McEliece and it is extremely parallelizable making it suitable for hardware implementations.

Finally, we propose an efficient way to convert this new protocol into an IND-CCA2 secure cryptosystem. This IND-CCA2 variant is inspired on the Fujisaki-Okamoto generic conversion [**?**] and its security is based on the hardness of breaking the IND-CPA security of the randomized McEliece [**?**].

The paper is organized as follows: we will begin by presenting some background on coding theory and the McEliece cryptosystem, as well as some security definitions and results. Next, in section 3, we present the cryptosystem and its efficiency analysis. In section 4 we present the security reduction and we analyze the

security parameters and key size. Finally we present an IND-CCA2 variant and its proof of security.

2. Preliminaries

We begin this chapter with some notation. Throughout this text, if v and u are two vectors over $\mathbb{Z}_2$, $v+u$ denotes their XOR, $v|u$ denotes their concatenation, $v.u$ denotes their inner product and $wt(v)$ denotes the Hamming weight of the vector v (the number of coordinates that are non-null). $\mathbb{F}_{2^p}$ is a Galois field of order 2^p and $\mathbb{Z}_2$ denotes the field of order 2. A vector $v \in \mathbb{F}^n$, where $\mathbb{F}$ is a field, is a vector with n coordinates and where each coordinate is an element of $\mathbb{F}$. For a polynomial $f(x) = f_0 + f_1 x + \dots f_n x^n$ over some field $\mathbb{F}$, the degree $\deg f$ is equal to n. If A is a matrix, A^T denotes its transpose.

Throughout this text, we will denote plaintexts by $\mathbf{m}$, and ciphertexts by $\mathbf{c}$. If $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a public-key cryptosystem, then $\mathcal{K}$ is its key creation algorithm, $\mathcal{E}$ is its encryption algorithm and $\mathcal{D}$ is its decryption algorithm. As usual, algorithm $\mathcal{K}$ receives as input the security parameter k written in unary 1^k and outputs a random pair of public and secret keys (pk, sk) . Moreover, algorithm $\mathcal{E}$ receives a public key and a message $(pk, \mathbf{m})$ and outputs the encryption of $\mathbf{m}$, that is $\mathcal{E}(pk, \mathbf{m}) = \mathbf{c}$. Similarly, $\mathcal{D}$ receives a private and a ciphertext $(sk, \mathbf{c})$ and outputs the decryption of such ciphertext $\mathcal{D}(sk, \mathbf{c}) = \mathbf{m}$.

2.1. Goppa codes

For $p \in \mathbb{N}$, we define a binary Goppa $\mathcal{G}$ by a polynomial $g \in \mathbb{F}_{2^p}[x]$ and a sequence of elements $L_1, \dots, L_n$ of $\mathbb{F}_{2^p}$ in , called L. The polynomial g should not have multiple zeros. The sequence elements must be different from each other and should not be a zero of g, i.e, $f(L_i) \neq 0$, for all $i = 1, \dots, n$. The code $\mathcal{G}$ is defined as follows:

$$\mathcal{G} := \mathcal{G}(f, L) = \left\{ c \in \mathbb{Z}_2^n : \sum_{i=0}^{n-1} \frac{c_i}{x - L_i} = 0 \mod g(x) \right\}$$

This code has dimension $k = n - mt$. Its parity-check matrix H can be computed in the following way [**?**]:

$$H = \begin{bmatrix} \frac{1}{g(L_0)} & \cdots & \frac{1}{g(L_{n-1})} \\ \frac{L_0}{g(L_0)} & \cdots & \frac{L_{n-1}}{g(L_{n-1})} \\ \vdots & & \vdots \\ \frac{L_0^{t-1}}{g(L_0)} & \cdots & \frac{L_{n-1}^{t-1}}{g(L_{n-1})} \end{bmatrix}$$

where $L_i \in L$, for every $i \in 1, \dots, n$.

To decode Goppa codes one can use Patterson's algorithm [**?**]. If the code is defined by a polynomial $g(x)$ in $\mathbb{F}_{2^p}[x]$ and with $\deg g(x) = t$, then this algorithm runs in time $\mathcal{O}(ntp^2)$ [**?**]. As n grows, p and t also grow. So, for large block messages, Goppa codes become inefficient for practical implementations [**?**].

2.2. LDPC codes

Low-density parity-check (LDPC) codes form a class of linear codes that are obtained from sparse bipartite graphs. Let X be a bipartite graph; the left nodes of X are called the message nodes and the right nodes are called the check nodes. We construct an LDPC from X in the following way: if we have a codeword w, we associate each left vertex of the graph with each bit of w; w is a codeword if, for all check nodes, the sum of the neighbor bits is zero.

In matrix terms, the parity check matrix H is the adjacency matrix of X. Since X is a sparse graph, H is a sparse matrix. Hence, the code is the set of words w such that $wH^T = 0$.

The great advantage of LDPC codes is their decoding algorithm. These codes can be decoded using belief propagation tehcniques which makes them extremely efficient [**?**, **?**, **?**, **?**, **?**].

LDPC codes differ greatly from algebraic geometry codes (like, for example, Goppa codes). For the later, it can be extremely difficult to find decoding algorithms, since they are constructed using combinatorial arguments and the decoding is not taken into account when creating the code. On the other hand, LDPC codes have very easy and fast decoding algorithms, since the decoding is taken into account when creating the code. Actually, we can say that LDPC codes are created based on an algorithm and so their biggest advantage is the fast decoding algorithms. These algorithms also scale very well for hardware implementation purposes [**?**, **?**, **?**, **?**].

LDPC codes should not be used per se in the McEliece cryptosystem [**?**]. The point in this paper is to combine it with Goppa code to achieve both security and scalability.

2.3. McEliece cryptosystem

The McEliece cryptosystem [**?**] is described in Algorithm 1.

The McEliece cryptosystem can be attacked in two different ways. Either the message encrypted is attacked, and the message is recovered from the ciphertext. Or the public-key is attacked, and the secret-key is obtained by attacking the public-key.

The security of the message in the McEliece cryptosystem relies on the Computational Syndrome Vector, which is a **NP**-complete problem [**?**].

Problem 1 (Computacional Syndrome Vector (CSV)). *Given a code $\mathcal{C} \in \mathbb{Z}_2^n$ of dimension $k \leq n$, a parity-check matrix H of $\mathcal{C}$, a syndrome $s \in \mathbb{Z}_2^r$ and $t \in \mathbb{N}$, find a word $x \in \mathbb{Z}_2^n$ such that $x \in S_H(s)^{-1}$ and x has weight $wt(x) \leq t$.*

Equivalently, the security of the message may rely on the Learning with Parity Noise, a classical problem in learning theory. We describe the problem following [**?**].

Algorithm 1 McEliece public key cryptosystem

Parameters: $n, t \in \mathbb{N}$ with $t << n$.

Key Creation: Choose a Goppa Code $\mathcal{G} \subset \mathbb{Z}_2$. This is a k dimensional code with length n that corrects t errors. Generate the matrices G, S and P where $G_{k\times n}$ is a generator matrix of $\mathcal{G}$, $S_{k\times k}$ is a non-singular matrix uniformly chosen and $P_{n\times n}$ is a permutation matrix uniformly chosen. Compute $G^{pub}_{k\times n} = SGP$.

Public key: (G^{pub}, t).

Private key: $(S, P, D_{\mathcal{G}})$ where $D_{\mathcal{G}}$ is a decoding algorithm for $\mathcal{G}$.

Encryption: Choose $e \in \mathbb{Z}_2^n$ randomly such that e has weight t. To encrypt a message $\mathbf{m} \in \mathbb{Z}_2^k$, one must compute the ciphertext $\mathbf{c} = \mathbf{m}G^{pub} + e$.

Decryption: To decrypt $\mathbf{c}$ one must compute $\mathbf{c}P^{-1}$ and then apply $D_{\mathcal{G}}$ to the result. Finally multiply by S^{-1} to get the message $\mathbf{m}$.

Problem 2 (Learning with Parity Noise (LPN))**.** *Let $l \in \mathbb{N}$. Let s (the secret) be a word in $\mathbb{Z}_2^l$. Given the pair*

$$(a, s.a + e)$$

where $a \in \mathbb{Z}_2^l$ and e is chosen using the Bernoulli distribution $\mathfrak{B}_\theta$ with parameter $\theta \in [0, \frac{1}{2}]$, find s.

The security of the public-key relies on the following problem, which is assumed to be hard [**?**].

Problem 3 (Goppa Distinguisher (GD))**.** *Given a matrix $M_{k\times n}$, output 1 if M represents a generating matrix of a Goppa code and output 0 if M was chosen uniformly at random (i.e., each coordinate of M was chosen uniformly at random).*

Given that the CSV (or, equivalently, the LPN) and the GD problems are hard, we can conjectured that the McEliece cryptosystem is secure.

There is no polynomial-time (classical or quantum) attack on the McEliece. But still, we would like to present some of the most famous attacks on the McEliece, namely the classical and quantum information set decoding attack.

The information set decoding (see [**?**] for a full description of the attack) is the most famous attack on the McEliece cryptosystem since the best known attacks to the cryptosystem are variants of this attack [**?**, **?**, **?**, **?**]. The complexity of the original information set decoding attack (ISD) is $\Omega\left(n^2 \frac{\binom{n}{k}}{\binom{n-t}{k}}\right)$ [**?**].

The complexity of the quantum set decoding attack (QISD) is $\Omega\left(n^2 \sqrt{\frac{\binom{n}{k}}{0.29^{\binom{n-t}{k}}}}\right)$ (see [**?**] fora full description of the attack).

In Table 1, we computed the key size of the McEliece for certain levels of security. The column security refers to the security against ISD. The column PQ security refers to the security against QISD. Although these are not the best attacks known, we will use these values to compare key size with the new cryptosystem. The security is measured in the logarithm (base 2) of the expected number of operations for the attack to be successful.

Parameters			2*Public key size (systematic)	2*S
n	k	t		
1024	524	50	32.8 KB	≈
2048	1696	32	74.6 KB	≈
4096	3616	40	217 KB	≈
6960	5413	119	1.05 MB	≈

Table 1: Parameters of the McEliece.

3. Proposal

Before delving into the formal construction, we give the main idea behind the construction. As it is well known, Goppa codes do not scale well since Patterson's algorithm is relatively inefficient for large block length messages [**?**]. To overcome this difficulty, we propose a cryptosystem where a message $\mathbf{m}$ is divided into several small blocks. Each one of these blocks will be encoded using a Goppa code. The resulting codeword, which is the concatenation of codewords of the chosen Goppa code, will then be encoded using an LDPC code. Errors will be added to the resulting codeword such that these errors can be corrected by the LDPC code and by each of the small blocks, using Patterson's algorithm. Note that, since these blocks are chosen to be relatively small, Patterson's algorithm is expected to be quite efficient.

We now formally present the public-key encryption scheme. Suppose a system with Alice and Bob, where Bob wants to send an encrypted message $\mathbf{m}$ to Alice using her public-key. Reading what follows with an eye on Figures 1, 3 and 2 might be useful.

Key Creation:

To create a pair of public and secret keys, Alice must choose a Goppa code $\mathcal{G}$, with generating matrix $G_{k\times n}$, that is able to correct t errors. Then she must randomly choose a non-singular square matrix

S, of size k, and a permutation matrix P, of size n. She computes $U_1 = SGP$, as in the original McEliece PKC. Now, let

$$U_1^{\oplus \ell} = \begin{pmatrix} U_1 & 0 & \dots & 0 \\ 0 & U_1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & U_1 \end{pmatrix}$$

where the matrix U_1 appears ℓ times in the diagonal of $U_1^{\oplus \ell}$. Note that $U_1^{\oplus \ell}$ is a matrix of size $k\ell \times n\ell$.

Now, Alice chooses a binary LDPC code $\mathcal{L}$ with generating matrix $\hat{G}$, a $n\ell \times m$ matrix, that is able to correct δ errors. She then chooses randomly another non-singular matrix $\hat{S}$, of size $n\ell \times n\ell$, and another permutation matrix, of size $m \times m$. She computes $U_2 = \hat{S}\hat{G}\hat{P}$. The matrix $U = U_1^{\oplus \ell} U_2$ will be part of her public-key.

To complete the key creation process, she is going to construct a list of vectors, that we will denote by A, in the following way: let $\{e_1, \dots, e_n\}$ be the canonical base of $\mathbb{Z}_2^n$. Let $\{e_1^j, \dots, e_\ell^j\}$ be a random permutation of ℓ uniformly chosen vectors of the canonical base. Set

$$v_j = (e_1^j | \dots | e_\ell^j) U_2$$

and repeat this process c times to create the list $A = \{v_1, \dots, v_c\}$, where $c \in \mathbb{N}$ is a constant such that $c >> t$. Note that each vector of A is the concatenation of l error vectors of size n and weight 1 that were expanded by U_2. To encrypt a message, Bob will choose error vectors of this list to create the ciphertext, as we will see later.

The public key consists of (U, A, t, δ). The private key consists of $(S, G, P, \hat{S}, \hat{G}, \hat{P})$ and the corresponding decoding algorithms for $\mathcal{G}$ and $\mathcal{L}$, which we will denote by $D_{\mathcal{G}}$ and $D_{\mathcal{L}}$ respectively.

Encryption:

To encrypt, Bob randomly chooses t vectors from A and XOR them. The result of this operation is

$$E = v_{i_1} + \dots + v_{i_t}.$$

The encryption of a block of message $\mathbf{m} = (\mathbf{m}_1 | \dots | \mathbf{m}_\ell)$ is obtained by computing

$$\mathbf{c} = \mathbf{m}U + E + r_\delta$$

where r_δ is random vector of $\mathbb{Z}_2^m$ with weight δ chosen by Bob. The factor $E + r_\delta$ represents the errors that will be corrected by the decoding algorithms (see Figure 2).

Decryption:

To decrypt a ciphertext $\mathbf{c}$, Alice first computes $\mathbf{c}\hat{P}^{-1}$ and applies the decoding algorithm $D_{\mathcal{L}}$ of the LDPC code to the result. Then, she multiplies the result by $\hat{S}^{-1}$ to obtain

$$\mathbf{m}U_1^{\oplus \ell} + \bigoplus_{j=1}^{t} (e_1^j \dots e_\ell^j) = \mathbf{c}'.$$

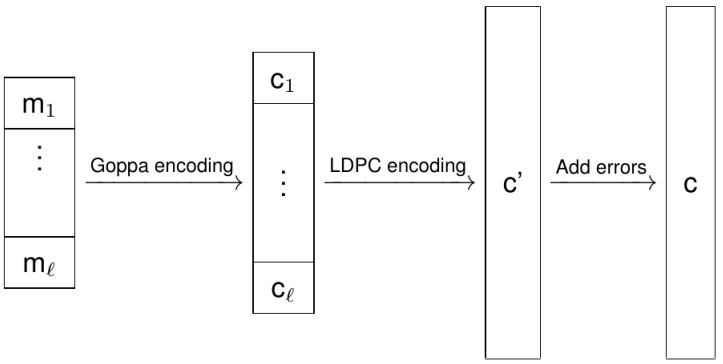

Figure 1: Encryption scheme

$$\begin{pmatrix} \\ r_\delta \\ \\ \end{pmatrix}^T + \begin{pmatrix} e_1^1 \\ \vdots \\ e_\ell^1 \end{pmatrix}^T U_2 + \dots + \begin{pmatrix} e_1^t \\ \vdots \\ e_\ell^t \end{pmatrix}^T U_2$$

Figure 2: Error construction scheme

Note that $\mathbf{c}'$ is just the concatenation of $\mathbf{c}_1, \dots, \mathbf{c}_\ell$ where each $\mathbf{c}_i$ is the encryption of $\mathbf{m}_i$ by the original McEliece. Then, Alice can use the decryption algorithm of the original McEliece to get $\mathbf{m}_i$ from $\mathbf{c}_i$ (i.e., multiply by P^{-1}, apply Patterson's algorithm to correct errors and multiply by S^{-1}). This task can be done in parallel, since each of the $\mathbf{m}_i$ is independent from the others. After she recovers all of the $\mathbf{m}_i$, she concatenates them to get $\mathbf{m}$. A scheme of the decryption algorithm is presented in Figure 3.

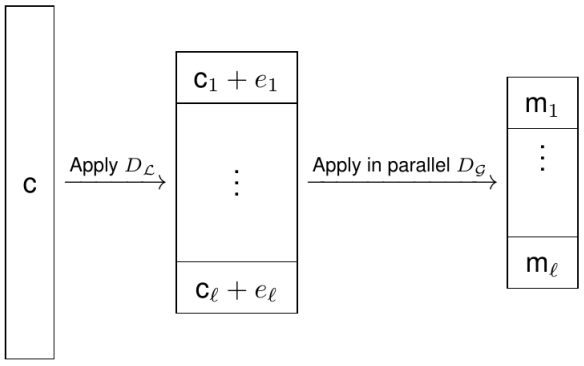

Figure 3: Decryption scheme: $D_{\mathcal{L}}$ and $D_{\mathcal{G}}$ denote the decoding algorithms of the LDPC and the Goppa codes, respectively.

The public-key cryptosystem is presented in Algorithm 2 in page 5. We will call **LDPC-based McEliece** to the above cryptosystem.

The reason for creating and publish the list of errors

Algorithm 2 LDPC-based McEliece cryptosystem

Security parameters: $n, m, t, \delta, \ell, c \in \mathbb{N}$.

Key Creation: Choose a Goppa Code $\mathcal{G}$ able to correct t errors using a decoding algorithm $D_{\mathcal{G}}$. This code has length n and dimension k. Choose an LDPC code $\mathcal{L}$ able to correct δ errors using a decoding algorithm $D_{\mathcal{L}}$. This code must have length m and dimension $n\ell$ Generate the matrices G, S, P, $\hat{G}$, $\hat{S}$ and $\hat{P}$ where:

$G_{k\times n}$ is a generating matrix of $\mathcal{G}$;

$\hat{G}_{n\ell\times m}$ is a generating matrix of $\mathcal{L}$;

$S_{k\times k}$ and $\hat{S}_{n\ell\times n\ell}$ are non-singular matrices randomly chosen;

$P_{n\times n}$ and $\hat{P}_{m\times m}$ are permutation matrices randomly chosen;

Compute $U_{k\times m} = U_1^{\oplus\ell}U_2 = (SGP)^{\oplus\ell}\hat{S}\hat{G}\hat{P}$;

Construct the list $A = \{v_1, \ldots, v_c\}$, where each $v_j = (e_1^j|\ldots|e_\ell^j)U^2$ and $\{e_1^j, \ldots, e_\ell^j\}$ is a random permutation of ℓ vectors of the canonical base of $\mathbb{Z}_2^{n\ell}$.

Public key: $(U, A = \{v_1, \ldots, v_c\}, t, \delta)$.

Private key: $(S, P, D_{\mathcal{G}}, \hat{S}, \hat{P}, D_{\mathcal{L}})$ where $D_{\mathcal{G}}$ is a decoding algorithm for $\mathcal{G}$ and $D_{\mathcal{L}}$ is a decoding algorithm for $\mathcal{L}$.

Encryption: Choose t vectors of A and sum them, the result of this operation is E. Choose $r_\delta \in \mathbb{Z}_2^m$ randomly such that r has weight δ. To encrypt a message $\mathbf{m} \in \mathbb{Z}_2^{k\ell}$ into a ciphertext $\mathbf{c}$, one must compute

$$\mathbf{c} = \mathbf{m}U + E + r_\delta.$$

Decryption: To decrypt $\mathbf{c}$, multiply $\mathbf{c}$ by $\hat{P}^{-1}$, apply $D_{\mathcal{L}}$ to take the error r_δ and multiply by $\hat{S}^{-1}$. We get $\mathbf{c}'$ after these operations. Then, in parallel, multiply each part of length n of $\mathbf{c}'$ by P^{-1}, apply the decoding algorithm $D_{\mathcal{G}}$ to take each error vector of weight t and multiply by S^{-1}. Concatenate each of the resulting $\mathbf{m}_i$ to get $\mathbf{m}$.

A (that will be corrected by the Patterson's algorithm) is that we do not have to publish the matrices U_1 and U_2. This is useful for two reasons: we had a little extra security since we strongly believe that it is hard to factorize U into two matrices U_1 and U_2 and, if U_1 and U_2 were public, the cryptosystem would be just a composition of cryptosystems.

The list of errors A that is made public can not have all errors of the form $(e_i^1|\ldots|e_i^\ell)U_2$, since this list grows exponentially (it takes time $\mathcal{O}(n^\ell)$ to produce this list). However, for practical purposes, we only need a constant number of them as long as this number is much greater than t (the number of errors corrected by the inner Goppa code). The owner of the public-key can update this part of the key from time to time, to avoid that the same error patterns are used in different encryptions.

There are two particular cases that we would like to reference. The first extreme case is when $\ell = 1$. In this case, the protocol is just the composition of two McEliece-based cryptosystems, the original and a variant using an LDPC code. In this particular case, we could find the factorization of U (which we want to keep secret) using the list A. Another particular case is when $l = |\mathbf{m}|$ (where $|\mathbf{m}|$ is the size of the message $\mathbf{m}$), which is not possible since there are no Goppa codes of size 1. These cases should be avoided.

The choice of the LDPC code is a crucial point in order to guarantee the efficiency of our proposal. On the one hand, the LDPC chosen must have a low decoding complexity. On the other hand, the LDPC must correct any pattern of errors of weight δ, for the sake of correctness of our cryptosystem. As far as we know, it seems that the only LDPC codes that have a decoding algorithm that guarantees a fraction of errors corrected are LDPC codes based on expanders [**?**, **?**, **?**, **?**].

Another aspect that we have to take into account while choosing the LDPC code is the redundancy of the code. Redundancy of the ciphertext is a problem that can not be avoided in the McEliece cryptosystem as we have seen before. But still, we should avoid adding to much redundancy in the ciphertext. For better information ratio, one must choose LDPC codes with the highest information ratio possible, without compromising the security.

4. Security analysis

In this section we present some arguments for the security of our cryptosystem. We prove in Propositions 4 and 5 that it is hard to break the LDPC-based cryptosystem, given that the original McEliece is hard to break. Also, we give some arguments on why we believe that it is hard to factorize the public-key of the cryptosystem.

Given this, we present the following lemma which proves that it is hard to recover messages from their

ciphertexts.

Proposition 4. *Breaking the security of the message of the LDPC-based McEliece cryptosystem is at least as hard as breaking the security of the message of the original McEliece.*

We have proved that it is hard to recover the message from the ciphertext. So now we will talk about the security of the public-key.

Proposition 5. *Breaking the security of the public-key of the LDPC-based McEliece cryptosystem is at least as hard as breaking the security of the public-key of the original McEliece.*

Although recovering the secret-key is hard (given that breaking the McEliece cryptosystem is hard), it may be a problem if an adversary is able to factorize the public-key U. Recall that the public-key is $(U, A = \{v_1, \ldots, v_c\}, t, \delta)$ where U is a matrix with $k\ell$ lines and m columns and it is the product of $U_1^{\oplus \ell}$ (a $k\ell \times n\ell$ matrix) by U_2 ($n\ell \times m$ matrix). Since $U_1^{\oplus \ell}$ represents the concatenation of ℓ original McEliece cryptosystem (whose public-key is indistinguishable from a random matrix), the real menace to the cryptosystem is if an adversary could find the which LDPC is being used. If she could do that, then breaking the cryptosystem would be equivalent to break the original McEliece for small parameters.

To find the LDPC code used in the cryptosystem, an adversary would need to factorize the matrix U. Note that some information is leaked from the list A, namely the parameters of the LDPC code used. To find the parameters, an adversary can proceed in the following way: The adversary chooses a vector from the list A. Then she chooses another vector from A that is linearly independent from the first chosen vector. Again, she chooses another vector that is linearly independent from the first two chosen vectors. She does this until there is any linearly independent vector to choose. Most likely, she will get a set of $n\ell$ vectors linearly independent since each vector

$$v_j = u_j U$$

where $u_j = (e_1^j | \ldots | e_e ll^j)$ is a vector of size $n\ell$. Let $\{v'_1, \ldots, v'_{n\ell}\}$ be the list obtained from this procedure. The matrix

$$U'_2 = \begin{pmatrix} - & v'_1 & - \\ & \vdots & \\ - & v'_{n\ell} & - \end{pmatrix}$$

is a change of basis of the matrix U_2. With this, the adversary can get the value $n\ell$ and, thus, she gets the parameters of the LDPC code, which is a $n\ell$ dimensional code of length m. But note that she has no more information, particularly on the new basis, since she does not know which basis $\{u'_1, \ldots, u'_{n\ell}\}$ was used to construct the list of vectors $\{v'_1, \ldots, v'_{n\ell}\}$. So, she can not recover U_2 from her matrix U'_2.

Another threat to our cryptosystem is the attack presented in [**?**]. But note that this attack only works if the parity-check matrix of the LDPC code is made public, which is not true in our approach.

With this in mind, we conclude that every known attack on the cryptosystem does not work. Even if an adversary was able to somehow find the matrix U_2 or the corresponding parity-check matrix, she would still have to find the number of blocks ℓ and to break ℓ McEliece ciphertexts. This lead us to strongly believe that it is hard to recover the secret-key from the public-key. Given this, we can state that our cryptosystem is secure.

Note that the cryptosystem must also be robust against quantum attacks, otherwise we could quantumly attack the original McEliece as well. Also, since our protocol is based in the McEliece, we strongly believe that every attack on the McEliece (and every way to defend it from attacks) can be adapted to our cryptosystem.

If instead of using an LDPC code, we use a Goppa code (with larger parameters) we can prove that it is hard to factorize the public-key. To prove that, we assume that the Goppa Distinguisher problem is hard and, therefore, we cannot distinguish that public-key from a uniformly chosen matrix. But using a Goppa code may compromise the efficiency of the cryptosystem.

5. Efficiency

In this section we will analyze the circuit complexity of the LDPC-based McEliece cryptosystem. The following result gives us an upper-bound for the circuit size complexity of a program from its time complexity.

Lemma 6 ([**?**]). *Let $T(n) : \mathbb{N} \to \mathbb{N}$. If A is a program that runs in time $\mathcal{O}(T(n))$ then A can be implemented in a circuit of size $\mathcal{O}\left(T(n)^2\right)$ where n is the size of the input.*

The proof of this theorem is done by implementing each instruction of A sequentially in a circuit. So, the theorem tells us that A can be implemented in a circuit of size (and depth) $\mathcal{O}\left(T(n)^2\right)$ but does not guarantee that there is not a circuit that computes A with a smaller size and depth, i.e., it gives us an upper-bound for the circuit size. A smaller circuit may possibly be constructed by optimizing the number of gates or parallelizing some instructions of A for smaller depth complexity.

Before we start the analysis, recall that our cryptosytem uses a lot of matrix operations, so we prove the following lemma that gives us the circuit complexity of multiplying two matrices.

Lemma 7. *Let $\mathcal{C}$ be a circuit that computes the product of two matrices $A_{n\times k}$ and $B_{k\times m}$. Then $\mathcal{C}$ has size $\mathcal{O}(nkm)$ and depth $\mathcal{O}(\log k)$.*

Let us start by analyzing the circuit complexity of the encryption.

Proposition 8. *The encryption algorithm of the LDPC-based McEliece cryptosystem can be implemented in a circuit of size $\mathcal{O}(mk\ell)$ and depth $\mathcal{O}(\log k\ell)$ plus the circuit size and depth of choosing t random vectors of A and a random vector of size m and weight δ.*

The next theorem gives us the circuit complexity of decrypting a ciphertext.

Proposition 9. *The decryption algorithm of the LDPC-based McEliece cryptosystem can be implemented in a circuit of size*

$$\mathcal{O}\left(m^2 + (n\ell)^2 + (ntp^2)^2 + k^2 + \mathcal{S}^{\mathcal{D}}_{LDPC}\right)$$

and depth

$$\mathcal{O}\left(\log(mn^2\ell k) + (ntp^2)^2 + \mathcal{DS}^{\mathcal{D}}_{LDPC}\right)$$

where $\mathcal{S}^{\mathcal{D}}_{LDPC}$ is the circuit size of the decoding algorithm of the chosen LDPC and $\mathcal{DS}^{\mathcal{D}}_{LDPC}$ its depth.

Note that, as long as we keep the parameter n (and consequently t and p) when choosing the Goppa code to use, the cryptosystem has a very low depth complexity and should be quite efficient.

6. Security parameters and key size

We will analyze the parameters to use and the security expected of the cryptosystem taking into account the classical and the quantum information set decoding attacks. Recall that, for a k dimensional code of size n that corrects t errors, the complexity of the classical information set decoding attack (ISD) is

$$\Omega\left(n^2\frac{\binom{n}{k}}{\binom{n-t}{k}}\right)$$

and the complexity of the quantum information set decoding (QISD) attack is

$$\Omega\left(n^2\sqrt{\frac{\binom{n}{k}}{0.29\binom{n-t}{k}}}\right).$$

If one wants to use the ISD attack to our cryptosystem, note that one needs to use it to decode the LDPC code and then to decode each of the Goppa blocks. So, the complexity of this attack to our cryptosystem is $\Omega\left(m^2\binom{m}{n\ell}/\binom{m-\delta}{n\ell}\right)$, which corresponds to decode the LDPC code, plus $\Omega\left(\ell n^2\binom{n}{k}/\binom{n-t}{k}\right)$, which corresponds to decode each of the Goppa blocks. The total complexity of the attack is

$$\Omega\left(m^2\frac{\binom{m}{n\ell}}{\binom{m-\delta}{n\ell}} + \ell n^2\frac{\binom{n}{k}}{\binom{n-t}{k}}\right).$$

For a choice of parameters $m = 1024$, $n = 64$, $\ell = 12$, $k = 30$, $\delta = 70$, $t = 4$ the expected number of operations of an ISD attack is roughly 2^{181}. For these parameters, the key size (in the systematic form) is $(m - k\ell) \times k\ell \approx 30KB$ (kilobyte), which is much smaller than the McEliece public-key (roughly 10 times smaller for the same level of security).

Analogously, the complexity of the QISD attack for our cryptosystem is

$$\Omega\left(m^2\sqrt{\frac{\binom{m}{n\ell}}{0.29\binom{m-\delta}{n\ell}}} + \ell n^2\sqrt{\frac{\binom{n}{k}}{0.29\binom{n-t}{k}}}\right).$$

For the same choice of parameters above, the cryptosystem has a quantum security of approximately 107 bits.

With this in mind, we can estimate parameters for our cryptosystem. Those estimations are presented in Table 6.

Parameters						2* ublic key size (systematic)	2*Security	2
m	n	ℓ	k	δ	t			
1024	64	12	30	70	4	30 KB	≈ 172	
1000	12	50	4	60	2	20 KB	≈ 103	

Table 2: Parameters for the LDPC-based McEliece cryptosystem.

To maximize the security we need to enlarge the size of the Goppa blocks. This leads to a increase in the number of operations for the information set decoding attack. But, as we have seen in the previous section, if each of the Goppa blocks become to large, the cryptosystem is not so efficient. So, there is a trade off between security and efficiency: if one needs efficiency then it is better to choose smaller Goppa blocks; if one is looking for maximizing the security, one needs to choose larger Goppa blocks.

7. An IND-CCA2 secure version

In this section we will work exclusively in the random oracle model.[1] We begin by the definition of IND-CPA and IND-CCA2 security and PA security. Next, we present a relevant result on the indistinguishability of the McEliece cryptosystem.

Definition 10. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary to a cryptosystem $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Consider the IND-CPA game:*

1. *A pair of public and secret key is created using $\mathcal{K}$. $\mathcal{A}$ has access to pk;*
2. *$\mathcal{A}$ can do a polynomial number of queries to an encryption oracle;*

[1] In the random oracle model, we assume the existence of an ideal hash, i.e., a hash function that returns truly uniformly random numbers.

3. *$\mathcal{A}$ submmits two messages $\mathbf{m}_0$ and $\mathbf{m}_1$;*

4. *A bit b is chosen randomly from $\{0,1\}$ and $\mathbf{c} = \mathcal{E}(pk, \mathbf{m}_b)$ is sent to $\mathcal{A}$;*

5. *Again, $\mathcal{A}$ can do a polynomial number of queries to an encryption oracle;*

6. *$\mathcal{A}$ outputs $\bar{b}$, a guess for b.*

The IND-CCA2 games is similar with an extra condition: $\mathcal{A}$ can do a polynomial number of queries to a decryption oracle, except that she can not aks for the decryption of $\mathbf{c}$. Given any of these games, we formally define the advantage of the adversary $\mathcal{A}$ in the following way:

$$\mathtt{Adv}_{ind-aaa}^{\Pi,\mathcal{A}} = 2.Pr\left(\bar{b} = b\right) - 1$$

where $aaa = cpa$ if $\mathcal{O} = \epsilon$ and $aaa = cca2$ if $\mathcal{O} = \mathcal{D}(sk, \cdot)$. The advantage measures the adversary's chances of winning the game.

Definition 11 (Plaintext Awareness [**?**])**.** *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, $\mathcal{A}$ an adversary, $\mathcal{H}$ a cryptographic hash function and K an algorithm called the* knowledge extractor*. Consider the following game between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.*

- *$\mathcal{C}$ creates a pair of keys $(pk, sk) = \mathcal{K}(1^k)$. He publishes pk.*

- *$\mathcal{A}$ has access to the oracle $\mathcal{H}$ and to a encryption oracle. She can do a polynomial number of queries to each oracle. Eventually, she outputs (τ, η, y) where $\tau = \{(h_1, \mathcal{H}_1), \ldots, (h_{q_H}, \mathcal{H}_{q_h})\}$ is a list of queries $h_1, \ldots, h_{q_H}$ to the $\mathcal{H}$ oracle and the corresponding answers $\mathcal{H}_1, \ldots, \mathcal{H}_{q_H}$, $\eta = \{y_1, \ldots, y_{q_E}\}$ is a list of answers received by $\mathcal{A}$ of the queries done to the encryption oracle and $y \notin \eta$.*

- *$\mathcal{C}$ uses algorithm K, that receives as input (τ, η, y, pk), to try to find the decryption of y.*

We define

$$\mathtt{Adv}_{pa}^{K,\Pi,\mathcal{B}}(k) := Pr[K(\tau, \eta, y, pk) = \mathcal{D}(sk, y)].$$

We say that K is a $\lambda(k)$-knowledge extractor if K runs in polynomial time and $\mathtt{Adv}_{pa}^{K,\Pi,\mathcal{B}}(k) \geq \lambda(k)$.

If Π is IND-CPA secure and there exists a $\lambda(k)$-knowledge extractor K, where $1 - \lambda(k)$ is a negligible value, then we say that Π is plaintext aware (PA) secure.

Theorem 12 ([**?**])**.** *If a public-key cryptosystem is PA secure then it is IND-CCA2 secure.*

To our purpose, note that the McEliece cryptosystem is not IND-CCA2 secure [**?**]. In fact, it is not even IND-CPA secure [**?**]. To construct a IND-CPA secure version of the McEliece, one just needs to pad a random string to the message and encrypt this pad. This is called the randomized McEliece cryptosystem, which was presented in [**?**].

Theorem 13 ([**?**])**.** *The randomized McEliece is IND-CPA secure given that the LPN and the GD problems are hard.*

Proposal:

We now present an IND-CCA2 variant of the LDPC-based McEliece cryptosystem.

To achieve the IND-CCA2 property, we will take out the malleability of the cryptosystem. The idea is very simple: the error that will be corrected by the LDPC (denoted by r_δ, just like in the previous section) that will be used in the construction of a ciphertext will be chosen accordingly to a cryptographic hash function. If we do that, and given a ciphertext, it becomes impossible to create other valid ciphertexts from a valid one, which is the idea of the IND-CCA2 attack.

We choose a cryptographic hash function $\mathcal{H}$ such that $\mathcal{H}$ maps inputs of arbitrary size to a fixed size output. Let $K = \binom{m}{\delta}$, where δ is the number of errors corrected by the LDPC code of length m in the cryptosystem[2] and let us choose $\mathcal{H}$ such that its output set is $\{1, \ldots, K\}$. We define a new function $\mathcal{H}^e$ using $\mathcal{H}$:

$$\mathcal{H}^e(x) := \mathtt{Conv}\left[\mathcal{H}(x)\right]$$

where $\mathtt{Conv}$ is a bijection between $\{1, \ldots, K\}$ and the error vectors of size m and weight δ. Note that, as m grows, K grows as well. So the probability of finding collisions for this function is asymptotically impossible.

We now formally present the construction. When we encrypt a message $\mathbf{m}$, first we choose a random value s of fixed size and compute

$$r = \mathcal{H}^e(\mathbf{m}|s).$$

Also, instead of just encrypting $\mathbf{m}$, we will encrypt $\mathbf{m}|s$. This will be the error vector to be added to the ciphertext in the encryption algorithm and that will be corrected by the LDPC decoding algorithm. In the decryption, one just has to verify if $r_\delta = \mathcal{H}^e(\mathbf{m}|s)$. If so, the ciphertext was a valid one. If not, then the ciphertext is rejected as invalid.

Algorithm 3 briefly describes the IND-CCA2 version of the LDPC-based McEliece.

This protocol is similar to the Fujisako-Okamoto generic conversion [**?**]. The difference is that in our conversion only part of the randomization of the cryptosystem is determined by the cryptographic hash

[2] Once again, we see the importance of choosing an LDPC that corrects a constant fraction of errors.

Algorithm 3 IND-CCA2 version of LDPC-based McEliece

Security parameters: Same as in LDPC-based McEliece cryptosystem plus a cryptographic hash functions $\mathcal{H}$.

Key Creation: Same as in LDPC-based McEliece.

Encryption: Choose a random string s and encrypt $\mathbf{m}|s$ using the encryption algorithm of the LDPC-based McEliece, except that the random vector of weight δ will not be chosen randomly, but accordingly to $\mathcal{H}$, i.e., $r_\delta = \mathcal{H}^e(\mathbf{m}|s)$. The ciphertext will be $\mathbf{c} = (\mathbf{m}|s)U + E + \mathcal{H}^e(\mathbf{m}|s)$.

Decryption: Apply the decryption algorithm of the LDPC-based McEliece to get $\mathbf{m}|s$. Compute the error vector $r_\delta = \mathbf{c} + (\mathbf{m}|s)U + E$. Check if $\mathbf{c}$ is a valid ciphertext by checking if r_δ is equal to $\mathcal{H}^e(\mathbf{m}|s)$. If it is not, reject $\mathbf{c}$.

function while, in the generic conversion by Fujisako and Okamoto, all the randomization is determined by the cryptographic hash function.

We will now prove that this algorithm is indeed IND-CCA2 secure. First, we show that it is IND-CPA secure.

Proposition 14. *The LDPC-based McEliece cryptosystem is IND-CPA secure in the random oracle model given that the randomized McEliece is IND-CPA secure. Thus, it is IND-CPA secure given that the LNP and the Goppa Distinguisher problems are hard.*

Proposition 15. *For any q_H, there exists a (λ)-extractor K for the IND-CCA2 version of the LDPC-based McEliece, where $1 - \lambda$ is a negligible value.*

We are now able to state that this version of the cryptosystem has the IND-CCA2 property, by plugging the two previous theorems and noting that PA-security implies IND-CCA2 security (see Theorem 12).

Theorem 16. *The IND-CCA2 version of the LDPC-based McEliece is IND-CCA2 secure given that the randomized LDPC-based McEliece is IND-CPA secure.*

Acknowledgements
The author would like to thank ...

References

[1] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.